Holistic Methods for Visual Navigation of Mobile Robots in Outdoor Environments

Computer Engineering Group
Faculty of Technology
Bielefeld University

A thesis submitted
for the degree of Doctor of Engineering
by

Dario Differt

May 2017

Thesis Supervisor:	Prof. Ralf Möller
External Examiner:	Dr. Wolfgang Stürzl
External Examiner:	Prof. Dr. Matthias Franz

Revised version, July 2017
Gedruckt auf alterungsbeständigem Papier gemäß ISO 9706

Abstract

Encouraged by the growing market of self-driving cars, the research interest in the field of visual navigation in outdoor environments strongly increases. Even though recent development shows that in the near future road transport may drastically change, it conceals that these developments are restricted to a limited class of visual navigation problems. Methods which show reliable performance to navigate cars on streets may break as soon as their constraints are violated. For example, visual navigation methods suitable for self-driving cars can likely not be used to automate the navigation of mobile robots, e.g. lawn-mower robots. Mobile robots designed for domestic tasks need to be small, light-weight, and cheap to produce; all factors effectively limiting the number and quality of sensors usable for visual navigation. Moreover, mobile robots can be exposed to various challenges as for example strong illumination changes, dynamic scenes with moving objects, and non-planar movement.

In this thesis we aim to develop visual navigation methods suitable for navigation in challenging outdoor environments. Due to their high amount of visual information, we primarily use panoramic images captured by fish-eye cameras. To maximize the information utilization, we use holistic methods which perform pixel-wise comparisons between panoramic images rather than selecting individual features. While RGB camera images provide both brightness and color information, these information dependent on the illumination conditions, for example, images captured at the same location but at different daytimes may look different. Inspired by the study of social insects such as ants, bees, and wasps, we examine approaches to extract the skyline — a binary image where each pixel is classified either as ground object or sky — using either only ultraviolet input or contrast color vision between ultraviolet and green light. To represent panoramic images (e.g. the skyline) directly in the frequency domain, we use the basis of real spherical harmonics. Furthermore, we suggest multiple concepts which are important for developing visual navigation methods directly in the basis of spherical harmonics, including techniques to use hemispherical panoramic images instead of full-spherical panoramic images and the derivation of sparsity relations to efficiently compute rotations.

These two topics — extracting the skyline and using the basis of real spherical harmonics to represent panoramic images — form the theoretical framework from which we derive three methods for visual navigation in outdoor environments: First, we suggest a method for visual localization which uses the amplitude spectrum of the skyline to uniquely describe locations. Second, we propose the visual 3D compass to rotationally align panoramic images. Third, we generalize a visual homing method called warping for non-planar movement. All methods are designed for wheeled robots which are exposed to strong illumination changes and tilt.

The overall results of this theses reveal that panoramic images are a viable source of information for visual navigation in outdoor environments. Moreover, we could show that visual localization and homing can be performed relying solely on the skyline as visual input instead of RGB camera images. This makes the skyline interesting for both robot applications and models of insect navigation. Furthermore, the basis of real spherical harmonics allows us to appropriately represent panoramic images and formulate computationally efficient methods. Requiring only a single fish-eye camera, our methods are feasible for mobile robots which have strict limitations on weight and size.

Acknowledgements

At the time when I was a undergrad, I managed to miss the deadline to sign in for a robot-themed seminar but nevertheless decided to attend to it. Even though the maximal number of participants was reached, I was — to my surprise — not thrown out of class. Looking back, I might have never gotten in touch with the interesting and fulfilling subject of robotics without the seminar's host's generosity. Therefore I want to thank Prof. Dr. Wolfram Schenck for giving me the possibility to gather first experiences in the field of robotics. After I graduated, it was also Wolfram who told me about the open PhD position at the Computer Engineering Group in the Faculty of Technology at Bielefeld University.

Over the last four years when I was working on my dissertation, many people supported me and were crucially involved in its completion. First of all, I would like to thank my supervisor Prof. Dr. Ralf Möller for the endless support, discussions, and valuable feedback on my work. He gave me the opportunity to work in application-oriented areas in which I learned a lot; interestingly even about problems which to me, as a former theorist, would have never occurred. Moreover, I want to thank the entire Computer Engineering Group, Angelika Deister, David Fleer, Dr. Lorenz Hillen, Annika Hoffmann, Michael Horst, Dr. Alexander Kaiser, and Constanze Schwan, for their help, advice, and support during work, but also the chatter and trips to the climbing hall in our spare time. Special thanks go to Klaus Kulitza who helped me to craft various experimental setups. Furthermore, I want to thank Dr. Wolfgang Stürzl for the help about any camera and camera lens related problems as well as his inspiring ideas which alone could fill another dissertation.

I want to thank my former fellow students with whom I spent some of the most wonderful time as an undergraduate and graduate student. Moreover, I want to thank all the people who continuously distracted me from writing my dissertation. In a positive way. To name a few, these are my handball team and football group which incorporated me throughout the last years and my friends who know how to enjoy some quality time. Finally, I want to thank my family for their support throughout my life and Dominique for her love and perseverance to listen to all my little robot-themed problems.

Table of Symbols

Symbol	Description
x	Variable, constant
$\bar{x}$	Complex conjugate (for scalars, vectors, etc.)
$\vec{x}$	Vector
$\mathbf{X}$	Matrix
$\mathbf{X}^T$	Transpose
$\mathbf{X}^\dagger$	Conjugate transpose
$\mathbf{R}_{\vec{v},\alpha}$	Rotation matrix around axis $\vec{v}$ by angle α
$f \circ g$	Concatenation of two functions, i.e. $(f \circ g)(x) = f(g(x))$
$f \cdot g$	Point-wise product of two functions, i.e. $(f \cdot g)(x) = f(x)g(x)$
$\langle x, y \rangle$	Standard scalar product
$\mathbf{X} \bigoplus \mathbf{Y}$	Direct sum (definition 3.3)
$\mathbf{X} \bigotimes \mathbf{Y}$	Kronecker product (definition 3.5)
$\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{C}$	The sets of natural, integer, real, and complex numbers
$\mathrm{Mat}(m \times n, \mathbb{R})$	The set of all matrices with dimension $m \times n$ and entries from $\mathbb{R}$ (equivalently for $\mathbb{C}$)
$\mathrm{Mat}(n, \mathbb{R})$	The set of all matrices with dimension $n \times n$ and entries from $\mathbb{R}$ (equivalently for $\mathbb{C}$)
S^2	The set of all points on the unit sphere
$SO(3)$	The rotation group in $\mathbb{R}^3$, here represented by the set of rotation matrices
$\delta_{x,y}$	Kronecker delta

Contents

CHAPTER 1

Introduction

With a rapidly growing market for self-driving cars — current estimates expect that around 15% of all cars sold in the year 2030 will be self-driving (Kaas et al., 2016) — the public interest in autonomous navigation increases noticeably. However, not only the car market seems to be on the verge: Despite rapid advances in the development of mobile robots as for example service, cleaning, or lawn-mower robots, the demand on the consumer market was negligible (Elkmann et al., 2009, Prassler et al., 2016). Over the last years — maybe due to the increased public awareness of self-driving cars — the market got traction and forecasts report that it will grow massively in the next decade (Business Insider, 2015, The Robot Report, 2016).

In most applications both self-driving cars and mobile robots have to perform comparable navigational tasks (e.g. drive towards a goal location) without human intervention. However, the prerequisites for self-driving cars and mobile robots differ strongly: Self-driving cars are able to carry a wide range of sensors — for example a speedometer, global positioning system (GPS), light detection and ranging (LIDAR), inertial measurement units (IMU), video camera, or ultrasonic sensors — and the fusion of these sensor data (on powerful on-board computers) provides a nearly complete representation of the car's surrounding. This high amount of accessible information allows the implementation of sophisticated methods for autonomous navigation. In contrast, mobile robots are more restricted in their size, weight, and total costs. Preferably only a low number of small, lightweight, and low-cost sensors is used, increasing the difficulty to represent the robot's surrounding. This lack of information increases the difficulty to reliably perform navigational tasks. In this work we aim to derive methods for autonomous navigation of mobile robots using a minimum of sensory input; here only a single fish-eye image.

In this chapter we introduce basic concepts used throughout this work: After giving a brief introduction into the field of autonomous navigation in section 1.1, we discuss in section 1.2 visual navigation with a focus on localization and homing. Since we use panoramic camera images for visual navigation throughout this work, we present in section 1.3 different types of omnidirectional cameras which can be used to capture panoramic images. Finally, we give in section 1.4 a concise outline of this work.

1.1 Autonomous Navigation

The term autonomous navigation describes a wide field of techniques and methods which reduce the necessity of human interaction to control air, water, or ground robots. With an increasingly better access to highly accurate sensors as for example cameras, IMU, or LIDAR, the field of autonomous navigation is rapidly growing in recent years. A search on `https://scholar.google.de/` for the term 'autonomous navigation' gives an increasing number of matches in recent years; for example in each of the years 2000, 2005, 2010, and 2015 a total number of 650, 1420, 2420, and 3430 works were published.

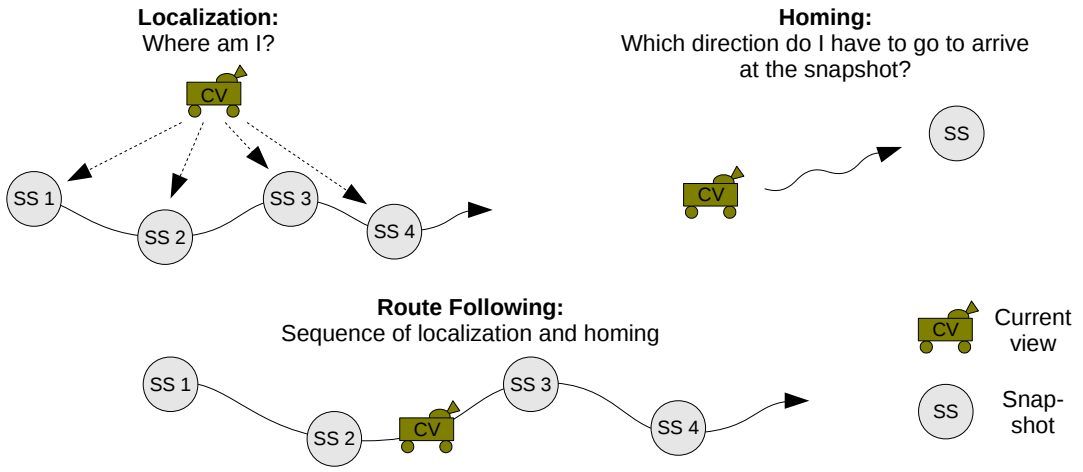
In this work, we are less interested in developing high-level navigation strategies as for example required for optimal room coverage in domestic cleaning robots. Instead, we discuss methods for rather basic visual navigation tasks which could later be used in more complex navigation

strategies: Localization (section 1.2.1), homing (section 1.2.2), and route following (section 1.2.3).

In general, arbitrary sensors can be used to obtain abstract representations of the robot's surrounding. Therefore we refer to sensory input or a (commonly more sparse) representation of it as **scene descriptors**. The term scene descriptor is chosen on purpose in an unspecific way to underline that arbitrary information can be used; often methods can be adapted to work with arbitrary scene descriptors (e.g. seqSLAM, section 4.3.3). For our purpose, scene descriptors are mostly panoramic camera images (section 1.3) or the binary skyline image (chapter 2).

All methods in this work aim to find a correspondence between the scene descriptors captured at the current robot location (**current view**) and at some previously visited location (**snapshot**). For example, for the task of visual localization we aim to determine the robot's location by comparing its current view with all snapshots captured at previously visited locations. In the following sections, we briefly introduce visual navigation, visual homing, and path following, and give an overview over commonly used methods.

Example 1.1: Autonomous Navigation



Two fundamental questions of autonomous navigation are 'Where am I?' (localization, section 1.2.1) and 'Which direction do I have to go to arrive at an already known snapshot?' (homing, section 1.2.2). By repeatedly solving these questions, a robot is able to follow a previously driven route (route following, section 1.2.3).

1.2 Visual Navigation

Navigation which solely relies on camera input is called **visual navigation**. There are various visual navigation problems, including loop-closure (Labbe and Michaud, 2013, Cummins and Newman, 2007), room segmentation (Bormann et al., 2016), visual odometry (Corke et al., 2004, Nistér et al., 2006), and simultaneous localization and mapping (Tardif et al., 2008, Konolige and Agrawal, 2008). In this work, we focus on the problems of visual localization (section 1.2.1), visual homing (section 1.2.2), and route following (section 1.2.3). Often these fields overlap and an exact distinction is not possible. Moreover, related fields as for example shape alignment (Ren et al., 2014, Collet et al., 2009) can be used for visual navigation (e.g. aligning camera images).

1.2.1 Visual Localization

By scanning the environment for obstacles, a robot is able to drive in a direction without inflicting collisions, however without any contextual knowledge of its environment — i.e. a **map** (section 1.2.3) — more complex navigational tasks as for example driving towards a goal location cannot be performed. However, even if a map of the robot's **working area** — that is the area in which the robot operates — is available, the robot needs to localize itself first to make use of it.

By manually performing training runs with the robot or by letting the robot explore its environment automatically, scene descriptors from different places in the robot's working area can be collected and stored in a map (**mapping**). As soon as a map is available, the robot can use it to localize itself. Methods which perform *simultaneous localization and mapping* are called **SLAM** methods; a survey of current SLAM methods can be found in Fuentes-Pacheco et al. (2015). Generally, localization and mapping methods can be divided into two categories: First, **appearance-based methods** use image patches as scene descriptors and do not rely on spatial geometries. Examples are holistic methods (Möller et al., 2010), but also feature-based methods (Booij et al., 2007, Cummins and Newman, 2008); a more extensive introduction to both can be found in section 1.2.2. Second, **metric-based methods** identify distinct image features, e.g. using feature-based methods, and relate them with known image features (Nistér et al., 2006, Haralick et al., 1994). A review on visual localization can be found in Lowry et al. (2016). While appearance-based methods are used to build topological and topometrical maps in the robot's working area, metric-based methods are used to build metric maps (section 1.2.3).

Now we assume the following scenario for appearance-based localization: The robot performed a training run and built up some map by collecting scene descriptors in its working area. Afterwards, the robot is placed at a randomly chosen location of its working area[1]. Before the robot is able to drive towards a designated goal location in the map, it has to localize itself in the map first. During the training run the robot stored a scene descriptor at each visited location (**snapshots**). The most simple form of localization is to compare the scene descriptor at the robot's current location (current view) and with all previously stored snapshots. Choosing the most similar snapshot, the robot's location in its map can be estimated. This rather simple approach has to deal with various challenges, especially if camera images are used:

- **Translational and rotational offsets**: Commonly the current view is not recorded at the exact same location and/or in the same orientation as the snapshots. A pixel-wise comparison between the current view and the closest snapshot is therefore not successful. In the worst case, a snapshot recorded at an entirely different location might look more related to the current view than the snapshot captured at the closest location. This can be avoided by using a rotation-invariant or translation-invariant distance measure for image comparison (Menegatti et al., 2004, Simard et al., 1998). However, the amount of information used in rotation-invariant or translation-invariant distance measures is often decreased. As a consequence, images captured at different locations may appear more similar. Another approach is to rotationally align the current view with each snapshot previous to each comparison using external information (e.g. an IMU, Zsedrovits et al. (2014)) or a visual compass (section 1.2.4).
- **Dynamic scenes**: Images captured at the same location but on different times often significantly differ: The visual appearance of a scene can change due to illumination changes (outdoor: sun movement, weather changes; indoor: lights turned on or off) or changes of the scene's geometry (pedestrians, appearance or disappearance of obstacles). By preprocessing images these kind of effects can be reduced, for example by removing shadows from images (Corke et al., 2013, Maddern et al., 2014) or identifying moving objects in the scene (Wang et al., 2003).
- **Ambiguities**: Environments with repetitive structures such as hallways or forests often provide only few distinguishable features. By comparing the current view with all snapshots, multiple matches can be found. One way to encounter ambiguities is to compare sequences of current views and snapshots (Milford and Wyeth, 2012). A problem with sequence-based approaches is that they often assume that the robot drives along a path without junctions

[1] Placing a robot at a random new location without any indication is a challenging task for most localization algorithms which rely on previous location estimates. This problem is sometimes referred to as the **kidnapped robot problem** (Choset et al. (2005), section 9.1).

(route following, section 1.2.3), otherwise the effort to compare all possible sequences increases due to the curse of dimensionality (Mount and Milford, 2016). Another way is to use multiple location hypotheses simultaneously (Dellaert et al., 1999): By updating the beliefs for multiple hypotheses over time, wrong hypotheses are discarded as soon as they do not comply with the sensory data anymore.

- **Inaccurate estimates**: Even for the case that the snapshot closest to the current view could be found, the robot's location can only be estimated. Based on the density and coverage of map locations, the maximal accuracy of the location estimation is limited. The accuracy can simply be increased by collecting snapshots at more locations, however the time needed to perform the training increases as well as the memory storage size. To overcome the latter, a priority in creating scene descriptors is to reduce their memory usage as far as possible, for example by shrinking the images (Milford, 2013).

Various methods have been suggested to perform visual localization on images. These can generally be divided in the three classes; for a comparison of various visual localization methods see Horst and Möller (2017):

Holistic methods perform a pixel-wise comparison between the current view and all snapshots and search for the best match by minimizing some image distance measure. An advantage of holistic methods is that the complete image information is used such that interferences, as for example pedestrians or moving objects, commonly have a small impact on the overall image comparison. However, illumination changes which affect the whole image can cause discrepancies between current views and snapshots captured at the same location but at different dates. Applying preprocessing, as for example edge-filtering, the effect of illumination changes can be reduced. Another weak point of holistic methods is that rotations of the robot between current views and snapshots need to be corrected. Therefore the current view and snapshot need to be rotationally aligned before they can be compared, a problem which can be solved using a visual compass (section 1.2.2.1, section 1.2.4). Holistic methods are introduced in more detail in section 1.2.2 for the task of visual homing.

Instead of using a complete image as scene descriptor, **signature-based methods** extract and use selected information — as for example color histograms (Werner et al., 2007) or amplitude spectra (Menegatti et al., 2004) — from each image. The information vector containing the extracted information is referred to as *signature*, which allows for fast comparison between current views and snapshots by simply computing their difference using some vector norm. Using appropriate signatures, rotation-invariance can be obtained or interfering influences as for example illumination changes can be reduced (Hillen, 2013, Zhou et al., 2003). Depending on the complexity of the signatures, their extraction can be time-consuming.

A special case of signature-based methods are **feature-based methods**. Instead of extracting a signature for the complete image, a signature is extracted for set of salient pixels in each image (*features*). The features are chosen using a *feature detector* and the signature for each feature is created using a *feature descriptor*. By *matching* similar features in current views and snapshots it can be estimated if the images were captured at the same location. Common feature-based visual localization methods are FABMAP (Cummins and Newman, 2009) and CAT-SLAM (Maddern et al., 2012). A more detailed introduction into feature-based methods is given in section 1.2.2.

In chapter 4, we suggest a localization method which uses the amplitude spectrum of skyline images as scene descriptors. Therefore, our method is a signature-based method. Note that not all localization methods fit into one of these classes. For example, McManus et al. (2014) shows that machine learning can be used to identify image patches of interest and create for each image patch a signature which is robust against illumination changes and perspective changes caused by camera movement and rotation.

1.2.2 Visual Homing

In section 1.2.1 we discussed how a robot can determine its location using a set of previously captured snapshots. Now we assume that the robot tries to reach a previously visited goal location. To reach the goal location, the robot has to determine the pose of the current view relative to the snapshot[2]. Often we are only interested in the direction towards the location where the snapshot was captured, called the **home vector**. As soon as the home vector is known, the robot is able to drive towards the goal location. It is clear that for successful homing both the current view and the snapshot have to contain overlapping information of the environment. Using camera images as scene descriptors, the current view and snapshot have to contain landmarks (e.g. doors, cars, trees, etc.) visible in both images. With an increasing distance between the current view and snapshot, the amount of overlapping landmark information decreases. As a consequence, the difficulty to determine the home vector increases.

Homing methods suffer from similar problems as localization methods: Illumination changes or movement of obstacles reduce the overlapping information between the current view and the snapshot. Moreover, ambiguities lead to mismatches between visual features, for example a robot moving along a hallway might repeatedly observe a similar visual scene. This periodicity can hardly be prevented using a camera only.

Homing methods do commonly only return an estimate for the relative pose between the current view and snapshot, but do not consider additional information such as reachability (e.g. obstacles, stairs). Obstacle avoidance, path planning, and other complex navigational tasks are normally solved on a higher level (González et al., 2016). In this work, we only consider the most simple task of path planning, namely route following (section 1.2.3), which can be achieved by repeating homing for multiple snapshots along a previously driven route.

A wide range of alternative methods have been suggested which can be used for homing. For example, methods based on geometries (e.g. vanishing points (Bazin et al., 2008, Lee and Yoon, 2015)) or optical flow (e.g. Horn-Schunk (Bruhn et al., 2005) or Lucas-Kanade (Tamgade and Bora, 2009)). In the following, we focus on holistic and feature-based methods.

1.2.2.1 Holistic Methods

As stated by Möller et al. (2014), holistic methods for visual homing can be separated into the following categories: Warping methods, DID methods (descent in image distances), parameter methods, multi-snapshot methods. In this work, we focus on warping methods (chapter 6); a detailed introduction and literature overview of holistic methods can be found in Hillen (2013), chapters 3 and 5.

Holistic methods use the complete current view and snapshot to perform a pixel-wise comparison. As a consequence, small changes in the image as caused by pedestrians or moving shadows often have negligible influence on the image comparison. To decrease the influence of illumination changes, which affect large parts of the image, warping methods often use preprocessing techniques (e.g. edge filters) or appropriate distance measures (Möller, 2016a). A disadvantage of holistic methods is the necessity of rotationally aligned images. Commonly, holistic methods circumvent this problem by requiring rotationally aligned images. This can for example be achieved using additional sensors (e.g. an inertial measurement unit) or methods to estimate the rotational offset from the current view and snapshot directly using a visual compass (section 1.2.4).

Warping methods extend the idea of the visual compass by performing a search for both the rotational and translational offset between the current view and snapshot. This is done by internally simulating a movement of the robot by appropriately distorting the current view. By searching for the movement hypothesis which minimizes the pixel-wise image distance between the

[2] Note that the relative pose between two camera images has 5 degrees of freedom (**DoF**) in 3D-space: The orientation (3 DoF) and the direction towards the snapshots pose (2 DoF). Since no depth information (distance to objects) is available, the distance to the snapshot pose cannot be determined from using single images only. Additional sensory information, e.g. LIDAR or stereo cameras, can be used to provide the necessary depth information.

distorted current view and the snapshot, the rotational and translational offsets can be determined. Warping was introduced by Franz et al. (1998) for robots limited to planar movement. Their approach uses panoramic images with a single row (one-dimensional images), e.g. obtained by averaging a panoramic image column-wise. The methods 2D- and min-warping suggested by Möller et al. (2010) enhance this idea by using panoramic images (two-dimensional images). The generalization of warping for non-planar movements is difficult since the degrees of freedom are increased. We suggest a generalization of warping for non-planar movement and provide a more detailed introduction to warping methods in chapter 6.

DID methods (descent in image distances) use that the distance measure between two rotationally aligned images increases smoothly with the spatial distance. By computing or estimating a gradient for the image distance measure, an agent can be led towards a goal location (Möller and Vardy, 2006). Parameter methods reduce images to a sparse representation via a parameter vector[3]. As for DID methods, the agent is led towards the goal location by performing a gradient descent. The idea of multi-snapshot methods is to collect multiple snapshots at locations in the vicinity of a goal location and store a home vector for each snapshot. By searching for the snapshot which minimizes the image distance measure with the current view, the agent can move in the direction of the goal location by heading along the stored home vector (Graham et al., 2010).

1.2.2.2 Feature-Based Methods

In contrast to holistic methods, feature-based methods do not use complete images for comparison, but search each image for a set of salient pixels called **features**. Feature-based methods work in two stages: First, a **feature detector** searches for **key-points** (i.e. salient pixels) in the image. Second, a **feature descriptor** creates an information vector for each key-point. Note that there is no explicit definition of key-points, each feature detector can choose different pixels as key-points and each feature descriptor can describe them differently. The aim of feature descriptors is manifold: The information vector should be small to reduce the memory usage and at the same time contain sufficient information to uniquely describe each feature. Depending on the task, the descriptor should be invariant against rotation, translation, and zooming of the camera and illumination changes. Since both the feature detector and descriptor are — at least theoretically — independent of each other, arbitrary feature detectors and descriptors can be combined, however not all combinations are meaningful. An overview of common feature detectors and descriptors can be found in table 1.1. Instead of using key-points, an alternative approach is to detect and describe *image patches* which aims to make features better distinguishable from each other (McManus et al., 2014).

Assuming that a set of features has been extracted from both the current view and snapshot, similar features which appear in both images need to be matched. The most simple approach is by comparing all features in the current view against all features in the snapshot and search for the best matches (**brute-force matcher**). With an increasing number of features, this approach becomes computational expensive. A frequently used alternative to brute-force matchers is the approximate FLANN matcher (Fast Library for Approximate Nearest Neighbors, Muja and Lowe (2009)).

After a set of matches has been found between features of both images, the position of the features in the images can be used to deduct the rotational and translation (up to a scaling factor) offset between the current view and snapshot. The calculation requires at least five correctly matched features (Stewenius et al., 2006), alternative methods which require more features exist but are rarely used.

Often it is not sufficient to use the best five matches, since a single mismatch can ruin the pose estimation. In most applications, the pose estimation is repeated for random choices of five matches until a pose estimation is found which fits for the majority of the remaining matches. This technique is called random sample consensus (RANSAC, Fischler and Bolles (1981)). The

[3] For visual localization (section 1.2.1), we called a similar approach *signature-based methods*.

Name	Feature detector	Feature descriptor	Author
SIFT	✓	✓	Lowe (1999)
SURF	✓	✓	Bay et al. (2008)
ORB	✓	✓	Rublee et al. (2011)
BRISK	✓	✓	Leutenegger et al. (2011)
FAST	✓	✗	Rosten and Drummond (2006)
BRIEF	✗	✓	Calonder et al. (2010)
FREAK	✗	✓	Alahi et al. (2012)

Table 1.1: Overview of commonly used feature detectors and descriptors.

complete process of feature detection and description, matching, and pose estimation is visualized in example 1.2. A comparison between holistic methods and feature-based methods can be found in Fleer and Möller (2017).

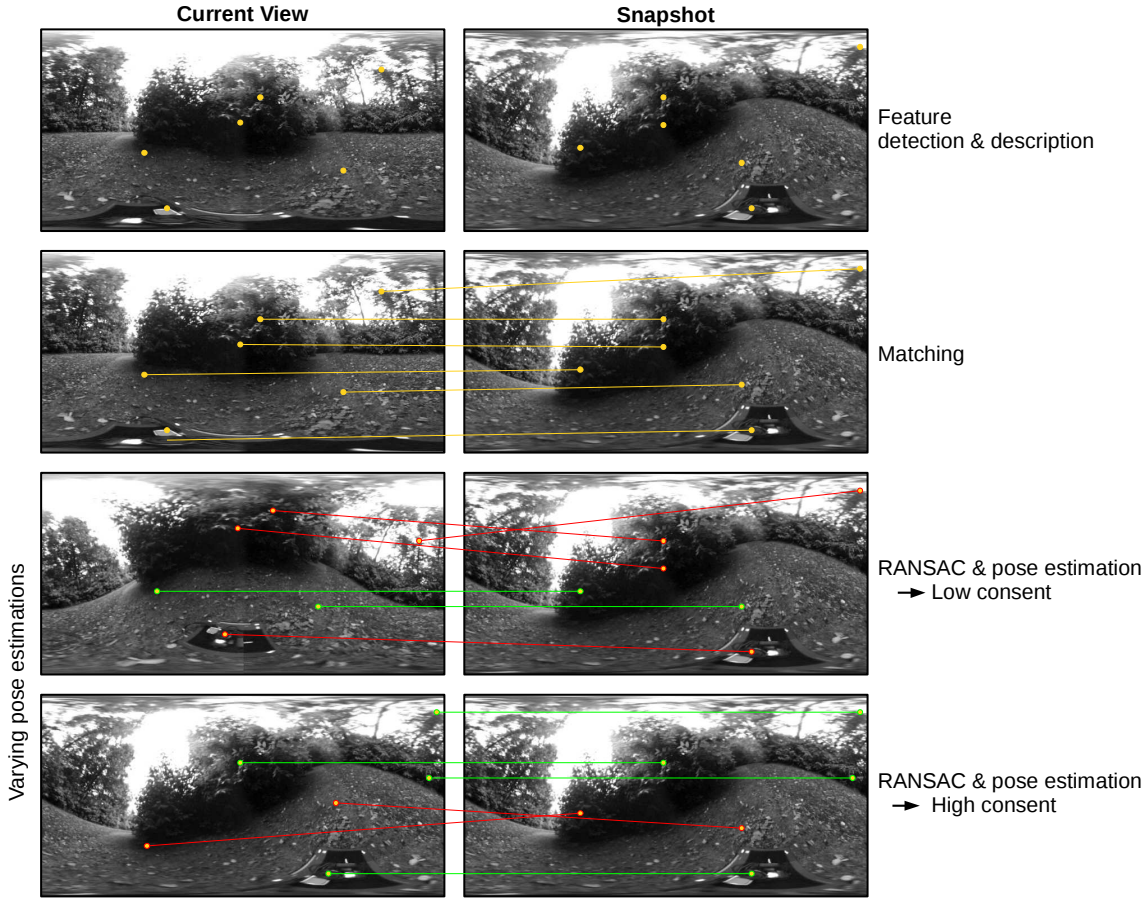
1.2.3 Route Following

A common task in robot navigation is to drive along a previously known route (Franz et al., 1998, Weber et al., 1999, Goedemé et al., 2005): In a **training run** the robot is driven manually and collects scene descriptors along the route. The scene descriptors can be collected after the robot traveled a specific distance or after some time has passed. The intervals do not have to be fixed, but can be chosen to optimize the data storage or information value (Denuelle and Srinivasan, 2016). For example, a new scene descriptor can be stored as soon as the difference to the last stored scene descriptor exceeds some threshold. In a **test run**, the robot is placed at the beginning of the route. Using a suitable localization method, the robot should now localize itself on the route by comparing its current view with the stored snapshots along the route (section 1.2.1). After the robot successfully localized itself, it can perform homing (section 1.2.2) to drive towards the next snapshot on the route. Assuming that both the localization of the robot and the calculation of the home vector were successful, the robot is able to follow the route by continuously repeating this process.

For localization and route following it is mandatory to have a representation of the robot's working area, i.e. a map. The most simple representation of a map is a linear graph, where each snapshot is only connected to its predecessor and, if available, successor. This representation is a special case of a topological map; in the following we briefly discuss common types of maps which can be used to perform route following.

- **Metrical maps**: Sensor information from each location is integrated into a map which preserves spatial information. The advantage of metric maps is that — once the robot is localized — the navigation of the robot can be significantly simplified by using odometry information. However, the creation of a metric map is problematic: Either external information of the map has to be available (e.g. a building plan) or the map needs to be constructed by the robot while driving (SLAM). For the latter, with an increasing size of the map small errors accumulate; in the worst case the map becomes inconsistent. This is mainly due to erroneous measurements of the sensors and an increasing uncertainty of the robot's location. Metrical maps are for example used by Engel et al. (2014), Davison et al. (2007), Weiß et al. (1994).
- **Topological maps**: A topological map is a graph, i.e. a set of vertices and edges. Each vertex represents a location of the robot, while an edge between two vertices indicates if the robot is able to directly move between these locations. Topological maps do not contain any metric information: For example, two locations which are spatially close to each other can be separated by many edges in the graph. The main advantage of topological maps is that they

Example 1.2: Feature-Based Methods



Feature-based methods for pose estimation consist of multiple stages. Initially, interesting pixels in the image are identified and for each an information vector is created using a feature detector and descriptor (table 1.1). Afterwards the features are matched by searching for similar information vectors in the current view and the snapshot. Since false matches could be found, it is commonly not recommended to only use the best matches for pose estimation. Instead, pose estimation is repeatedly performed for a random set of matches (RANSAC). The remaining matches, which are not used for pose estimation, are then used for evaluation.

can be simply built by adding vertices to the graph, where each new vertex is connected by an edge with its predecessor. A previously visited location can then be visited by following the graph's edges to the corresponding vertex. While topological maps are theoretically sufficient to return to any location in the graph, the graph does not allow reasoning over the spatial working area, e.g. shortcuts cannot be derived. Another problem with topological maps is that the robot should recognize if a location was already visited and therefore has been previously added as a vertex to the graph. In this case a **loop-closure** (Arroyo et al. (2014), Sünderhauf and Protzel (2011), Cummins and Newman (2009)) can be performed to merge the edges in the graph. Note that a wrong loop-closure (e.g. merging two identically looking images of different locations) or a missed loop-closure (e.g. the lighting conditions changed between both visits) can significantly worsen the maps quality. Topological maps are for example used by Gerstmayr-Hillen et al. (2013).

- **Topometrical maps**: By combining metrical and topological maps, some disadvantages

of both can be overcome. As for metric maps, topometrical maps have a global coordinate system but simultaneously — as for topological maps — a graph with vertices for all visited locations and edges between neighboring locations. Again, the metric map will be corrupted by erroneous sensor measurements, however the graph structure allows to reconstruct the local area around the robot by using information of the metric map which is related to the directly neighboring locations only. Moreover, tasks as path planning can be performed on the graph instead of the (eventually corrupted) metric map. As for topological maps, the construction of a topometrical map has to recognize loop-closure to keep the graph consistent. Topometrical maps are for example used by Dayoub et al. (2013) and Möller et al. (2013).

1.2.4 Visual Compass

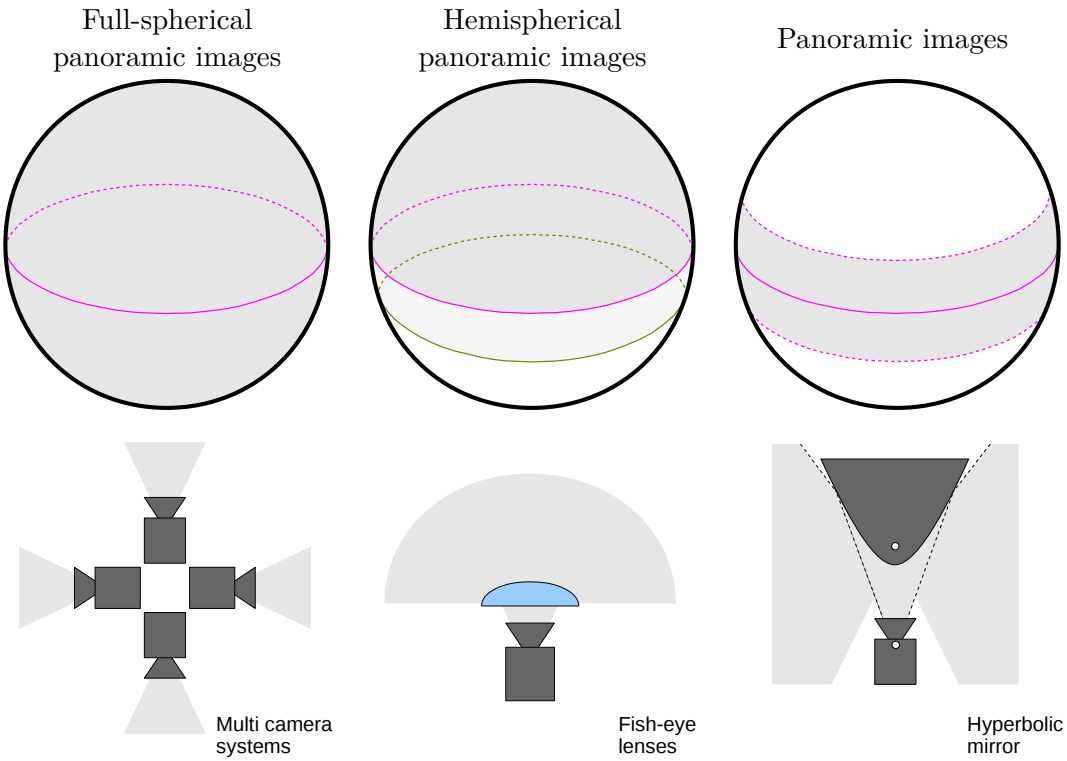
The original visual compass (Zeil et al., 2003) is a holistic method to rotationally align panoramic images and is often used as a preliminary stage for ongoing methods as localization or loop-closure. It uses the observation that with an increasing rotational offset, the pixel-wise image distance between two images increases smoothly (Zeil, 2012). By performing an exhaustive search over possible camera orientations, the rotational offset can be determined. As soon as the rotational offset is known, the current view and snapshot can be aligned. Commonly, the visual compass is used to determine the azimuthal orientation for robots moving on a plane. A disadvantage of the visual compass is that it can only determine the rotational offset between the current view and snapshot. Image differences caused by a translational offset of the robot cannot be estimated and, even worse, lead to an erroneous rotation estimate. Therefore the visual compass should preferably only be used for rotation estimations of current views and snapshots which were captured at roughly the same location. As shown by Stürzl and Mallot (2006), the visual compass can be implemented efficiently in the frequency domain. In chapter 5, we give a more extensive introduction to the visual compass and show how the visual compass can be enhanced to work for rotations around arbitrary axes.

1.2.5 Biologically Inspired Visual Navigation

The remarkable ability of social insects, e.g. ants, wasps, or bees, to navigate between their nest and foraging sites has been the subject of many navigation studies. As discussed in more detail in chapter 2, visual navigation plays a decisive role in insect navigation, and since insects possess only tiny brains, they must have developed parsimonious algorithms to process the visual information. Therefore visual navigation of insects is an important source of inspiration for visual navigation methods focused on low memory usage and computational costs. Especially holistic or signature-based methods, which are commonly rather simple in contrast to feature-based methods, are often inspired by the study of insect navigation or used to model insect behavior (reviews: (Denuelle and Srinivasan, 2016, Zeil, 2012, Franz and Mallot, 2000)); this includes the visual compass (Zeil et al., 2003), warping (Möller, 2012), the average landmark vector model (Lambrinos et al., 2000, Goldhoorn et al., 2007), and methods based on flow fields (Strübbe et al., 2015) or scene similarity (Baddeley et al., 2012).

Another important source for visual navigation strategies is the study of of rodent brains (O'Keefe, 1976). It showed that specialized *place cells* exist which become active as soon as the animal enters a specific location. It could be shown that simple navigational tasks can be performed ,y mimicking the functionality of place cells using artificial neural networks (Arleo and Gerstner, 2000, Arleo, 2000). Moreover, by implementing these neural networks in a more comprehensive framework, Milford and Wyeth (2010) showed that long-term visual navigation can be performed using visual information only.

Example 1.3: Omnidirectional Cameras



The sketches show the field of view (top, gray areas) of different omnidirectional camera systems (bottom). Full-spherical panoramic images can be obtained using a camera sensor which consists of multiple cameras facing in different directions. By stitching the camera images, a full-spherical panoramic image of the complete scene can be obtained. Hemispherical images have an opening angle of at least 180° or more and can be captured using fish-eye lenses. If we state no prerequisites to the field of view, we simply use the term panoramic image. For example, panoramic images with a small field of view but high resolution can be captured using a hyperbolic or parabolic mirror in front of a camera equipped with a perspective or telecentric lens, respectively. Panoramic images captured using a hyperbolic or parabolic mirror cannot contain the poles (white dots) since they are either blocked by the camera or the mirror.

1.3 Omnidirectional Camera Sensors

Cameras are compact, lightweight, and low-cost sensors with moderate power consumption, making them a preferable choice for the use on robot platforms. Moreover, most robot platforms can use cameras for multiple other purposes as for example obstacle detection, which reduces the overall number of sensors. Commonly, cameras are equipped with perspective lenses with an opening angle of around 45°. These lenses are able to capture the visual scene in front of the camera with a high resolution and are — since they do not suffer from distortions as for example fish-eye lenses — the preferred lens type to extract visual features (section 1.2.2.2). As an alternative to perspective lenses, omnidirectional camera sensors are used to obtain panoramic images. An overview of commonly used omnidirectional camera setups is shown in example 1.3. In comparison to perspective images, panoramic images have a larger opening angle, but smaller image resolution. Moreover, full-spherical panoramic images cannot be mapped onto an image plane without distortions; a possible mapping using spherical coordinates can be seen in example

3.4.

The construction of an omnidirectional camera sensor is a non-trivial problem, but with an increasing interest in panoramic imaging a rapid development in omnidirectional camera sensors arose over the last decades (Benosman and Kang (2001), section 2). The first known patent for such an omnidirectional camera sensor has been proposed by D.W. Rees in 1970 (patent number: US3505465 A). Rees proposed to use a hyperbolic mirror in front of a camera to obtain an omnidirectional camera sensor with a single point of view[4]. Three types of omnidirectional camera sensors frequently used to obtain panoramic images are in the following briefly discussed:

- **full-spherical panoramic images**: These images gather the complete field of view at the camera's location. Full-spherical panoramic images can be created by mapping the images of multiple cameras — each facing in a different direction — together into a single image. In practice, the creation of full-spherical panoramic images is complicated: All cameras have to be calibrated in order to map their individual images into a single image. These calibrations are tedious and prone to errors with an increasing number of cameras. Stitching algorithms (Abraham and Simon, 2013) can be used to reduce the error at the transitions between neighboring images. Moreover, the camera — or at least the chassis at which the camera is mounted — is always visible, and reduces the effective field of view. Recently, cameras with the possibility to capture full-spherical panoramic images become accessible at the consumer market, therefore rapid progress in the development of full-spherical panoramic image sensors can be expected.
- **Hemispherical images**: A camera with an opening angle of at least 180° can be used to capture hemispherical images. These images have the advantage that they can be recorded using a single camera equipped with a fish-eye lens. The mapping of the image to spherical coordinates is — compared to the mapping of multiple cameras as needed for full-spherical panoramic images — simple and can be estimated using various toolboxes; in this work we use the OcamCalib toolbox (Scaramuzza et al., 2006). A disadvantage of fish-eye camera images is that the resolution and distortion of the captured image depends on the viewing angle; both decrease towards the rim of the fish-eye image.
- **Panoramic images**: The mass production of fish-eye lenses started in the 1960's (Stafford et al., 2004), however until fish-eye lenses reached a sufficient viewing angle, omnidirectional imaging was mainly achieved using **catadioptric camera systems**. These consist of a hyperbolic or parabolic mirror placed in front of a camera. By design, catadioptric camera systems cannot collect hemispherical or full-spherical panoramic images since both poles are either blocked by the camera or the mirror. Advantageously, the curvature of hyperbolic and parabolic mirrors have multiple degrees of freedom which can be used to optimize the resolution of the panoramic image for certain viewing angles. However, the mirror needs additional space and adds weight to the experimental setup, making it commonly unsuitable for small and lightweight robots such as micro air vehicles. The experimental setup we used to gather omnidirectional multispectral images in chapter 2 uses a hyperbolic mirror and was designed using our *hyperbolic mirror toolbox* (Differt, 2014).

In this work, we collected multiple databases with full-spherical panoramic images for systematic tests (appendix D). The full-spherical panoramic images allow us to create artificial images with arbitrary viewing angles and directions. These images are used to test our proposed visual 3D compass (chapter 5) and homing method (chapter 6). Due to the effort necessary to capture full-spherical panoramic images, it is often more convenient to work with hemispherical images in robot applications. Therefore we assume throughout this work that the visual input of a robot is

[4] In the best case, each world point should be focused by a camera lens into a single point of view (focal point). An omnidirectional camera sensor which fulfills this requirement can be modeled by the *effective pinhole* model (Benosman and Kang (2001), section 4.2).

always hemispherical with an opening angle of 180° or 220° as they can be captured by common fish-eye lenses available at the consumer market.

1.4 Outline

In this section we give a concise overview over the topics covered within this work. Each topic as well as our results are covered in detail in their corresponding chapters. A summary over all results can be found in chapter 7.

In chapter 2 we introduce the skyline — a binary image where each pixel is classified either as ground object or sky — as an illumination-invariant representation for outdoor navigation. The advantage of the skyline is its illumination invariance: Due to the strong impact of lighting and weather conditions on the visual appearance of a scene in outdoor environments, images recorded at the same location but different times may differ strongly. In contrast, the skyline is (at least theoretically) not affected by those disturbances. Inspired by the study of social insects such as bees, wasps, or ants, we use either ultraviolet-only (UV) images or a color contrast between UV and green light for classification. We aim to answer the two questions: 'Does skyline extraction benefit from the color contrast to extract the skyline using linear thresholding?' and 'Do either fixed (for all images) or variable (for each individual image) thresholds show best results?'.

For robots driving on bumpy terrain, a suitable representation of panoramic images is crucial. Intuitively, panoramic images can be projected onto a sphere to represent the visual scene around the robot, however it is not possible to represent a sphere as a two-dimensional image (e.g. example 1.2) without introducing distortions. In chapter 3 we give an introduction to real spherical harmonics (RSH), which form an orthonormal basis for real-valued functions defined on the unit sphere. Using the basis of RSH, panoramic images can be represented directly as functions on the sphere in the frequency domain. We show how the amplitude spectrum and bispectrum can be calculated to compare panoramic images and how the translational and rotational movement of a robot in the basis of RSH can be simulated. The theory derived in this chapter was implemented in a C++ library.

In the following three chapters we suggest methods for autonomous robot navigation in outdoor environments: In chapter 4 we suggest a method to localize the robot using the skyline which is extracted using a camera sensitive to UV-only. It is shown that the skyline is a reliable feature for outdoor localization and thus the representation of the skyline by its amplitude spectrum is partially tilt invariant. We examine the performance of our localization method on challenging tracks and compare it with competing state of the art localization methods. In chapter 5 we exploit properties of the RSH to generalize the visual compass as stated by Zeil et al. (2003) to arbitrary rotations. The visual 3D compass can be used to rotationally align panoramic images which were collected at roughly the same location but under different orientations of the robot. Various parameter settings are systematically tested to increase the performance of the visual 3D compass. Finally, the visual 3D compass is compared with feature-based methods. In chapter 6 we examine the possibility to perform visual homing for non-planar movement directly in the basis of RSH. We suggest 3D-warping as a generalization of 2D-warping (Möller et al., 2010) and show that it performs well on skyline-segmented images. By combining warping methods with the visual 3D compass, we examine the homing performance of 2D-, min-, and 3D-warping for non-planar movement.

In chapter 7 we give a summary, discussion, and outlook over all topics covered in this work.

CHAPTER 2

Multispectral Skyline Extraction

Evidence from behavioral experiments suggests that insects use panoramic views of their environments as cues for visual navigation. Especially the appearance of ground objects in the front of the sky influences the navigational behavior of insects. However, changes of lighting conditions — over hours, days, or possibly seasons — significantly affect the appearance of the sky and ground objects. One possible solution to this problem is to extract a binary skyline by an illumination-invariant classification of the environment into two classes, ground objects and sky. In this section we examine the idea of using two different color channels available for many insects (UV and green) to perform this classification. First, we collected skyline databases of suburban scenes, where the skyline is dominated by trees and artificial objects like houses as well as from mineral skylines (stones, sand, earth). On this databases, we show that a 'local' UV classification with adaptive thresholds applied to individual images leads to the most reliable classification. Furthermore, we show that a 'global' classification with fixed thresholds (trained on an image dataset recorded over several days) using UV-only information is only slightly worse compared to using both the UV and green channel. Second, we collected a wide variety of ground objects to examine their spectral characteristics under different lighting conditions. On the one hand, we found that the special case of diffusely illuminated minerals increases the difficulty to reliably separate ground objects from the sky. On the other hand, the spectral characteristics of this collection of ground objects covers well with the data collected in the skyline databases, increasing — due to the high variety of ground objects — the validity of our findings for novel environments. Third, we collected omnidirectional images, as often used for visual navigation tasks, of skylines using a UV-reflective hyperbolic mirror. We could show that 'local' separation techniques can be adapted to the use of panoramic images by splitting the image into segments and finding individual thresholds for each segment. Contrary, this is not possible for 'global' separation techniques. This chapter is a compilation of our publications Differt and Möller (2015) and Differt and Möller (2016).

2.1 Introduction

2.1.1 Navigational Abilities of Social Insects

The study of social insects — like bees, wasps or ants — in their natural environments and in artificial setups has accumulated extensive knowledge of their cognitive abilities. Especially the remarkable ability to navigate between important locations within their habitat, such as the nest or feeding sites, has received strong attention. Insects employ a repertoire of navigational strategies including following pheromone trails, path integration, and visual navigation (reviews: Cheng and Freas (2015), Graham (2010), Madl et al. (2015), Wolf (2011), Wystrach and Graham (2012), Zeil (2012)). In this work, we focus on visual navigation strategies of insects which have been studied in bees (Horridge, 2005, Collett and Kelber, 1988), wasps (Collett and Rees, 1997), and ants (Collett and Collett, 2002), specifically desert ants (Wehner et al., 1996) as these cannot use pheromone trails.

Since visual navigation requires the processing of a large amount of sensory information, it can

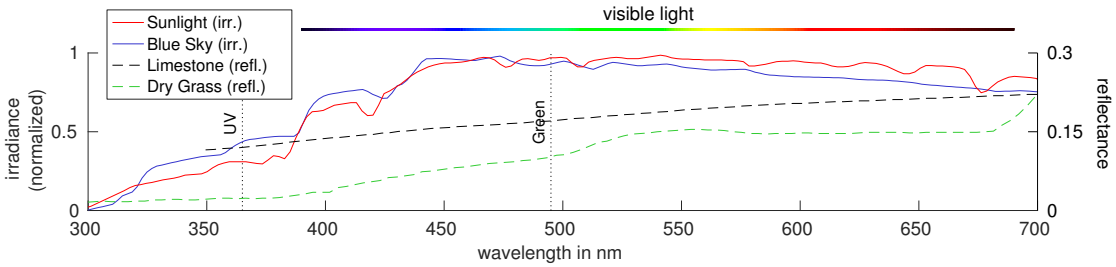


Figure 2.1: Irradiance spectrum (solid lines) of sunlight (red), clear blue sky (blue), and reflectance spectra (dashed lines) of limestone (black) and dry grass (green). The peaks of the glass filters used to obtain images in the UV and green channel are depicted (dotted lines). Note that the green channel is inspired by the insect *Cataglyphis bicolor* and does not match exactly with the perception of colors by humans (color bar). Figure created by author from data in Bird and Hulstrom (1983), Nann and Riordan (1991), Clark et al. (2007).

be suspected that clever mechanisms have evolved which are sufficiently parsimonious such that they can be handled by the tiny brains of insects. It is widely accepted that insects characterize places by retinotopic visual memories, called snapshots (Cartwright and Collett, 1983, Wehner et al., 1996, Collett and Zeil, 1997, Judd and Collett, 1998, Judd et al., 1999). We assume that this snapshot and the current images to which it is compared are preprocessed in a way that benefits the subsequent stages of the navigation mechanisms regarding both simplification and robustness.

2.1.2 Skyline as Landmark Cue

Insects are known to rely on salient landmarks cues as reference points for visual navigation (Cartwright and Collett, 1983, Durier et al., 2003, Wehner and Räber, 1979) and on navigational cues from the shape of the skyline of objects in front of the sky (Heusser and Wehner, 2002, Julle-Daniere et al., 2014, Graham and Cheng, 2009a). Similar results could be observed by Pratt et al. (2001), who showed under laboratory conditions that insects use prominent walls ('skyline') for navigation. Actually both salient landmarks and the skyline are simultaneously used during navigation, dependent on the current situation (e.g. 'Are landmarks cues still visible?', 'Does the visible skyline match the remembered skyline?') (Fukushi and Wehner, 2004, Wystrach et al., 2011). However, it could be shown that insects prefer distant landmarks which strongly contribute to the skyline, contrary to close and salient landmarks which do not contribute to the skyline (Collett and Kelber, 1988, Graham and Cheng, 2009b, Rosengren and Fortelius, 1986). Even more, changing constitutive parts of the skyline reduced the success of navigation significantly (Fukushi, 2001, Schwarz et al., 2014). Taken together we can claim that for insects the extraction of the skyline is an essential step in providing necessary navigational information.

By reducing the snapshot to a binary representation which classifies each pixel as either part of the ground or the sky, a simple representation of a scene (including the skyline information) could be obtained. However, such a classification is not trivial, since the illumination of the same scene captured under different lighting conditions (position of the sun, weather, time of day) strongly varies. An image captured at dawn is not only darker than an image of the same scene captured at noon, but also the spectrum of the sunlight (and thus also the light falling on objects) shifts from red to blue. Furthermore, changes in the sun position, moving shadows, or clouds change the appearance of a scene. Therefore it is important that the separation between the terrestrial and celestial parts of an image is invariant to illumination changes.

2.1.3 Perception of Light

Typically, the photoreceptors of an insect are sensitive to three different wavelengths, called L ('long'), M ('medium') and S ('short'). For hymenoptera these wavelengths are around $S = 340\,\text{nm}$ (UV), $M = 430\,\text{nm}$ (blue) and $L = 540\,\text{nm}$ (green). In rare cases, wavelengths around 600 nm

(red) or even higher (infrared) can be perceived (Briscoe and Chittka, 2001, Chittka, 1996, Menzel and Backhaus, 1991). Figure 2.1 shows irradiance spectra of direct sunlight and blue sky as well as reflectance spectra of limestone and dry grass. For objects on the ground, the reflectivity in the UV channel is commonly small (especially compared to visual light), while the UV portion of direct sun light and blue sky is high. Therefore, there is a high contrast between sunlight reflected from ground objects and the sky. As proposed by Wehner (1982), this allows a simple classification between ground objects and the sky (resulting in a binary skyline) in the UV channel by determining an appropriate threshold. This idea is supported by behavioral experiments with the ant *Melophorus bagoti*, where it could be shown that their navigational abilities are significantly decreased as soon as UV light is blocked (Schultheiss et al., 2016). The lower reflectivity of ground objects in the UV channel compared to the green channel, motivates an alternative approach: Instead of only using the UV channel for skyline extraction, a contrast of the UV and green channel can be used (Möller, 2002). This separation is, on the one hand, based on the reflectivity spectra of vegetation (Chittka et al., 1992, 1994, Gumbert et al., 1999), soil, or rocks (Sgavetti et al., 2006): A smaller amount of light with a short wavelength (e.g. UV) is reflected by terrestrial objects compared to light with a longer wavelength (e.g. green), such that reflected light has a small ratio of UV compared to green light intensity. On the other hand, light scattered by the sky (diffuse skylight) has a high ratio of UV light compared to green light due to Rayleigh scattering. Therefore, a color-opponent coding between two different colors, in this case some kind of 'UV-green ratio', may provide an illumination-invariant skyline separation. Kollmeier et al. (2007) explored different combinations of wavelengths for a skyline separation and found that the quality of separation increases with the distance between the two wavelengths. With respect to most insects, this would be the combination of the L and S channels. We chose UV/green (in the following UV/G) since it is the only combination available to the desert ant *Cataglyphis bicolor* (Mote and Wehner, 1980).

The spectral characteristics of ground objects under natural illumination as well as the difference in the spectra of light emitted directly from the sun and diffuse lighting from the sky has been examined by Möller (2002) and Kollmeier et al. (2007): Using a handheld device containing two photodiodes sensitive to UV and green light, databases have been collected which contain log UV/G data for a wide range of ground objects and sky patches. The data show that a simple linear separation of the logarithmized UV/G data (in the following denoted as log UV/G) is sufficient to correctly classify most of the collected data into two classes, ground objects and sky. Since the databases contain objects and sky patches recorded under different lighting and weather conditions this suggests that — using an appropriate log UV/G contrast mechanism — a threshold can be found to reliably extract the skyline.

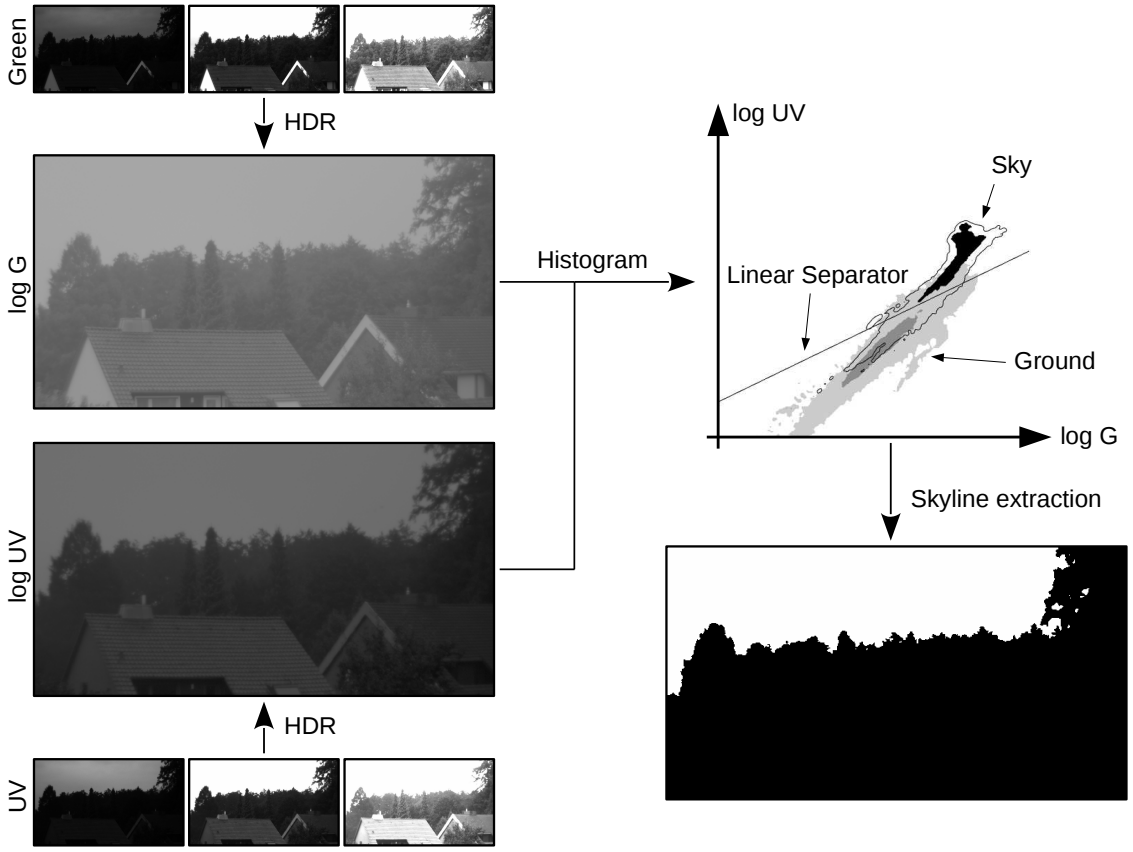
Insects are able to perceive several orders of magnitude of light intensity, which is mainly achieved by two different mechanisms: First, the insect eye is able to adapt its sensitivity to light over time. This adaption can be achieved over seconds up to hours, depending on the type of eye (Warrant, 2006, Greiner, 2005). Second, the photoreceptors do not respond in a linear way to a light stimulus, but it was found that the response is nearly logarithmic under bright light conditions (Laughlin, 1989, 1994). The second point is equivalent to the more general Weber-Fechner law which states that stimuli are perceived in a logarithmic way (Fechner, 1860, Goldstein, 2014). In this work, we collect the logarithmic absolute light intensities of different scenes over complete days. The effects of local adaption are discussed in section 2.4.3.

2.1.4 Global and Local Classification Methods

All methods tested in this work are based on the same idea (example 2.1 and figure 2.4): First, the HDR image data (UV and green) of all pixels is projected onto a one-dimensional plane (contrast mechanism). Second, we determine a threshold value to perform the classification between ground objects and the sky. There are mainly two approaches for the second step:

- **Local**: Unsupervised methods, where the threshold is learned for each HDR image pair (UV

Example 2.1: Skyline Extraction



Overview of the different processing steps: For each channel (UV and green) the raw input images with varying exposure times (left, small images) are combined into HDR images (left, large images). Afterwards a 2D histogram is built (right, top), showing the distribution of the log UV and log G values for each pixel of either a database of multiple image pairs ('global' methods) or of a single image pair ('local' methods). Finally, a linear separator is used to classify each pixel as either ground or sky, providing an image of the skyline (right, bottom) which is widely illumination-invariant.

and green) individually.

- **Global**: Supervised methods, where the threshold is learned offline on a given, hand-labeled dataset.

For local methods, a classification of ground objects and the sky can be obtained by calculating the histogram of the projected data and determining an appropriate threshold value. While the implementation of thresholding algorithms is simple, the determination of the threshold value may be a rather complex task for insects. Whether biologically plausible versions of local methods can be found is currently open.

Global methods use a fixed threshold value which is learned on a large set of training data. For global methods, the classification between ground objects and the sky reduces to the calculation of a scalar product and a simple comparison (section 2.2.8). Such a separation would be computationally cheap and could be accomplished by a simple neuronal network, making it a plausible model for skyline extraction in insects. Both local and global methods use a fixed projection angle which is learned (supervised) from hand labeled data. In a neural model, the projection (scalar product) could be encoded by fixed synaptic weights.

2.1.5 Related Work

Skyline extraction has been the subject of several studies in the past including applications for aircrafts (Dusha et al., 2007), marine vehicles (Gershikov et al., 2013), and cars (Neto et al., 2011). However, in these studies it was assumed that the distance to the objects which contribute to the shape of the skyline is large such that the skyline in an image can be approximated by a line. The suggested algorithms are therefore not suitable to detect arbitrary skyline shapes as they may contribute to the visual navigation of insects.

For complex skyline shapes, local methods have been successfully used to extract the skyline using only the UV channel only on mobile robots (Stone et al., 2014, 2016). However, in these studies skyline extraction was mainly performed in suburban environments or in forests. In this chapter we aim to examine the possible application of skyline extraction also in mineral-rich environments like deserts, rocky landscapes, or beaches.

As an alternative to the UV channel, it has been suggested to use the infrared channel (Bazin et al., 2009) or visual light (Shen and Wang, 2013) for skyline extraction. These methods optimize energy functions to find an optimal skyline, a rather complex approach which is computationally expensive (compared to simple thresholding) and not feasible as a model for insects.

Besides skyline extraction, several applications have been found for using UV imagery in outdoor robotic experiments: Based on the observation that many insects use UV and green receptors to detect polarization, the detection of polarization patterns in the sky has been tested in both channels (Carey and Stürzl, 2011). They also used that in UV images clouds are nearly invisible and that the contrast between ground objects and sky is increased to improve the distinction between ground objects and the sky. This observation is for example also used to estimate the attitude (Tehrani et al., 2012b) or to correct the drift of gyroscopes (Tehrani et al., 2012a) of unmanned aerial vehicles (UAV).

2.1.6 Contributions

In this work we perform three different experiments to evaluate the benefits of multispectral contrast mechanisms compared to single channel vision for skyline extraction in insects and robot applications.

First, we collected a total of ten multispectral (UV and green) datasets of different skylines: Three datasets contain skylines made of stones, sand, and earth (each over a complete day) in the UV and green channel. Additionally we collected seven datasets in a forest/suburban environment, each recorded on a different day. The amount of collected sky data therefore covers a total of ten days including dry and sunny days as well as rainy days. The mineral skyline databases greatly increase the variety of spectral characteristics for objects an insect or robot might encounter in outdoor environments. On these datasets we examine if skyline extraction can be performed using local and global separation techniques.

Second, we collected a wide variety of ground objects (e.g. undergrowth, shrubs, pavement, etc.) to increase the database of potential landmarks. Some of these ground objects were collected under three different lighting conditions (directly exposed to the sun, in the shadow during a sunny day, diffusely lighted during a cloudy day) to show the influence of lighting conditions on ground objects. We show that especially diffusely illuminated minerals — e.g. stones lying in shadows — increase the difficulty to classify ground objects and sky. Moreover, we show that the spectral characteristics of most objects which could be encountered by insects or robots do not differ noticeably from the data collected in the skyline datasets, extending the validity of our results to a wider range of environments.

Third, omnidirectional images have been collected to show the influence of the varying spectral irradiance over the sky. As we will see, for global and local methods, the classification quality is strongly decreased for omnidirectional images. While local methods can easily be adapted by splitting the image into multiple segments and using an individual threshold for each, the adaption of global methods is not possible without violating the idea of a simple classification as it could

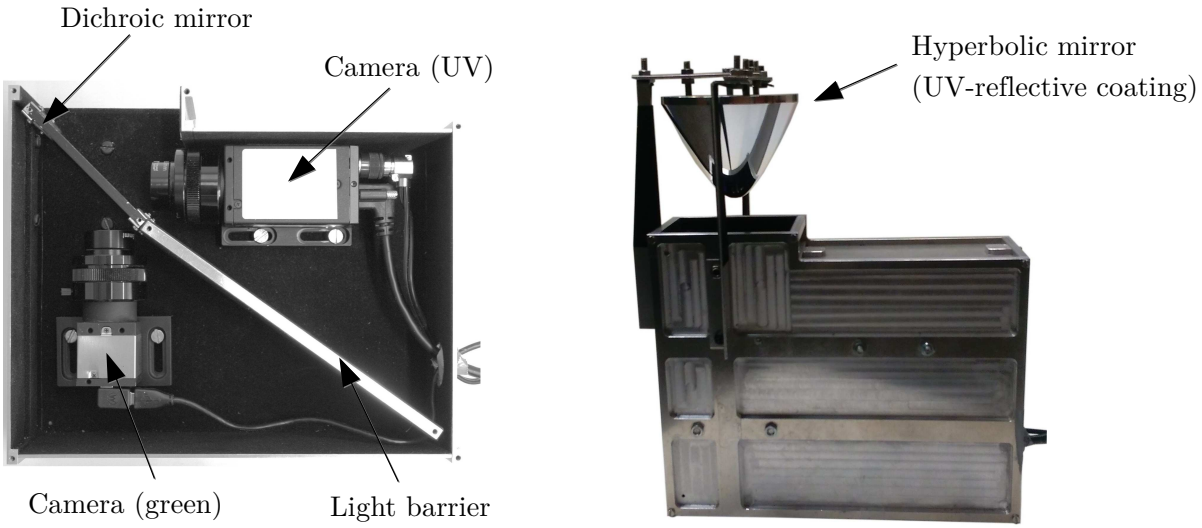


Figure 2.2: Left: The experimental setup (cover-plate removed) with two cameras recording the UV- and green-channel simultaneously. Right: To capture panoramic images, a hyperbolic mirror can be mounted. The mirror has a reflective (aluminum) and a protective layer (magnesium fluoride) to provide a nearly constant reflection over a wide range of wavelengths (including UV and visible light).

Channel	Name	Thickness	Peak	Bandwidth
UV	UG11 BG40	3 mm 1 mm	350 nm	50 nm
G	BG7 GG475	2 mm 2 mm	500 nm	70 nm

Table 2.1: Glass filter combinations for the UV and green channel. All filters are manufactured by *Schott*.

be used by insects. Calculating thresholds for multiple segments also reduces the effect of lens flare to the process of skyline extraction, which may lead to errors during the process of skyline extraction (Stone et al., 2016). Moreover, for omnidirectional images (e.g. using a fish eye lens), the skyline can be extracted completely and allows the use of existing navigational algorithms for panoramic images (Basten and Mallot, 2010, Friedrich et al., 2008, Möller et al., 2010).

Summarized, this study aims to answer two question: On the one hand, we want to answer the question if insects could benefit from multispectral vision as suggested by Möller (2002) (here: UV/G) to extract the skyline as an illumination-invariant landmark using parsimonious algorithms; and on the other hand, if local methods yield robust and computationally cheap (omnidirectional) skyline extraction usable for robot navigation.

2.2 Materials and Methods

2.2.1 Experimental Setup

For the simultaneous recording of a scene in two different wavelength bands we designed and built an optical system (figure 2.2, (a)). It contains two cameras, equipped with different filters, which observe the same scene via a dichroic mirror.

A monochrome camera (*UI-2220SE-M* by *IDS*) is used to record the green light range of a scene, while a specialized UV camera (*CM-140GE-UV* by *JAI*) is used for the UV range. Both cameras are equipped with the same UV-graded lens (*UV0928CM* by *UNIVERSE*) with a focal length of 9 mm, an f-number of $F/2.8$, and a full opening angle of 38°. To restrict the spectral range of both cameras, additional filters (table 2.1) have been mounted in front of each lens such

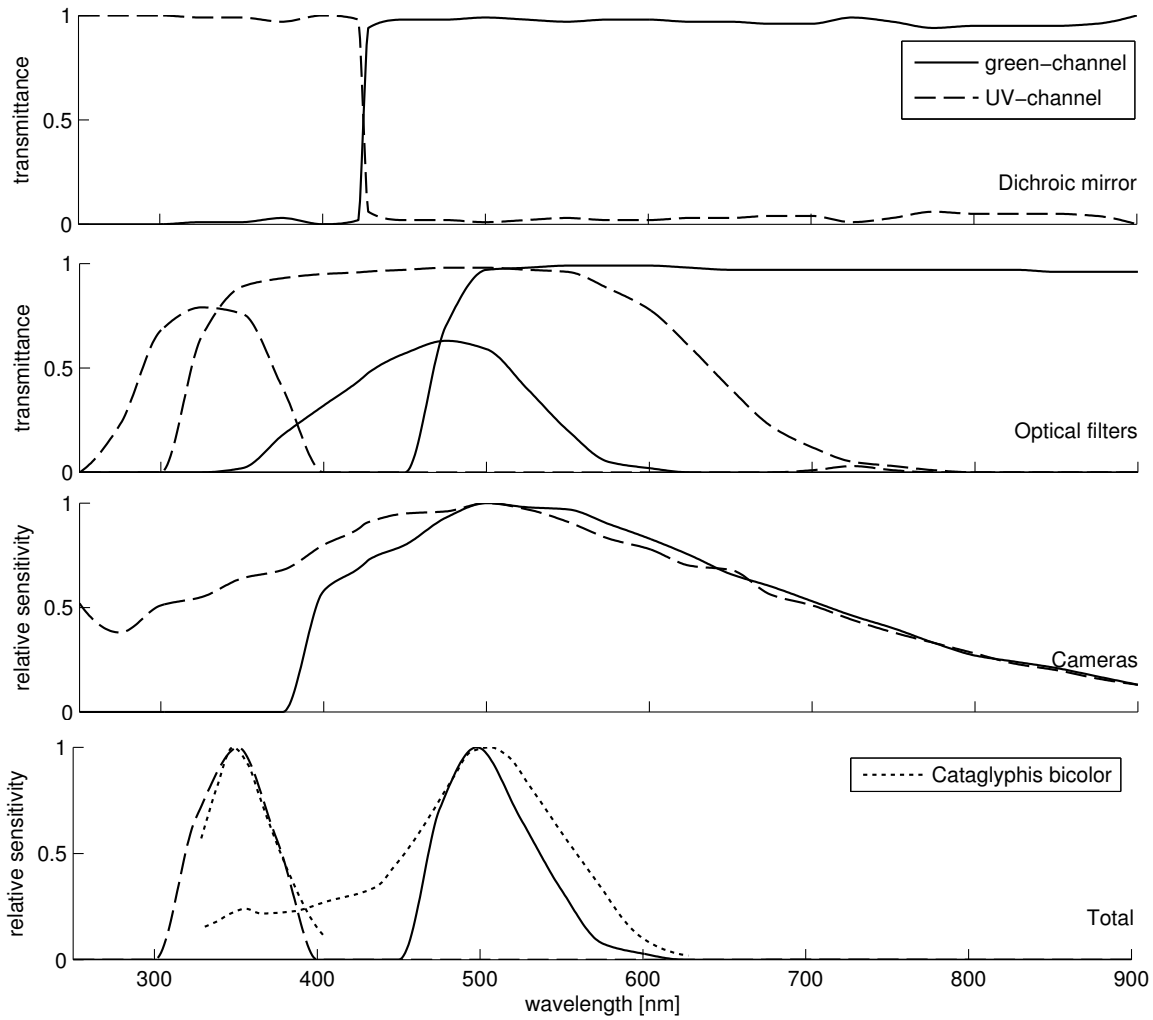


Figure 2.3: The spectra of the different optical parts installed in the experimental setup. The dashed lines correspond to the UV channel, the continuous lines to the green channel. From top to bottom: The transmittance of the dichroic mirror, the transmittance of the mounted optical filters, and the relative camera sensitivities. The bottom plot shows the normalized final relative sensitivities resulting from combining all optical components. For comparison, the spectral sensitivity of UV and green photoreceptors of the desert ant *Cataglyphis bicolor* (Mote and Wehner, 1980) are plotted as dotted lines.

that the sensitivity spectra of the visual channels have peaks at 350 nm and 500 nm, respectively (figure 2.3), which corresponds to the spectral sensitivity of desert ants (e.g. *Cataglyphis bicolor* (Mote and Wehner, 1980)). The dichroic mirror (*T425lpxr* by *Chroma*) mounted at an angle of 45° in front of both cameras is used to redirect the incoming light, which is split at a wavelength of 425 nm,to the two cameras. Due to the opening angle of the cameras objective, incident light may hit the dichroic mirror at angles between 31° to 61°, resulting in a shift of the dichroic mirrors spectrum of up to 20 nm in both directions. Also polarized light shifts the spectrum by up to 5 nm, resulting in a total maximal shift of 25 nm of the spectrum (plots for both cases provided by *Chroma*). However this has no effect to the total relative sensitivity since the split wavelength of 425(25) nm occurs between the bands of the mounted optical filters (compare figure 2.3 and table 2.1). In contrast to a 50/50 beam-splitter, the dichroic mirror does not halve the intensity of the incoming light. To avoid diffuse lighting, the inside of the setup is lined with black velour (*d-c-fix* by *Hornschuch*) and a light barrier was inserted to optically separate both cameras.

The experimental setup can be enhanced by adding a hyperbolic mirror to capture panoramic images (figure 2.2) with a maximal opening angle of around −20° to 45° relative to the horizon.

Database	Date	Temperature in C°	Condensation in l/m^2	Sunshine duration in h
Forest / Suburban	31 Aug. 2014	8.8 - 19.4	0.2	3.5
	01 Sept. 2014	7.9 - 19.2	0.8	0.6
	02 Sept. 2014	8.8 - 18.2	0.1	0
	03 Sept. 2014	9.9 - 21.4	0	7
	04 Sept. 2014	8.9 - 24.6	0	10.6
	05 Sept. 2014	10.6 - 26.3	0	5
	06 Sept. 2014	13.5 - 26.5	11	6.9
Stones	23 Aug. 2015	8.6 - 25.4	0	11.6
Sand	26 Aug. 2015	15.3 - 27.1	0	11.6
Earth	29 Aug. 2015	8.5 - 24.2	0	10.1

Table 2.2: Overview of the weather conditions during the recording days of the skyline database (Schmidt, 2014, DWD, 2014).

The mirror has a reflective (aluminum) and a protective layer (magnesium fluoride) to provide a nearly constant reflection over a wide range of wavelengths (including UV and visible light). The construction of the hyperbolic mirror is described in detail in our technical report (Differt, 2014).

2.2.2 Calibration

Since the images of the two cameras may slightly be offset, a custom image registration is performed: Around 150 points were manually selected in different pairs of UV-green images to optimize an affine mapping between both cameras. The optimization process reduces the average mapping error between the manually selected points in both images to approximately one pixel. The residual error depends on the distance to the observed objects in the scene and might be increased for objects close to the camera. To cope with the different resolutions of the cameras (768×576 (green) and 1380×1040 (UV) pixel), the UV image was mapped to the green image. Furthermore, both images are cropped such that the final images have a resolution of 550×300 pixel.

2.2.3 Data Collection

We collected three types of databases: **Skyline databases** containing images of both ground objects as well as sky, **panoramic skyline databases** which are similar to skyline databases but contain panoramic images, and **object databases** containing a wide variety of ground objects under differing lighting conditions. In the following we describe in detail how these databases were acquired:

For the skyline databases we collected data over seven subsequent days from 31 Aug. 2014 to 06. Sept. 2014 in a suburban area of Bielefeld which allowed shots of both natural (trees, shrubs) and artificial (house walls, roofs) objects. On each day, the camera system was set up at midnight in different positions and orientations (example 2.2 (a)). During these seven days, images under many different weather conditions were captured (table 2.2). The first three days were cloudy and wet with the sun barely visible, while the next three days were sunny and clear. The last day started with a storm until it brightened up around noon. To obtain a direct comparison between cloudy and sunny days, the camera setup was placed at the same spot on 02. Sept. 2014 and 03. Sept. 2014. Furthermore, we collected data of mineral skylines over three days between 23. Aug. 2015 and 29. Aug. 2015 (example 2.2 (b)). We created three different skylines in front of the camera, formed by (dry) stones, sand, and earth. We were not able to add a cover for our non-waterproof camera setup as this would have influenced the lighting conditions of the skyline. We therefore could not record images on days with rain or high humidity and had sunshine durations on the recording days between 9 to 12 hours (table 2.2). To avoid direct incident sunlight onto

the camera, which could damage the sensors or lead to over-exposed images, the camera setup was pointing north in all recordings of skyline databases. We nevertheless obtain a wide range of azimuthal angles between the camera and the sun from around 45° to 180°. An image series for both channels over a wide range of exposure times (section 2.2.4) was captured in 5 minute steps.

For the object databases we collected a total of 61 records with a wide variety of ground objects (without sky). Some of these objects (patches of grass, trees, stones, gravel, earth, and sand) have been recorded under three different lighting conditions: Objects (a) lying in the shadow on a sunny day, (b) directly exposed to sunlight on a sunny day, and (c) exposed to diffuse sky light on a cloudy day. The remaining objects recorded (without specific lighting conditions) contain foliage, fir sprigs, shrubs, pavement, undergrowth, and straw to ensure a wide range of collected data (example 2.3).

For the panoramic skyline databases we collected a total of 12 panoramic images on 12. May 2016 and 27. June 2016 during sunny weather to examine the influence of direction-dependent lighting (example 2.2 (c)). The panoramic images were recorded using a hyperbolic mirror mounted in front of the experimental setup (section 2.2.1). For each panoramic image we manually created a mask to mask out areas from the panoramic images which are corrupted by direct incident sunlight onto the camera.

2.2.4 HDR Imaging

We apply **High Dynamic Range** (HDR) imaging techniques to emulate the ability of insects to deal with a wide range of lighting conditions. HDR techniques can be used to superimpose a set of images of the same scene, but collected with different exposure times, to an irradiance image of this scene. An overview of different HDR algorithms is given by Bloch (2008). We based our HDR algorithm on the one introduced by Debevec and Malik (1998) which, despite its simplicity, achieves a performance comparable to the most recent but more complicated algorithms. For a detailed comparison between different HDR algorithms see Aguerrebere et al. (2014). The enhancements of our implementation over the method by Debevec and Malik (1998) are explained in detail in appendix B.1.

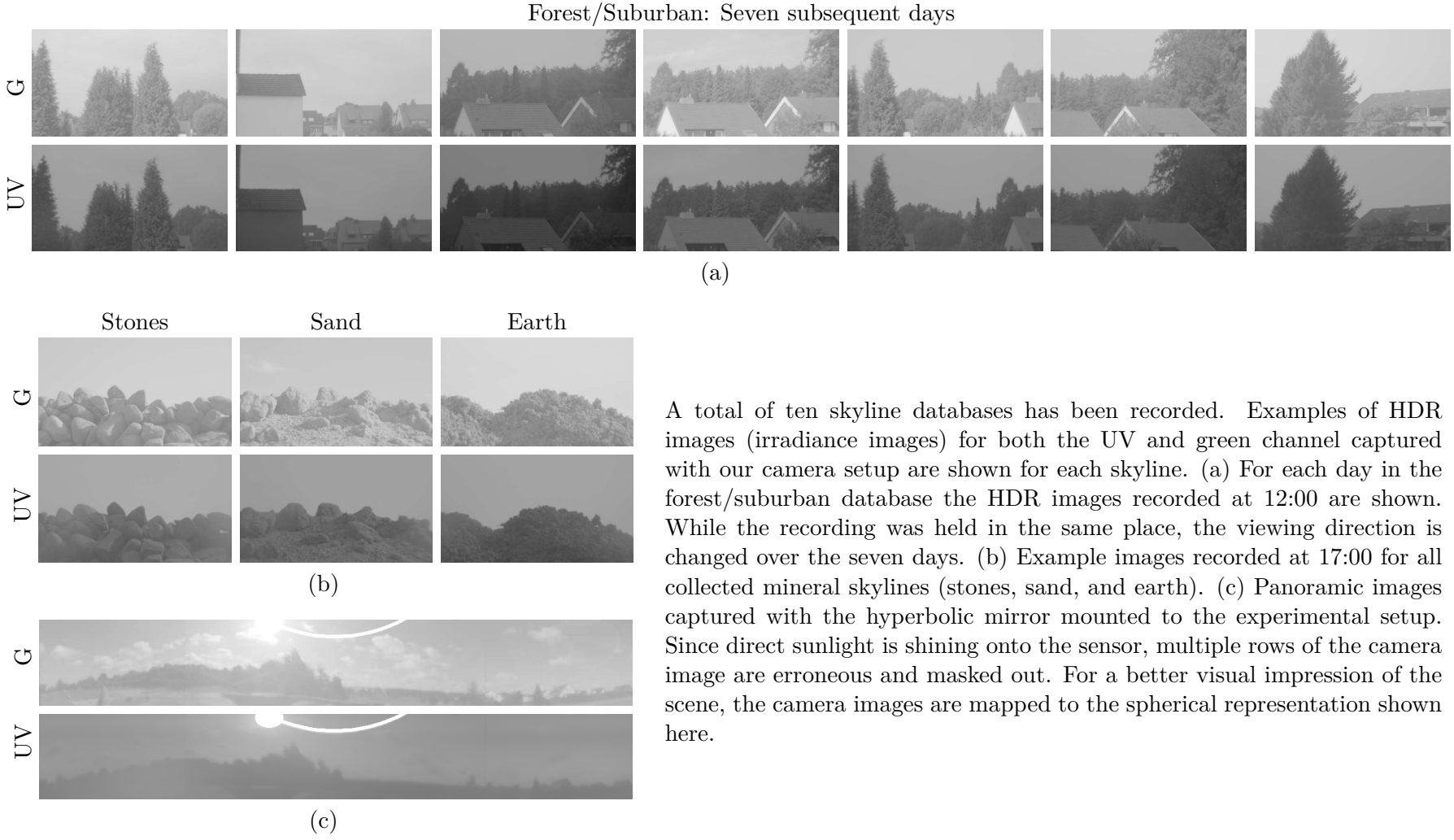
To achieve high accuracy in the determination of the irradiance it is crucial to provide a sufficient number of superimposable images with different exposure times to the algorithm. In order to deal with both bright and dark irradiance conditions, we captured a total of 11 images with exposure times of 2^k ms with $k = -3, \ldots, 7$ and calculated the calculated the *logarithmic* irradiance $\log E$. The logarithmic scaling is inspired by the response of photoreceptors in the eye to increasing irradiance values as observed for insects (Laughlin, 1989, 1994) or humans (Fechner, 1860, Goldstein, 2014). Alternatively to a logarithmic scaling, the linear camera response could be used to increase the color contrast in images (Garcia et al., 2013, 2014). However, a logarithmic representation has two advantages over linear camera response: First, a large range of irradiance values (here: 14 camera stops) can be represented visually. Second, a log UV/G contrast — robust to global intensity changes (Möller, 2002) — can be implemented as a linear separator (section 2.2.8) . We mapped the observed values $\log E \in [-0.7, 9.5]$ linearly to 256 discrete values in order to allow direct plotting of the scene irradiance. For the whole database, the collected irradiance values of the UV and green channel cover a range of 14 stops[1].

2.2.5 Creation of Data Samples

For the skyline databases we recorded a total of $4.7 \cdot 10^8$ UV-green log irradiance data pairs (one for each pixel), resulting in approximately 900 MB raw data (excluding time/place information). To work with this amount of data we decided to simulate a measurement (this could be single days or special times of day) as **samples** containing data pairs equally drawn from both sky and ground class (each data pair is labeled to either sky or ground class by hand-drawn masks, section 2.2.7). Since for each data pair the capture time and date is known, samples can be

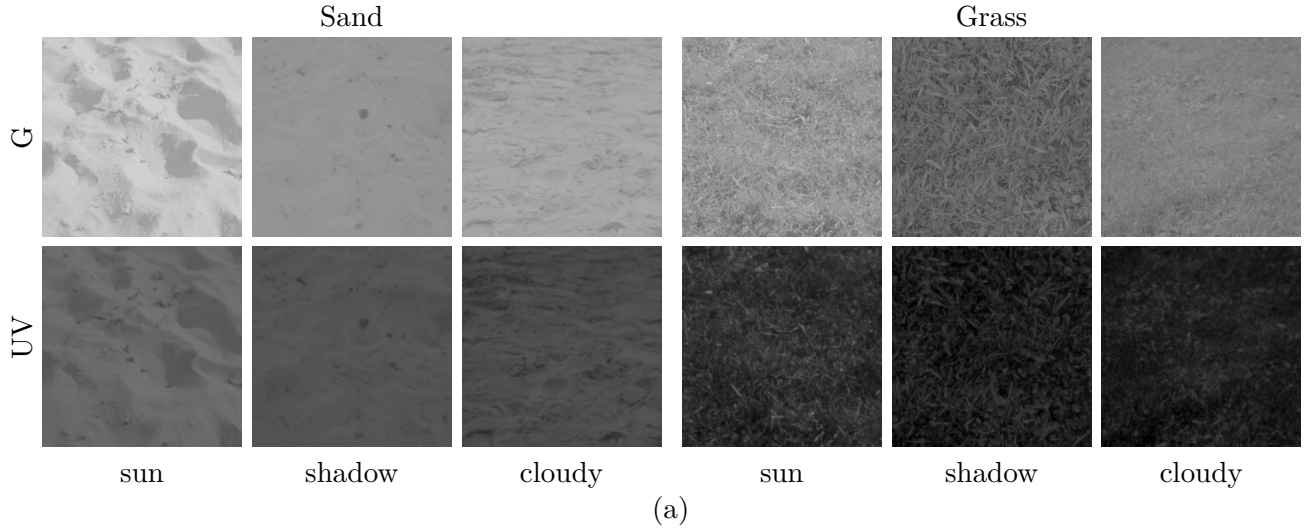
[1] In terms of photography one stop equals doubling (or halving) the amount of incident light.

Example 2.2: Skyline Databases

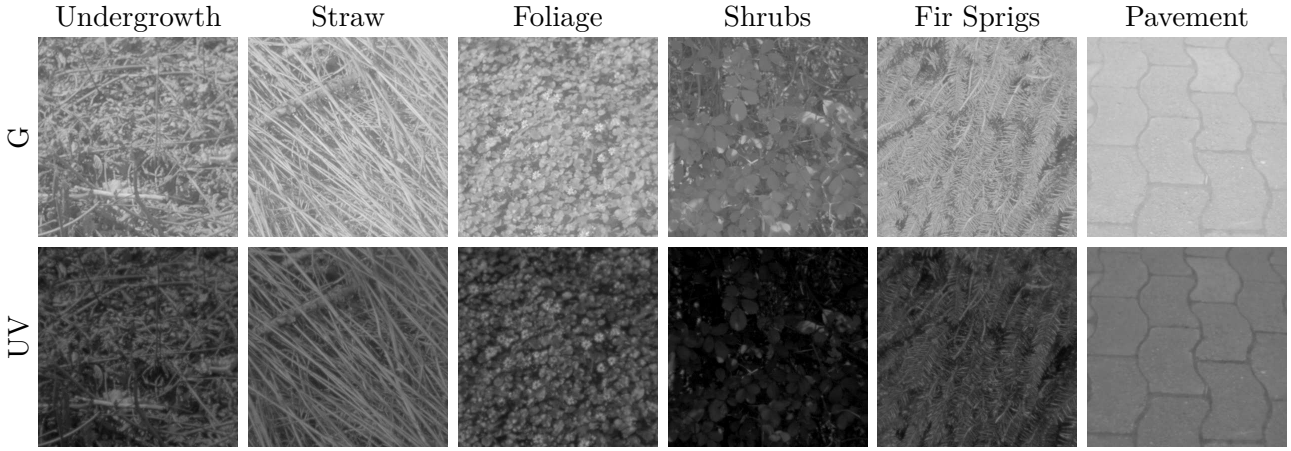


A total of ten skyline databases has been recorded. Examples of HDR images (irradiance images) for both the UV and green channel captured with our camera setup are shown for each skyline. (a) For each day in the forest/suburban database the HDR images recorded at 12:00 are shown. While the recording was held in the same place, the viewing direction is changed over the seven days. (b) Example images recorded at 17:00 for all collected mineral skylines (stones, sand, and earth). (c) Panoramic images captured with the hyperbolic mirror mounted to the experimental setup. Since direct sunlight is shining onto the sensor, multiple rows of the camera image are erroneous and masked out. For a better visual impression of the scene, the camera images are mapped to the spherical representation shown here.

Example 2.3: Object Database



(a)



(b)

(a) Single HDR images of sand and grass captured with our camera setup for both the UV and green channel. A total of three different images has been taken for each object: Two images were taken on a sunny day, one time with the object exposed to the sun (*sun*) and one time lying in the shadows (*shadow*). A third one was collected during a cloudy day (*cloudy*). (b) Examples of further collected data in the object database, which were not explicitly collected under specific lighting conditions.

obtained by simulating a probability experiment where the underlying probability distribution depends on the time and, if necessary, the date of interest. If not mentioned otherwise, the probability distribution is an uniform distribution. We denote a sample as $X_{\text{time of day}}$, e.g. X_{8-19} for the sample representing the collected data of all skyline databases between $8:00-20:00$. Note that for samples of all databases together (e.g. figure 2.5), we draw from the mineral skyline databases seven-times more samples than from the forest/suburban database (the forest/suburban database contains data over seven days). We use the forest/suburban database to analyze the effect of changing time and weather conditions for single days, therefore we denote the samples for each single day of this database as $X^i_{\text{time of day}}$, where $i = 1, \dots, 7$ is the index of the day. We denote the data pairs of a sample X by vectors $\mathbf{x}_i \in X,\ i = 1, \dots, n$. Since the class of each data pair is known, we divide the data into two classes, the sky class $\mathbf{x}^s_i, i = 1, \dots, n_s$ and the ground class $\mathbf{x}^g_i, i = 1, \dots, n_g$. Supported by the analysis of the influence of n_s and n_g (appendix B.3), we chose to draw $n_s = n_g = 10^5$ data pairs from each class, ground and sky, to realize a sample. Unfortunately, local separation techniques are only valid for single image pairs and need the complete image data since they are trained and tested on single image pairs, while global separation techniques can be trained and tested on the described samples (section 2.2.8). Therefore the training and testing of local separation techniques is performed on the complete image data instead of samples, however, to keep the notation simple, we refer to the raw data in form of images with the same notation.

2.2.6 Data Visualization

For visual inspection, the data obtained from the samples are visualized as a two-dimensional **log UV/G diagram** (e.g. figure 2.5) limited by the extreme values which occurred over all databases. The X-axis shows the irradiance value $\log G$ estimated by the HDR algorithm, the Y-axis the irradiance value $\log UV$, respectively. Each tick on the axes marks a camera stop and both axes have the same scale.

Since the discretized data is represented improperly by a scatter plot, we use a contour plot instead: Each data point is replaced by a (non-normalized) Gaussian function with standard deviation σ. The value at $\mathbf{x} \in \mathbb{R}^2$ for the sky class is then given by

$$d^s_\sigma(\mathbf{x}) = \sum_{i=1}^{n_s} e^{-\frac{\|\mathbf{x}-\mathbf{x}^s_i\|^2}{2\sigma^2}}, \tag{2.1}$$

the value $d^g_\sigma(\mathbf{x})$ of the ground class is calculated analogously. Finally, the values are normalized

$$D^s_\sigma(\mathbf{x}) := \frac{d^s_\sigma(\mathbf{x})}{\max\{d^s_\sigma, d^g_\sigma\}} \text{ and } D^g_\sigma(\mathbf{x}) := \frac{d^g_\sigma(\mathbf{x})}{\max\{d^s_\sigma, d^g_\sigma\}} \tag{2.2}$$

such that arbitrary contour levels in the interval $[0, 1]$ can be plotted.

2.2.7 Classification Rate

For each recorded day, a mask has been drawn by hand which labels each data point dependent on its pixel position as sky or ground. Sometimes it was difficult to choose if a pixel should belong to the sky or ground class, e.g. in semitransparent spots in the canopy. To provide best possible consistency over the data, we chose to always label these points as ground.

The masks are considered as the ground truth to evaluate the performance of the different separation techniques examined in this work. As performance measure for a sample $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ we consider the rate of correctly classified data pairs $\mathbf{x}_i$, in the following called **classification rate**. A formal definition of the classification rate for an arbitrary separation technique is given by

$$R(X) := \frac{\#\{\mathbf{x}_i \in X \mid \mathbf{x}_i \text{ correct classified}\}}{n}. \tag{2.3}$$

If the separation technique needs to be trained and the training set T differs from X, the classification rate is denoted by $R_T(X)$. This notation is used in appendix B.2.

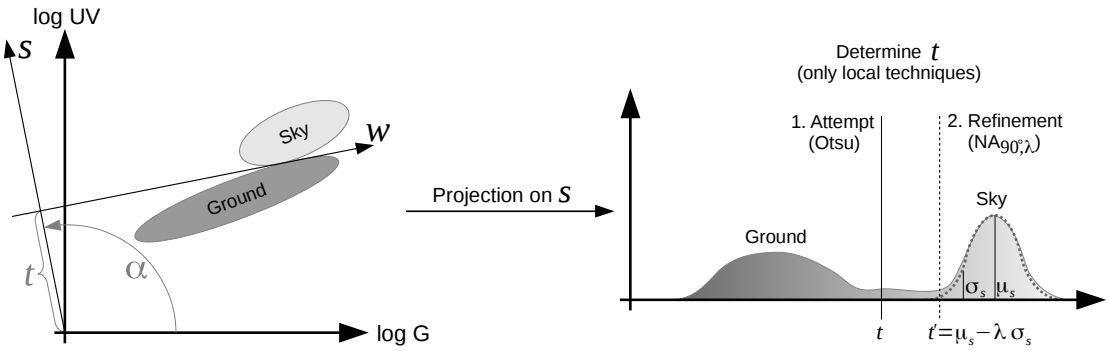


Figure 2.4: Left (projection): Global and local separation techniques project the log UV/G data onto a hyperplane s with angle α. The angle α is always fixed for all global separation techniques as well as the local technique $\mathrm{NA}_{\alpha,\lambda}$. Right (threshold selection): While the threshold value t is learned from a large set of training images for global separation techniques, local separation techniques learn it for each input image pair (log UV and log G) individually. Local separation techniques maximize Otsu's criterion by iterating the threshold values t. Furthermore, the technique $\mathrm{NA}_{\alpha,\lambda}$ approximates a normal distribution $\mathcal{N}(\mu,\sigma)$ to the sky class (based on the threshold t) and refine the threshold value based on this distribution as $t' = \mu - \lambda\sigma$, where λ is a correction factor.

2.2.8 Data Classification

The separation techniques described in this work can be divided into two different classes, which we chose to call global (e.g. *Binary mask*, *Fisher discriminant*) and local separation techniques (e.g. *Otsu method*, *Normal approximation*), to emphasize their behavior. **Global separation techniques** define a decision rule dependent on a given sample of training data. If the camera setup later records an arbitrary scene, the global separation techniques are not influenced by the recorded data. In contrast, **local separation techniques** are applied to a single recorded image and define a decision rule which is depending on and valid only for this image.

For example, the **Fisher discriminant** (section 2.2.8.1) calculates a single separating hyperplane for all entries of the database, while **Otsus method** (section 2.2.8.2) calculates a separating hyperplane for one specific image pair (UV and green) only. An overview of all tested separation techniques can be found in table 2.3.

We primarily want to investigate linear discrimination techniques which allow us to separate the sky and ground classes. Figure 2.4 sketches linear separation of 2D data as well as determining threshold values. In the general case, a linear separator for d-dimensional data containing two classes $\mathbf{x}_i^1 \in \mathbb{R}^d, i = 1,\ldots,n_1$ and $\mathbf{x}_i^2 \in \mathbb{R}^d, i = 1,\ldots,n_2$ is realized by a vector $\mathbf{w} \in \mathbb{R}^d$ and a threshold $t \in \mathbb{R}$, such that for all $\mathbf{x}_i^1$ and $\mathbf{x}_i^2$

$$\mathbf{w}^T\mathbf{x}_i^1 < t < \mathbf{w}^T\mathbf{x}_i^2. \tag{2.4}$$

The separating hyperplane $\mathbf{s}$, which splits the data space into two half-spaces, each containing one class of data, is an affine hyperplane which is orthogonal to $\mathbf{w}$ and passes through the value t on the one-dimensional hyperplane spanned by $\mathbf{w}$. Note that usually the data is not linearly separable. In this case the objective is to find a linear separator which optimizes the data separation regarding a convenient criterion.

If the class affiliation of the data points is known (supervised learning), a wide variety of linear discrimination techniques can be used to calculate $\mathbf{w}$; if not, the problem becomes more complicated and unsupervised learning techniques have to be applied. We chose to use the Fisher discriminant introduced by Fisher (1936) in the first case, while we use the method introduced by Otsu (1979) in the latter, since both methods are related by optimizing a similar criterion.

For our data $\mathbf{x}_i$ with dimension $d = 2$, the objective is to find a normalized vector $\mathbf{w}_\alpha$ with angle α (with respect to the first axis, in our case the green channel) which is used together with the threshold value t to project and separate the data. Regarding the distribution of both

classes in our data (figure 2.5) it can be assumed that for projection angles $-45° \leq \alpha \leq 135°$ the projected value of a data point of the ground class is in the majority of cases smaller than the projected value of a data point of the sky class, i.e. $\mathbf{w}_\alpha^T \mathbf{x}_i^g < \mathbf{w}_\alpha^T \mathbf{x}_i^s$. This assumption may fail for values close to the boundaries, especially around $-45°$. Since the separating hyperplane is the same for $\mathbf{w}_\alpha$ and $\mathbf{w}_\alpha + \pi$ we limit α to this range. In the following we describe the four separation techniques applied in this work.

2.2.8.1 Fisher Discriminant

The Fisher discriminant is a well known linear discrimination technique for supervised learning problems. The idea is to separate the data of two observed classes X_S and X_G with normal distribution by projecting them onto $\mathbf{w}$, such that the projected means are as far apart as possible (between-class variance) and the variances of the projected classes (within-class variance) are minimal. Note that instead of the variances, the scatter of each projected class can be calculated. Both versions can be found in literature (e.g. Bober et al. (2003)), however we chose to use the variance since it is independent of the class sizes. Suppose that $\mathbf{m}_\mathrm{S}, \mathbf{m}_\mathrm{G}$ and $\mathbf{C}_\mathrm{S}, \mathbf{C}_\mathrm{G}$ are the means and covariances of the two classes. Then the projection of the data on $\mathbf{w}$ should be maximized regarding the criterion

$$\mathcal{O}(\mathbf{w}) = \frac{(\mathbf{w}^T(\mathbf{m}_\mathrm{S} - \mathbf{m}_\mathrm{G}))^2}{\mathbf{w}^T(\mathbf{C}_\mathrm{S} + \mathbf{C}_\mathrm{G})\mathbf{w}} = \frac{(\mathbf{w}^T\mathbf{m})^2}{\mathbf{w}^T\mathbf{C}\mathbf{w}} \tag{2.5}$$

with $\mathbf{m} := \mathbf{m}_\mathrm{S} - \mathbf{m}_\mathrm{G}$ and $\mathbf{C} := \mathbf{C}_\mathrm{S} + \mathbf{C}_\mathrm{G}$. A closed-form solution of this problem is

$$\mathbf{w} = a\mathbf{C}^{-1}\mathbf{m} \tag{2.6}$$

for an arbitrary constant $a \in \mathbb{R}$ (Alpaydin, 2004). Now we can choose a such that $\mathbf{w}_\alpha := \mathbf{w}$ is a unit vector. Note that for the Fisher criterion — contrary to local separation techniques where we tested for a set of discrete projection angles — arbitrary angles α are possible. A threshold value t is not calculated by the Fisher discriminant. To find an optimal threshold value, we iteratively search for the threshold value t which maximizes the classification rate $R^{(\mathbf{w}_\alpha, t)}(X)$ of the labeled data.

2.2.8.2 Otsu method

The Otsu method is related to the Fisher discriminant by optimizing a similar criterion for the case of one-dimensional data without given class affiliation (unsupervised). At first, the data is projected on $\mathbf{w}_\alpha$ for a given projection angle α. The idea is now to split the data at each possible threshold t to label the data depending on their half-space affiliation

$$\mathbf{w}_\alpha^T \mathbf{x}_i^g < t < \mathbf{w}_\alpha^T \mathbf{x}_i^s. \tag{2.7}$$

As a consequence, $\mathbf{m}_\mathrm{S}$, $\mathbf{m}_\mathrm{G}$, $\mathbf{C}_\mathrm{S}$ and $\mathbf{C}_\mathrm{G}$ depend on t, such that we get

$$\mathcal{O}(\mathbf{w}_\alpha, t) = \frac{(\mathbf{w}_\alpha^T \mathbf{m}(t))^2}{\mathbf{w}_\alpha^T \mathbf{C}(t) \mathbf{w}_\alpha}, \tag{2.8}$$

where the projection angle α is given.

In contrast to the Fisher criterion, the Otsu method includes a dependency on the number of elements in each class. Denote by p_S and p_G, which also depend on α and t, the probabilities to draw an element of the corresponding class from the set of all data points. Then the criterion optimized by Otsu is (to increase the readability the parameters t and α are not shown)

$$\mathcal{O}'(\mathbf{w}_\alpha, t) = \frac{p_\mathrm{S} p_G (\mathbf{w}^T \mathbf{m})^2}{\mathbf{w}^T (p_\mathrm{S}\mathbf{C}_\mathrm{S} + p_G \mathbf{C}_\mathrm{G})\mathbf{w}}. \tag{2.9}$$

In the following we denote this criterion as *Otsu criterion*. The additional coefficients p_S and p_G guarantee that a small set of outliers does not form a class. Instead, the size of both classes tends to be balanced.

To start from the assumption that the Otsu criterion is large if the data can be split well at t, an optimal separation can be found by searching the threshold value t which maximizes $\mathcal{O}'(\mathbf{w}_\alpha, t)$. This approach has two advantages: First, we are able to perform the Otsu method on the single green or UV channel data by fixing $\alpha = 90°$ or $\alpha = 0°$, respectively. Second, we can search for the best separating hyperplane in the two-dimensional space by additionally iterating α in discrete steps to search the maximum value $\mathcal{O}'(\mathbf{w}_\alpha, t)$.

Since we project the data on the one-dimensional subspace $\mathbf{w}_\alpha$ we can make use of the Otsu algorithm to find an optimal threshold value t which minimizes $\mathcal{O}'(\mathbf{w}_\alpha, t)$. Otsu (1979) showed that in the one-dimensional case it is sufficient either to maximize the between-class or minimize the within-class variance to optimize the criterion $\mathcal{O}'(\mathbf{w}_\alpha, t)$. Furthermore the between-class variance can be calculated for each threshold value t in an iterative way, i.e. the actual between-class variance can be calculated by using the between-class variance of the previous step. The Otsu algorithm combines these two observations such that a fast thresholding algorithm is obtained. Note that we use the Otsu algorithm to calculate the threshold value t for any given projection $\mathbf{w}_\alpha$, and we also calculate the value $\mathcal{O}'(\mathbf{w}_\alpha, t)$ in order to compare the results for varying projections $\mathbf{w}_\alpha$.

Alternatively, a fixed value can be chosen for the projection angle α, in this case we denote it as Otsu_α instead of Otsu. We tested all combinations of values $\alpha \in \{-45°, -40°, \ldots, 135°\}$ for each dataset and chose the value which maximized the classification rate (table 2.6).

As stated above, the Otsu criterion depends on the number of elements in each class and prefers an even distribution of elements in both classes. Therefore it is necessary to test if Otsu's method is influenced by differences in the class sizes, since the images collected in this work contain around the same portion of sky and ground pixel. We cropped the collected images such that the portion between ground and sky pixels differed strongly. The result was that Otsu's method is not noticeable influenced by the different class sizes as long as at least $5-10\%$ of the pixel in each image belonged to either the ground or sky class. Furthermore, images which contain less then $5-10\%$ of sky or ground pixels are not of interest for this work, since it can be assumed that a skyline could not be extracted.

Optimizing the Fisher criterion $\mathcal{O}(\mathbf{w}_\alpha, t)$ for all projection angles α and thresholds t leads to poor clustering results: Often a small set of outliers is misinterpreted as a single class, due to the missing influence of the class sizes. As a result, effectively the whole image is classified as ground or sky.

2.2.8.3 Normal Approximation (NA)

We found that best results of the Otsu method are achieved by using a more sophisticated method enhancing the methods Otsu and Otsu_α, in the following denoted by NA_λ and $\text{NA}_{\alpha,\lambda}$, respectively. These methods perform an additional refinement step for the threshold value t as follows: After classifying the input data using the threshold value t, we fit a normal distribution $\mathcal{N}(\mu, \sigma)$ to the data in the sky class. By setting a new threshold as $t' = \mu - \lambda\sigma$, where λ is an appropriate correction factor, the classification rate can be increased. As for the method Otsu, the values of the projection angle α and the correction factor λ can be chosen arbitrary. We tested all combinations of values $\alpha \in \{-45°, -40°, \ldots, 135°\}$ and $\lambda \in \{0.1, 0.2, \ldots, 5\}$ for each dataset and chose the pair which maximized the classification rate (table 2.6).

2.2.8.4 Binary Mask

With equation (2.2) we are able to compute a binary mask for an arbitrary set of data points $\mathbf{x}_i^s$ and $\mathbf{x}_i^g$. The function

$$M_\sigma(\mathbf{x}) = \text{sgn}(D_\sigma^s(\mathbf{x}) - D_\sigma^g(\mathbf{x})) \tag{2.10}$$

Notation	Type	Method	α	Training	
				Dataset	Variable
$\mathbf{w}_{\text{UV}}$, $\mathbf{w}_{\text{G}}$, $\mathbf{w}_{\text{con}}$	Global	Linear separator	Fixed	X_{8-19}	t
$\mathbf{w}_{\text{F}}$	Global	Linear separator	Trained	X_{8-19}	t
M_{7-20}	Global	Binary mask	-	X_{7-20}	M_{7-20}
M_{8-19}	Global	Binary mask	-	X_{8-19}	M_{8-19}
Otsu_α	Local	Otsu method	Trained	X_{8-19}	α
Otsu	Local	Otsu method	Variable	-	-
$\text{NA}_{\alpha,\lambda}$	Local	Normal approx.	Trained	X_{8-19}	α, λ
NA_λ	Local	Normal approx.	Variable	X_{8-19}	λ

Table 2.3: Overview of all implemented and tested separators.

can be used as a decision rule for any value $\mathbf{x}$ which yields that $\mathbf{x}$ belongs to the sky class if $M_\sigma(\mathbf{x}) = 1$ and to the ground class if $M_\sigma(\mathbf{x}) = -1$. Otherwise no choice can be made. In order to increase the readability of the corresponding plots (see figure 2.11), all points $\mathbf{x}$ with $D^s_\sigma(\mathbf{x}) + D^g_\sigma(\mathbf{x}) < \varepsilon$ for the threshold value $\varepsilon = 10^{-6}$ are also classified as ground points.

2.2.9 Overview: Tested Separation Techniques

Based on the separation techniques presented in section 2.2.8, we now define different separators which are tested on the data samples X_{8-19} and X_{7-20}. An overview can be found in table 2.3.

Global separation techniques: Each global linear separator $(\mathbf{w}_\alpha, t)$ is uniquely described by its angle α and the threshold value t. Since the type of the global linear separator is mainly described by its angle α, we denote the separator as $\mathbf{w}_\alpha$ for specific values α. The most simple global linear separators are $\mathbf{w}_{\text{UV}} := \mathbf{w}_{90^\circ}$ and $\mathbf{w}_{\text{G}} := \mathbf{w}_{0^\circ}$ which project the data pairs to their UV or G value. This approach corresponds to a threshold separation in the single UV or green channel, respectively. The global linear separator $\mathbf{w}_{\text{con}} := \mathbf{w}_{135^\circ}$ is equivalent to the contrast measure or logarithmic ratio $r := \log \frac{UV}{G} = \log UV - \log G$. The basic idea described by Möller (2002) was to classify a data point by its ratio r. For data points of the sky class, this ratio should fulfill $r > 0$ while data points with a ratio $r < 0$ should belong to the ground class. Furthermore, Kollmeier et al. (2007) used the Fisher discriminant to find an angle α_F which optimizes the data separation regarding the Fisher criterion for a training data set X. We denote the corresponding linear separator as $\mathbf{w}_{\text{F}}$. To obtain the threshold values t (and the projection angle α_F for $\mathbf{w}_{\text{F}}$) needed to separate the data, the separators $\mathbf{w}_{\text{F}}, \mathbf{w}_{\text{con}}, \mathbf{w}_{\text{UV}}, \mathbf{w}_{\text{G}}$ are trained on X_{8-19}. Additionally to the linear separators we implemented the binary masks M_{8-19} and M_{7-20} for the corresponding data sets X_{8-19} and X_{7-20} using equation (2.10) with $\sigma = 1$ (approximately 0.05 stops).

In appendix B.2 (the results are shown in table 2.5) we perform a cross-correlation test on the forest/suburban database. The results show that overfitting does only marginally occur for single channel separation. This is to be expected, since the separator has only one degree of freedom (the threshold value) and the amount of collected data is huge. For contrast based separators $(\mathbf{w}_{\text{F}}, \mathbf{w}_{\text{con}})$ there are up to two degrees of freedom (the threshold value and for the Fisher discriminant the projection angle) such that the influence of training and test data is increased. Since we only collected a single day for each mineral skyline, we are not able to perform cross-correlation tests as previously done for the forest/suburban database, however we believe that the total amount of data collected in this work now covers a wide range of ground objects and various sky conditions. Therefore, the learned values should generalize well to novel log UV/G data. Moreover, training is done on a randomly drawn set of sample points from the respective database, while the tests are not only performed on a different set of sample points (table 2.4) but also on the complete set of HDR images (UV and green, figure 2.13). Note that for binary masks, overfitting the training data (the log UV/G data) is done on purpose to obtain an upper bound for the classification rate which can be obtained by a global (non-linear) separator.

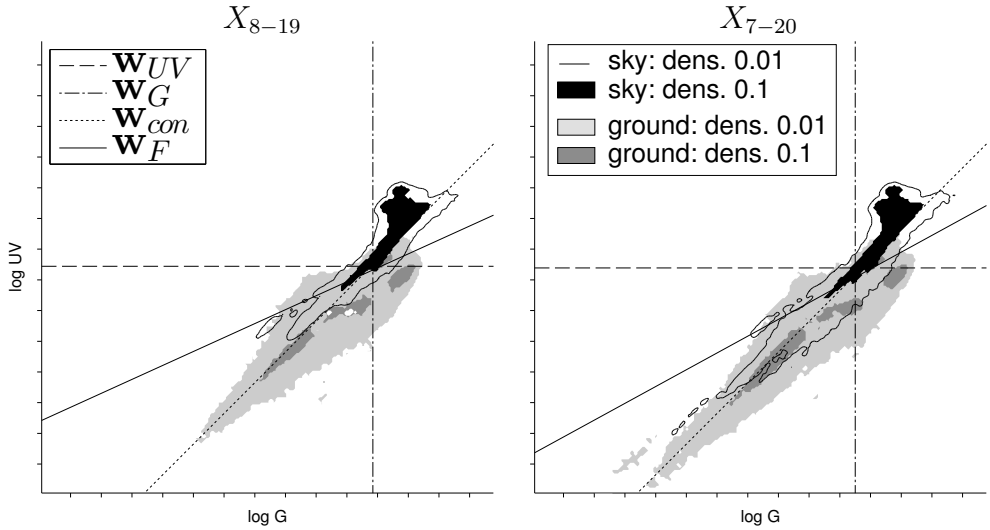


Figure 2.5: The plot shows the pooled data of all recorded datasets, where the same number of sample points has been taken from each of the four different datasets presented in figure 2.6 (*stones*, *sand*, *earth*, *forest/suburban*). The lines show the global linear separators which maximize the classification rate for each presented log UV/G dataset: $\mathbf{w}_{\mathrm{UV}}$ (UV-only), $\mathbf{w}_{\mathrm{G}}$ (green-only), $\mathbf{w}_{\mathrm{con}}$ (contrast: 1:1), and $\mathbf{w}_{\mathrm{F}}$ (contrast: Fisher discriminant). The classification rate of these separators can be found in table 2.4. The black region and the black outline show the density level of the sky class for the levels 0.1 and 0.01 Analogously, these levels are represented for the ground class by dark and light gray areas (section 2.2.6). On each tick mark, the irradiance value is doubled, representing a 'stop' in camera terminology.

Local separation techniques: Both local separation techniques (Otsu and NA) can be implemented in the same way as global separators, however the threshold value — and in some cases the projection angle — is determined for each image pair individually. By fixing α (both methods) and λ (NA only), these techniques are reduced to calculate the threshold t for the corresponding projection $\mathbf{w}_\alpha$. These separators are denoted as Otsu_α and $\mathrm{NA}_{\alpha,\lambda}$. With variable parameters, α can be determined by iterating $\alpha \in [-45°, 135°]$ in discrete steps of 5°, and the α value which maximizes the optimization criterion $\mathcal{O}(\mathbf{w}_\alpha, t)$ is used for the classification. This optimization has to be performed again for each input image X_i. We denote these separators as Otsu and NA. Note that NA_λ needs a fixed value λ to readjust the threshold value. Appropriate values for λ have been determined heuristically and are shown in table 2.6.

2.3 Results

2.3.1 Collected Data

For visual inspection, the log UV/G diagrams (section 2.2.6) are presented for different samples X. The log UV/G plots of the three days recorded with different mineral skylines (stones, sand, earth) as well as the log UV/G plot of the forest/suburban dataset are shown in figure 2.6. Note that only samples of the ground objects are restricted to the specific databases, while the sky data is always chosen from the complete sky data collected in all databases to enhance the variation of sky data as much as possible. Figure 2.5 shows the samples X_{7-20} and X_{8-19} representing the pooled data recorded in all skyline databases for the specified times of day. These times were chosen regarding the performance of the global separation techniques presented in figure 2.10 and are discussed later (section 2.3.2). In the plots, the contour levels 0.01 and 0.1 are highlighted, and we refer to them as border and center region, respectively. As can be seen, the border and center regions of both the sky and ground class cover confined regions. The sky and ground classes are distributed along two slightly offset lines with approximately unity slope. The center regions are compact and can be separated in both samples. However, the border regions of both classes

Global	$\mathbf{X_{8\text{-}19}}$				
	UV	**Green**	**Contrast**	**Fisher**	**Mask**
Stones	84%	78%	70%	84%	88%
Sand	**88%**	**60%**	**89%**	**94%**	**95%**
Earth	94%	88%	78%	94%	95%
Forest/Suburban	95%	92%	80%	96%	97%
All	88%	79%	79%	89%	92%

Global	$\mathbf{X_{7\text{-}20}}$				
	UV	**Green**	**Contrast**	**Fisher**	**Mask**
Stones	79%	75%	67%	79%	84%
Sand	**83%**	**60%**	**86%**	**89%**	**91%**
Earth	89%	84%	73%	88%	90%
Forest/Suburban	91%	89%	77%	92%	93%
All	83%	75%	75%	84%	87%

Table 2.4: The classification rates of the global separators introduced in section 2.2.8 — the linear separators $\mathbf{w}_{\mathrm{UV}}$ (UV-only), $\mathbf{w}_{\mathrm{G}}$ (green-only), $\mathbf{w}_{\mathrm{con}}$ (contrast: 1:1), $\mathbf{w}_{\mathrm{F}}$ (contrast: Fisher discriminant), and the binary masks (non-linear UV/G separator) — applied to the datasets shown in figures 2.5 and 2.6. The separation methods were trained and tested on two different samples (each containing 10^5 elements of the sky and ground class) drawn from the specified databases. An evaluation using single HDR image pairs (e.g. as captured by a mobile robot) as test data instead can be found in figure 2.13. Only for sand (highlighted), the classification rates show a noticeably increased performance for the UV/G contrast (Fisher discriminant) compared to UV-only separation. In all other cases, both methods show a similar performance, both slightly worse compared to the best possible performance of the binary masks.

cover a wide range of data points along these lines and overlap due to the small vertical distance between the lines. Compared to X_{8-19}, this overlap is stronger for sample X_{7-20} which contains data points recorded at additional hours during dawn and dusk.

For both samples X_{7-20} and X_{8-19}, the global linear separators $\mathbf{w}_{\mathrm{UV}}$, $\mathbf{w}_{\mathrm{G}}$ and $\mathbf{w}_{\mathrm{F}}$ are almost able to separate the center regions of the sky and ground classes. The contrast mechanism $\mathbf{w}_{\mathrm{con}}$ is also able to separate the major fraction of both center regions, but noticeable misclassifications are visible. Due to the strong overlap between the border regions, all global linear separators suffer from classification errors for data points in these regions. The classification rates of all separators on these samples are discussed later (sections 2.3.2 and 2.4.1) and can be seen in table 2.4. They confirm the visual impression that $\mathbf{w}_{\mathrm{UV}}$ and $\mathbf{w}_{\mathrm{F}}$ have approximately the same performance, whereas $\mathbf{w}_{\mathrm{G}}$ and the contrast mechanism $\mathbf{w}_{\mathrm{con}}$ perform noticeably worse.

An overview over the individual log UV/G plots of the three days recorded with different mineral skylines (stones, sand, earth) as well as the log UV/G plot of the complete forest/suburban dataset are shown in figure 2.6. As can be seen, all three mineral datasets show comparable distributions of the log UV/G data for ground objects, however the UV portion in the UV/G contrast is higher in the stone skyline compared to sand and earth skylines. Furthermore, the ground objects form mainly two clusters (which can clearly be seen in figure 2.15, compare section 2.3.6). This is a consequence of the lighting conditions, since each log UV/G data point (mostly) corresponds either to an object lying in the direct sun (higher amount of green compared to UV) or in the shadows (higher amount of UV compared to green). On the recording days the sky was mainly clear such that purely diffuse lighting occurred only rarely.

Larger differences can be found by comparing the three mineral skylines to the data collected in the forest/suburban database. It can be seen that the overall brightness of the collected ground objects in the forest/suburban database is lower compared to the mineral skylines, since recordings also include multiple days with bad weather conditions. Furthermore, the forest/suburban

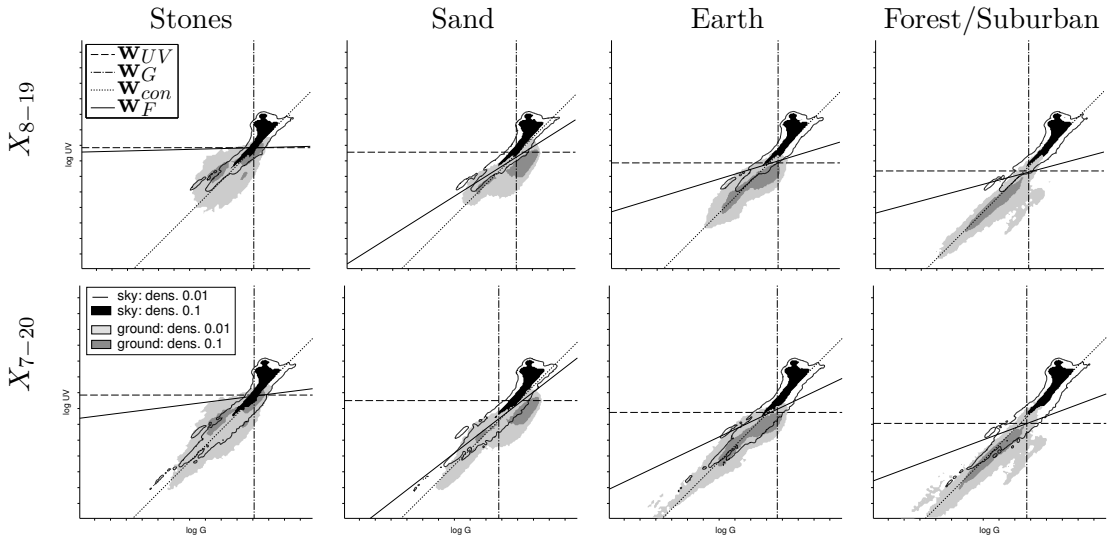


Figure 2.6: The columns show the log UV/G plots of the three mineral skylines (*stones*, *sand*, *earth*) and the skyline dominated by trees (*forest/suburban*). Note that in each plot the pooled sky data of all skyline databases are shown, representing a wide variety of weather conditions (the slightly different appearance is due to normalization). Each row shows the data recorded at daytimes between $8:00-20:00$ and $7:00-21:00$, respectively. See figure 2.5 for visualization notes.

database was collected over seven days with differing skylines with a wider variety of collected data, which leads to a higher spread in the corresponding log UV/G plot.

To examine the influence of weather conditions over several days in similar environments, figure 2.7 shows the samples X^i_{8-19} for each single day $i = 1, \ldots, 7$ of the forest/suburban database between $8:00-20:00$. Overall, the data of individual days look similar and only differ in specific points, the two most noticeable differences are caused by light reflected from roofs and facades (figure 2.7, day 2, A) and effects of a storm (figure 2.7, day 7, B). Comparing table 2.2 with figure 2.7, it can be observed that the greatest influence during the different days is caused by the weather: The first three days have in common that especially the sky classes form an elongated ellipse along a line with unity slope. As a consequence of the changing weather conditions over each of these days, the recorded values cover a broad range of different illumination levels over the day. In contrast, days 4-6 were sunny, leading to more compact ground and sky classes. On day 7 both weather conditions appeared: During the morning, heavy rain occurred at the record location, while the afternoon was sunny. As a result, both features described for days 1-3 and 4-6 can be observed. Interestingly, the position of the global linear separators $\mathbf{w}_{\text{con}}$, $\mathbf{w}_{\text{UV}}$ and $\mathbf{w}_{\text{G}}$ over all seven days is nearly the same. That means that the different conditions (and therefore training sets) only have a small effect on the resulting linear separators. However, the orientation and position of the global linear separator $\mathbf{w}_{\text{F}}$ varies noticeable over the seven days. As a result the classification rate may change noticeably if the training and test sets for $\mathbf{w}_{\text{F}}$ vary. This subject is addressed in more detail in section 2.3.2.

In each of the two preceding studies by Möller (2002) and Kollmeier et al. (2007), a sensor was built to collect a database with log UV/G data points. The construction of these sensors (handheld devices with only a single photodiode per channel) differs from the sensor used here (stationary setup with one CCD camera per channel). Since the results of this chapter partly stand in conflict with the conclusions drawn in these studies, we are interested in a comparison of the collected data. The high resolution of the cameras permits a sharp distinction between ground and class objects which allows the examination of points at boundaries between the sky and ground objects. This distinction was not possible with the preceding handheld sensors (around $10°$ full opening angle), therefore only patches were selected such that the sensor ideally covered only points belonging either to ground or sky class and thus no points at the border between

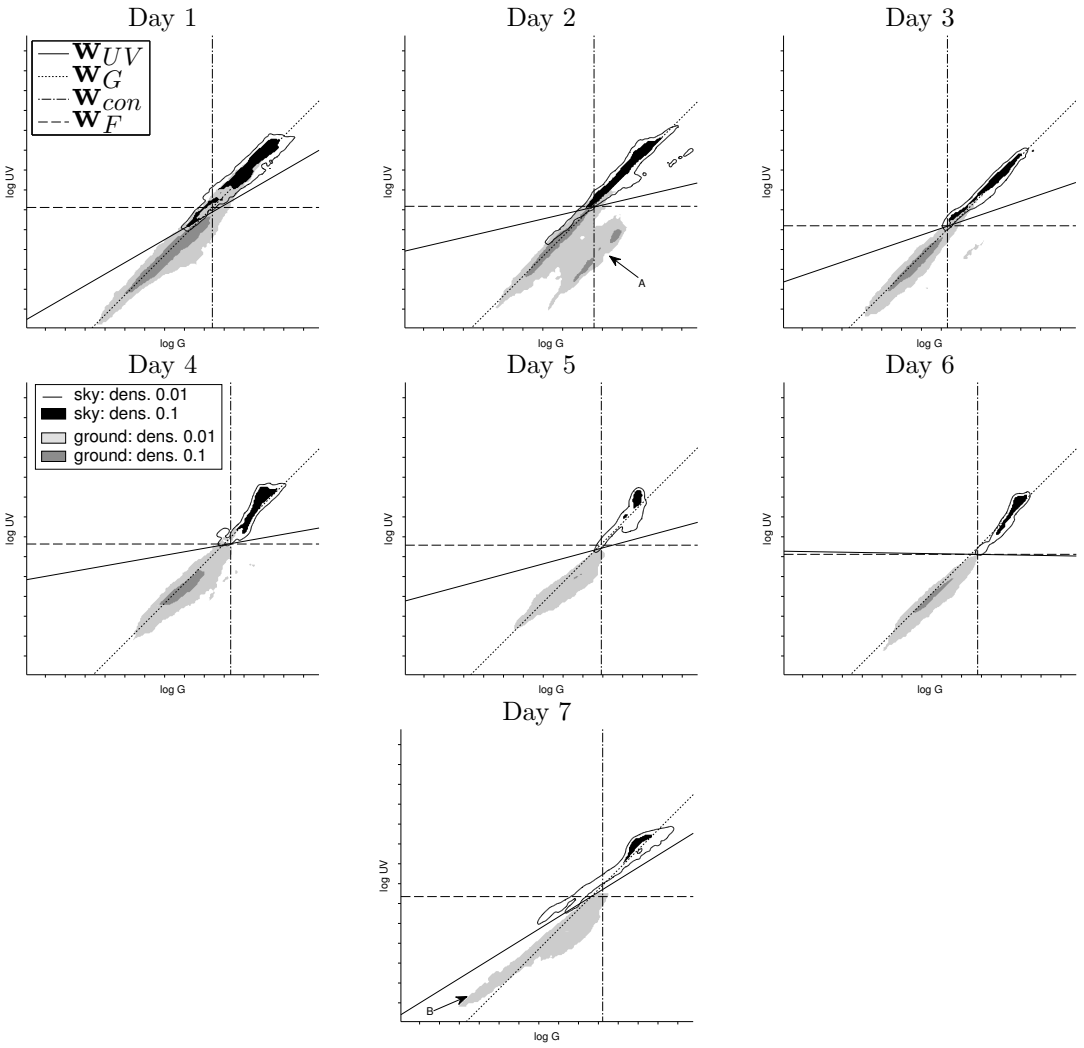


Figure 2.7: The plots of the samples $X^i_{8-19}, i = 1, \ldots, 7$ (from top left to bottom right), which represent the single days in the time between $8:00 - 20:00$, are shown to allow a direct comparison between different days. The classification rates for each day are shown in figure 2.10b. See figure 2.5 for visualization notes.

sky and ground were collected. However, these points on the border are particularly difficult to classify and may contain crucial information for navigational purposes (examples 2.4 and 2.5). Note that in Kollmeier et al. (2007) and this study the filters used — and therefore the filter characteristics — are the same. In Möller (2002) different filters were used, however the resulting filter characteristics were close to those used in this work (Möller (2002), figure 1). Figure 2.8 shows a data plot of the preceding studies and the corresponding time distributions at which the data points were collected. Furthermore the combined time distribution of the preceding studies is used to create a sample from the data points over all seven days collected in this work. Due to the comparably few data points collected in the preceding studies (both around 650 data points), the contour levels were adjusted in both plots such that the log UV/G diagrams are visually better comparable (figure 2.8). Even though the recording conditions differ in several aspects from this study with respect to the location, time of the year, or weather, the results are qualitatively similar. All classes in the diagrams show the elongated ellipsoid form, typical for the record over a wide range of times. Clearly the sky and ground classes in the study by Möller (2002) are more elongated compared to the other two studies. Especially the ground class is particularly scattered over a wider area. Reasons are the small number of data points, which leads to a widely spread class border, and that a wider range of different ground objects (e.g. stones, gravel,

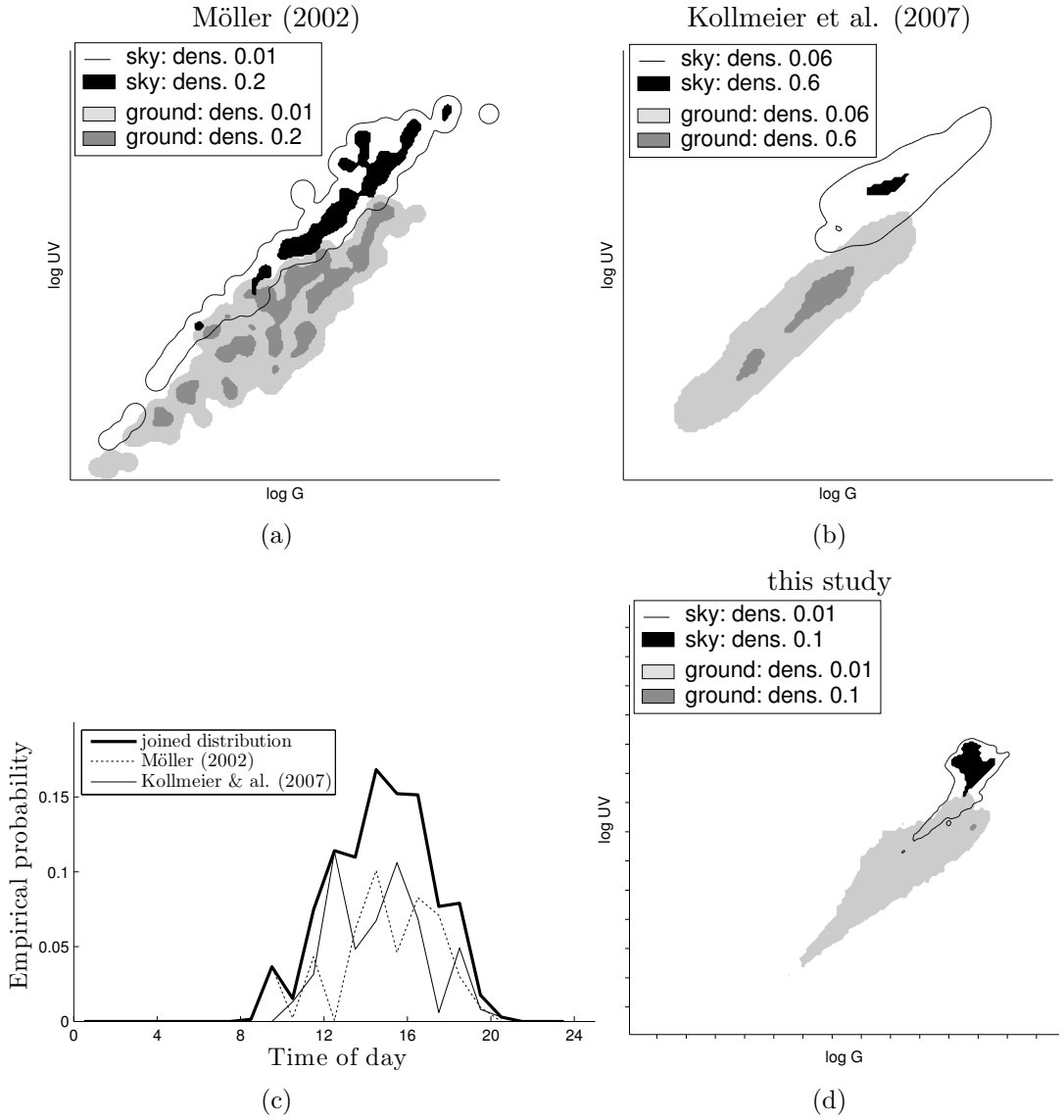


Figure 2.8: The experiments performed by (a) Möller (2002) and (b) Kollmeier et al. (2007) are plotted in the same way as described in section 2.2.6. Since comparably few data points were collected, the contour levels are adjusted such that the plots are better visually comparable to the plots presented in this work. A joined histogram of the times at which the samples presented in (a) and (b) were collected is shown in (c) (thick line: joined distribution; solid and dotted line: data by Kollmeier et al. (2007) and Möller (2002), respectively). By creating a sample over all seven days, but with the given joined time distribution shown in (c), a plot is created from the all ten skyline databases collected in this work (d) which can be compared to the original experiments. See figure 2.5 for visualization notes.

trees) under strongly varying lighting conditions were recorded. The data collected by Kollmeier et al. (2007) are nearly completely separable for both the center and border regions. However the number of collected data points is again relatively small, and nearly no data were collected in the morning or evening. Compared with the data collected in this work, all diagrams have in common that a contrast mechanism (i.e. $\mathbf{w}_{\text{con}}$) could mostly separate the sky and ground classes, even though the quality may be insufficient for further applications like visual navigation. This is partly attributable to the fact that the classification is very sensitive to environmental changes, since the corresponding data points lie close to the separating hyperplane. Thus already small environmental changes could lead to strong differences in the classification.

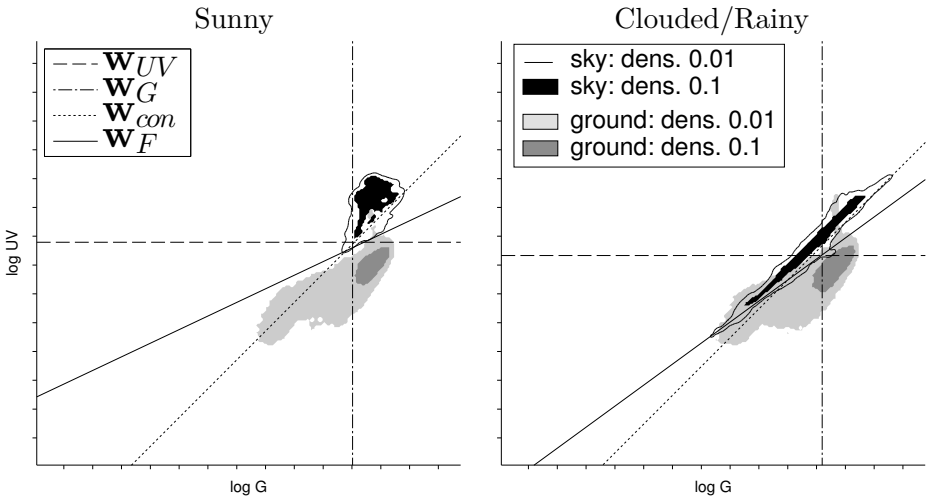


Figure 2.9: The plot shows the log UV/G data for the sand skyline together with the sky data (selected from all databases) of sunny days (left) and days where the sky was cloudy, including rain (right). While the classification rate is similar in both cases using the Fisher discriminant $\mathbf{w}_{\mathrm{F}}$ (98.7% and 93.0%), it differs strongly for the UV-only separator $\mathbf{w}_{\mathrm{UV}}$ (98.3% and 74.4%). See figure 2.5 for visualization notes.

2.3.2 Global Separation Techniques

Except for the stone skyline, the plots from figures 2.5 and 2.6 show that the ground portion tends to have a higher amount of green light, while the sky portion has a higher amount of UV light. This supports the idea that a log UV/G contrast measure (not necessarily 1:1) can be used to classify the data. Table 2.4 shows that both the global linear separators $\mathbf{w}_{\mathrm{UV}}$ (UV-only) and $\mathbf{w}_{\mathrm{F}}$ (Fisher discriminant) have classification rates close to the maximal achievable rates (binary masks, section 2.2.8.4) for all datasets, except of the sand skyline dataset. However, in comparison to $\mathbf{w}_{\mathrm{UV}}$ and $\mathbf{w}_{\mathrm{F}}$, the classification rates of $\mathbf{w}_{\mathrm{G}}$ and $\mathbf{w}_{\mathrm{con}}$ are not competitive with values around 80%. Besides the sand dataset, only a gain of maximal 1% is achieved by using $\mathbf{w}_{\mathrm{F}}$ on log UV/G data instead of $\mathbf{w}_{\mathrm{UV}}$ on the log UV data only. For the sand dataset we found a high discrepancy between the classification rates achieved by $\mathbf{w}_{\mathrm{UV}}$ and $\mathbf{w}_{\mathrm{F}}$ with 88.1% and 94.4% for dataset X_{7-20} and 82.2% and 89.0% for dataset X_{7-20}, respectively.

In figure 2.9 we show two additional plots of the sand dataset, but using two different kinds of sky data: one time from sunny days and one time from cloudy and rainy days (selected from all skyline databases). While both classes can easily be separated using the sky data of the sunny days — achieving classification rates of 98.7% and 98.3% for $\mathbf{w}_{\mathrm{UV}}$ and $\mathbf{w}_{\mathrm{F}}$, respectively — the classification gets more demanding when using the sky data of the cloudy/rainy days. In this case $\mathbf{w}_{\mathrm{UV}}$ has a classification rate of 74.4%, while $\mathbf{w}_{\mathrm{F}}$ still achieves 93.0%. We could observe this big difference between $\mathbf{w}_{\mathrm{UV}}$ and $\mathbf{w}_{\mathrm{F}}$ only in this special setup (sand skyline under cloudy/rainy weather conditions for the sky portion), in all other combinations they showed comparable performances.

As mentioned before, the recording conditions (location, time of the year, weather, etc.) influence the performance of the global linear separators. Figure 2.10 shows the classification rates of all global linear separators trained and tested on different times or days on the forest/suburban database. In figure 2.10 (a), each separator was trained and tested on a sample X_h created from the data points captured over all days from the specific hour starting at h. This allows us to examine the stability of each separator over the time of a day, showing interesting results: All separators start to show first positive results around 7:00 and stop working around 21:00, while the best results are achieved between $8:00-20:00$, which motivates the selection of the daytimes represented by the samples X_{7-20} and X_{8-19}. The performance of the separators differs strongly. The Fisher discriminant $\mathbf{w}_{\mathrm{F}}$ always leads to the best results. However, the performance of the

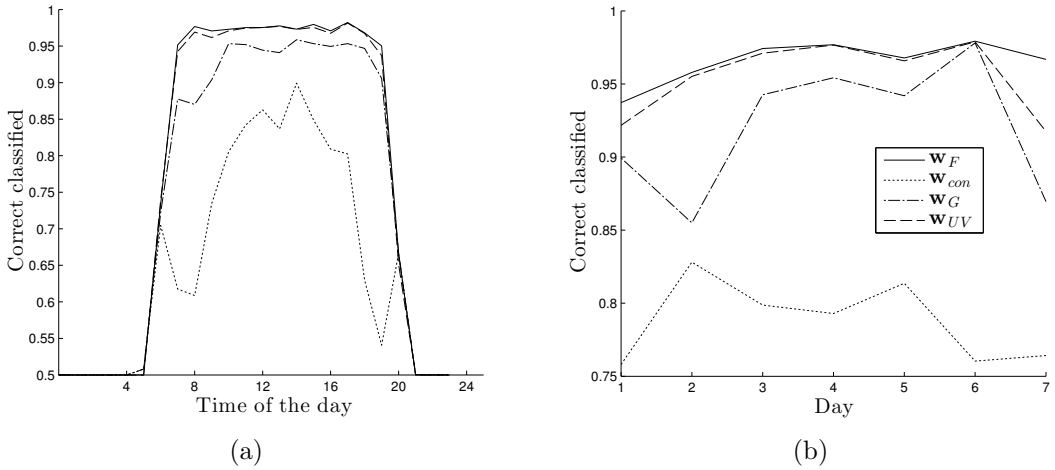
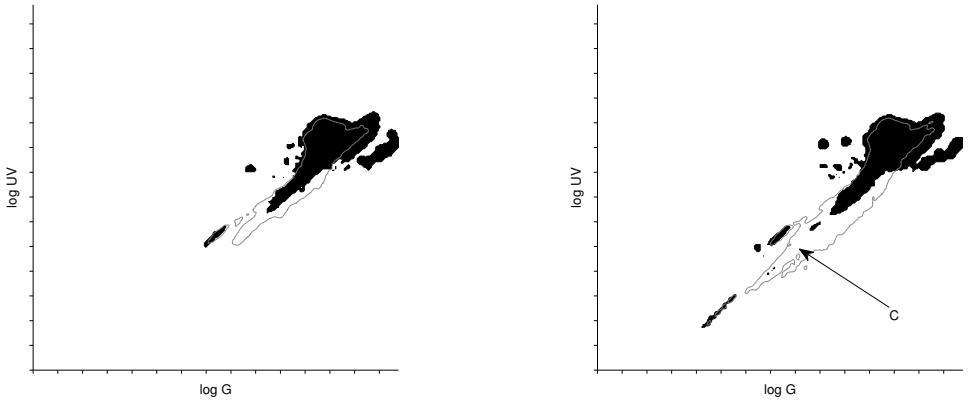


Figure 2.10: The figure shows the classification rates of the global separators on the forest/suburban database. It can be seen, that they strongly depend on the day and time at which the sample has been recorded. The optimal global separators have been trained on and applied to (a) each hour over all days ($X_h, h = 0, \ldots, 23$) and (b) each day in the time between $8:00-20:00$ ($X^d_{8-19}, d = 1, \ldots, 7$).

UV-only separator $\mathbf{w}_{\mathrm{UV}}$ is only marginally weaker. The green-only separator $\mathbf{w}_{\mathrm{G}}$ is less stable around the morning and evening and shows a stable, but worse classification rate during noon and afternoon. The contrast mechanism $\mathbf{w}_{\mathrm{con}}$ only shows acceptable classification rates around noon with strong light intensities. In the morning and evening, its classification rate drops significantly to around 0.6 which is only slightly better than random class separation. In figure 2.10 (b), the global linear separators were trained and tested on the samples X^i_{8-19} for a specific day i in the forest/suburban database. Again the Fisher discriminant $\mathbf{w}_{\mathrm{F}}$ and the UV separator $\mathbf{w}_{\mathrm{UV}}$ show the best classification rates over all days with a slightly better performance on the sunny days $4-7$ (table 2.2). The classification rate of the green-only separator $\mathbf{w}_{\mathrm{G}}$ shows the same preference for sunny days, but suffers from strong fluctuations of the classification rates over the remaining days. The fluctuations may be caused by the portion of artificial objects in the scene with strong specular reflectance, particularly on day 2 and 7, reflecting sunlight directly onto the camera, whereas the artificial objects in the scenes on days $3, 4$ and 6 rarely reflected incoming sunlight directly onto the camera (example 2.2). The classification rate of the contrast mechanism $\mathbf{w}_{\mathrm{con}}$ shows weak performance compared to the other global linear separators and also suffers from strong fluctuations over the different days. However, no clear relations between the scenes and the weather are evident for the contrast mechanism.

The upper limit for the classification rates of all global separators for a given sample is obtained by creating a mask as described in section 2.2.8.4. Note that if no class affiliation could be chosen for a data point (i.e. $D^s_\sigma(\mathbf{x}) + D^g_\sigma(\mathbf{x}) < \varepsilon$), it is assigned to the ground class. If trained and tested on the same sample, the masks M_{7-20} and M_{8-19} are the best possible global separators for the samples X_{7-20} and X_{8-19}. Both masks are shown in figure 2.11. As could be expected, for both masks the black region, which assigns each data point within to the sky class, roughly coincides with the gray outline of the contour level 0.01 of the sky class. While the border region of the sky class of sample X_{8-19} is nearly congruent with its mask M_{8-19}, this is not the case for sample X_{7-20} together with its mask M_{7-20}: The sky class is not completely classified by the mask and a part of the border region (figure 2.11 (b), C) is classified as ground class. This is a result of the overlap between the border regions of the sky and ground class since the sample contains additional data collected in the morning and evening.

Since global separation techniques need to be trained, it is necessary to examine also the influence of the training and test data on the classification rate. To test this we trained each separator on the data collected on six of the seven days of the forest/suburban database and performed two



(a) (b)

Figure 2.11: The binary masks (a) M_{7-20} and (b) M_{8-19} created with the decision rule defined by equation (2.10) trained on the corresponding samples X_{7-20} and X_{8-19} (sky: black; ground: white). For visual comparison, the 0.01 contour line of the sky class of the corresponding sample is added to each plot (gray line). See figure 2.5 for visualization notes.

		$\mathbf{w_F}$	$\mathbf{w_{con}}$	$\mathbf{w_{UV}}$	$\mathbf{w_G}$	$\mathbf{M_{8-19}}$	$\mathbf{M_{7-20}}$
$\mathbf{X_{8\text{-}19}}$	t-test	$-$	$++$	$++$	$++$	$++$	$++$
	WSR	$\circ$	$++$	$++$	$++$	$++$	$++$
$\mathbf{X_{7\text{-}20}}$	t-test	$-$	$++$	$\circ$	$++$	$++$	$++$
	WSR	$--$	$+$	$\circ$	$++$	$+$	$++$

Table 2.5: To test if global separation methods generalize well to new data sets, the classification rates on training data sets and unknown data sets are compared. The table shows the p-values of the corresponding statistical tests (B.2) for all global separation methods. The p-values are represented by symbols which split $p \in [0, 1]$ into five equally sized intervals: Minus symbols stand for small p-values, plus values for the opposite ($p \in [0, 0.2] : --$, $p \in [0.2, 0.4] : -$, $\ldots$, $p \in [0.8, 1] : ++$). Global separation methods which tend to generalize well to new data are therefore marked with plus symbols.

experiments: First, we tested the separator on the remaining seventh day (test-set T_1). Second, we tested the separator on one of the six training days (test-set T_2). Then we compared the distribution of the test-sets T_1 and T_2: If a separator generalizes well to new/unknown data, the test-sets should have the same distribution, i.e. the classification rates do not differ significantly. The analysis of the distributions of the test-sets has been done in appendix B.2 in detail, table 2.5 shows the results for all global separation techniques. As can be seen, most global separation techniques ($\mathbf{w}_{\text{con}}$, $\mathbf{w}_{\text{G}}$, M_{8-19}, M_{7-20}) seem to generalize well to unknown data. Separator $\mathbf{w}_{\text{F}}$ shows strong variation between the test-sets, leading to the assumption that it cannot cope well with unknown data. Finally separator $\mathbf{w}_{\text{UV}}$ shows an interesting behavior: While it generalizes well to new data only if daytime samples providing strong sunlight intensities are considered (X_{8-19}), it struggles to cope with new situations including dim light conditions (X_{7-20}).

2.3.3 Local Separation Techniques

As the global separation techniques — which learn the best threshold value t for a given sample such that the class separation is optimal — the local separation techniques depend on up to two parameters which can be optimized by supervised learning. Four different versions of Otsu's method are described in this work of which one does not depend on any parameter trained in a supervised way (Otsu). The remaining local separation techniques depend on one (Otsu_α, NA_λ) or two parameters ($\text{NA}_{\alpha,\lambda}$). The parameters α and λ for these methods are trained for each database individually to maximize the classification rates on the sample X_{8-19}. Examples of determining

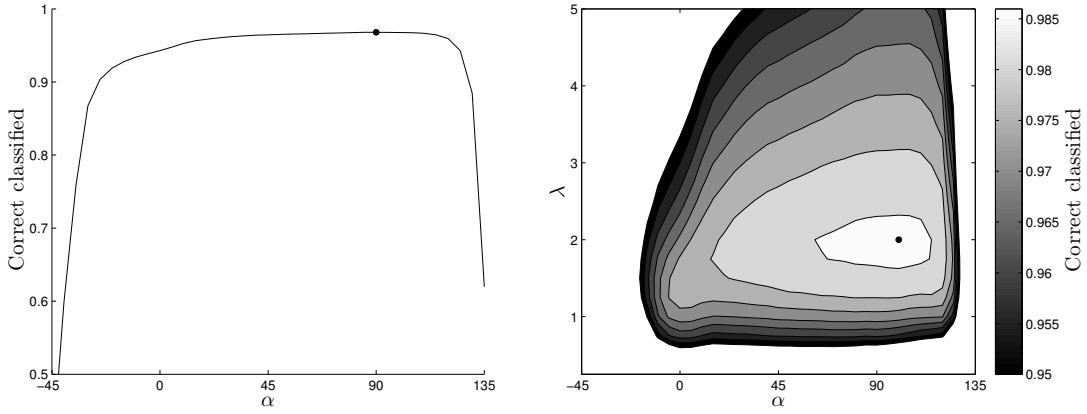


Figure 2.12: The classification rate (on the forest/suburban database) of the separation methods (a) Otsu_α and (b) $\text{NA}_{\alpha,\lambda}$ is depending on the parameters α and λ. Both methods are tested with varying parameter values over the data set X_{8-19}. The parameters with the maximal classification rate are marked by a filled circle. As can be seen, both methods perform well for angles close to 90°. The value $\lambda = 2$ shows the best performance for $\text{NA}_{\alpha,\lambda}$.

the values α (and λ) for Otsu_α and $\text{NA}_{\alpha,\lambda}$ on the forest/suburban database are shown in figure 2.12. As can be seen, the training of the parameter α for Otsu_α shows a peak at $\alpha = 85°$, however the gain compared to $\alpha = 90°$ (UV-only separation) is small. Outside of the region $-40° \leq \alpha \leq 130°$, the classification rate drops noticeably due to the difficulty in deciding which class is which on the projected hyperplanes (hand-labeling would be necessary). Similar results are observed for the training of the parameters α and λ for $\text{NA}_{\alpha,\lambda}$. For $\text{NA}_{\alpha,\lambda}$, both parameters are changed simultaneously, showing best results for $\alpha = 100°$ and $\lambda = 2$. Again the gain compared to $\alpha = 90°$ (UV-only separation) is small. Recalling the definition of $\text{NA}_{\alpha,\lambda}$ (section 2.2.8), the normal distribution fitted to the sky class is in this case limited to the interval $[\mu_s - 2\sigma_s, +\infty]$. This interval theoretically contains approximately 97.7% of the data points of the sky class, the remaining 2.3% strongly overlap with the ground class and are removed from the sky and assigned to the ground class.

As described in section 2.2.8.2, the parameterless Otsu method chooses the projection plane $\mathbf{w}_\alpha$ such that the Otsu criterion is maximized. Figure 2.14 shows a detailed plot for the Otsu separation applied on a scene captured on 02. Sept. 2014 at 8:10 (example 2.5). As can be seen, the projection plane $\mathbf{w}_\alpha$ which maximizes the Otsu criterion is not necessarily the projection plane which maximizes the classification rate. Two peaks for the criterion can be observed at around 90° and −10° which correspond to UV-only and approximately green-only separation. However the UV-only separation achieves a classification rate around 97%, while the green-only separation is only around 95%.

We tested the local separation techniques on all databases and the results are presented in table 2.6. They show a superior performance to the global separation techniques by achieving a classification rate of up to 99%. The correction values λ which showed the best performances were between $3.0 - 3.6$ for the mineral skylines and 2.0 for the forest/suburban skyline. While the correction value λ should be chosen depending on the environment and weather conditions to achieve best results, it still outperformed all other methods for all values $\lambda \in [2, 4]$ on all databases. In the following, we set the projection angle $\alpha = 90°$ such that Otsu_α and $\text{NA}_{\alpha,\lambda}$ perform a UV-only separation. Furthermore, we set $\lambda = 3.0$ for NA_λ and $\text{NA}_{\alpha,\lambda}$ as a trade-off between the best values found for λ (table 2.6).

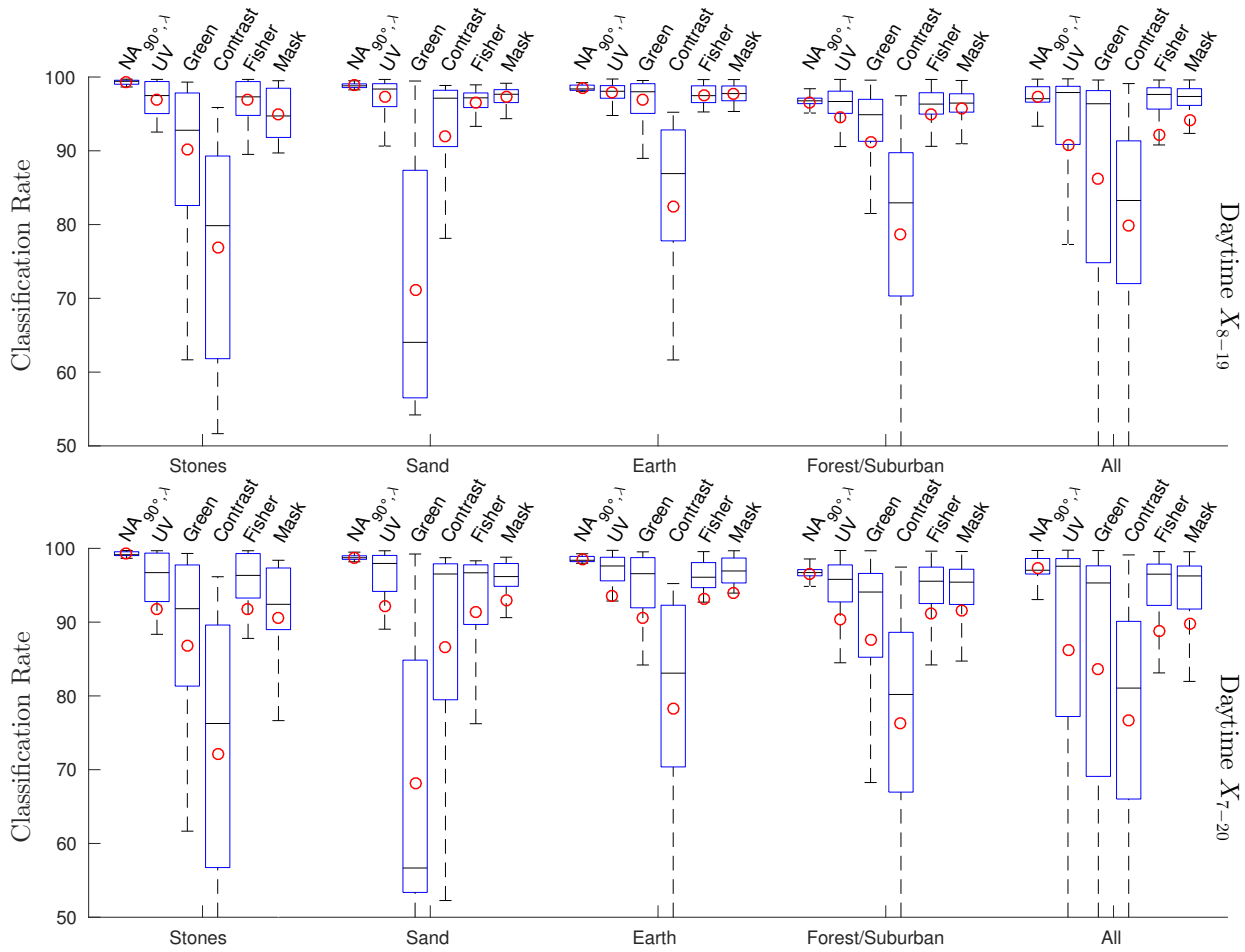
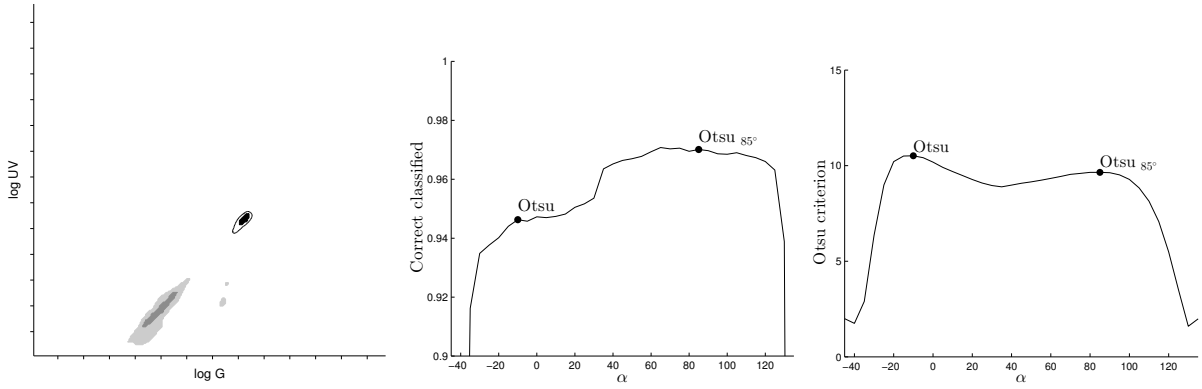


Figure 2.13: Box plots of the classification rates for the local separation technique $\mathrm{NA}_{90°,\alpha}$ and all global separation techniques (UV/Green/Contrast/Fisher/Mask). As for the results presented in table 2.4, the global separation techniques were trained on samples drawn from the specified databases, but in contrast this evaluation uses single HDR image pairs (e.g. as captured by a mobile robot) as test data. The numbers of HDR image pairs classified in each box plot are for X_{8-19} (top) and X_{7-20} (bottom): Stones, sand, earth: $n = 132/156$. Forest/Suburban: $n = 924/1092$. All: $n = 1320/1560$. The box plots show the distribution of the classification rates for all separation techniques and databases. Each plot shows the mean (red circle), median (black line), the 25th and 75th percentiles (blue box), and a coverage of $3\sigma \hat{=} 97.7\%$ (black dashed lines). For better readability, outliers are not shown.

Local	$\mathbf{X_{7\text{-}20}}$				**Ø**	
	Otsu	$\mathbf{NA_\lambda}$	$\mathbf{Otsu_\alpha}$	$\mathbf{NA_{\alpha,\lambda}}$	$\boldsymbol{\alpha}$	$\boldsymbol{\lambda}$
Stones	99.1%	99.5%	99.0%	99.6%	**91.3°**	3.6
Sand	98.9%	99.3%	98.9%	99.3%	**87.5°**	3.6
Earth	98.3%	98.9%	99.0%	99.5%	**88.8°**	3.1
Forest/Suburban	96.5%	98.4%	96.8%	98.6%	**92.5°**	2.0

Table 2.6: The classification rates of the local separators introduced in section 2.2.8 — namely Otsu, NA_λ, Otsu_α and $\mathrm{NA}_{\alpha,\lambda}$ — applied to the datasets shown in figure 2.6. Note that for the latter three separators (which are based on α and/or λ) the result of the best parameter combination found is presented. The last columns show the mean values of these parameters α and λ for all tested methods on both sets, X_{8-19} and X_{7-20}. Since the results for the dataset X_{8-19} are similar, they are not presented. The best separation angle (highlighted) is approximately $\alpha = 90°$, which corresponds to a UV-only separation.



(a) (b) (c)

Figure 2.14: A detailed plot for the local separation technique Otsu applied to a scene captured on the 02. Sept. 2014 at 8:10 (example 2.5). (a) Log UV/G plot corresponding to the recorded sample (see figure 2.5 for visualization notes.). (b) Classification rate for Otsu_α, dependent on the projection plane $\mathbf{w}_\alpha$. (c) Otsu criterion for the best threshold value calculated by Otsu_α. Since the real classification rate (b) is not known to the Otsu algorithm, it chooses the value α for the projection plane $\mathbf{w}_\alpha$ which maximizes the Otsu criterion (c). As can be seen, there are two peaks in (c), however the corresponding classification rates differ markedly. Since the maximum is found at $-10°$, the Otsu algorithm does not perform an optimal separation on the sample. For comparison $\text{Otsu}_{85°}$, which yields the best results over the forest/suburban database, is shown.

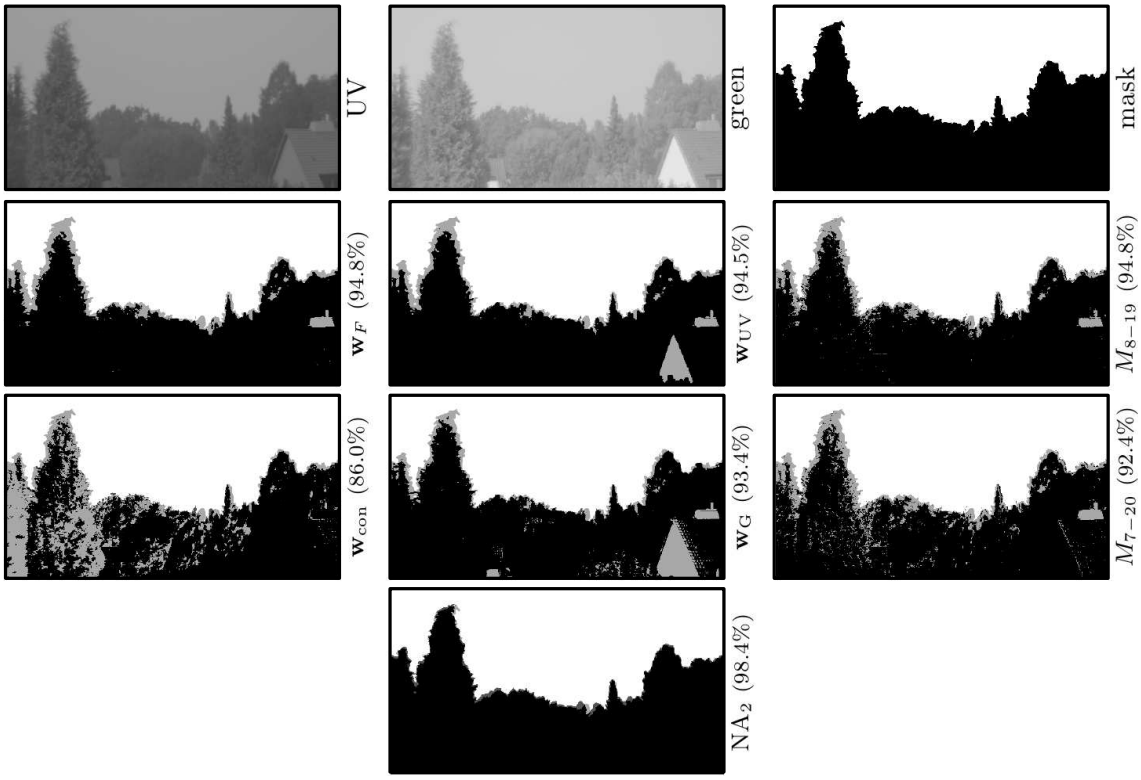
2.3.4 Comparison between Global and Local Separation Techniques

To compare all global separation techniques in this work against each other, the classification rates are calculated for the samples X_{7-20} and X_{8-19} and presented in table 2.4. All global separation techniques show comparable results with classification rates around 85−95% except for the green-only separation and the contrast mechanism $\mathbf{w}_{\text{con}}$ with $85-90\%$ and $75-80\%$, respectively. The additional hours in the morning and evening (sample X_{7-20}) strongly decrease the performance of all global separation techniques. As expected, the best classification rate for the samples X_{7-20} and X_{8-19} are obtained by M_{7-20} and M_{8-19}, respectively. In contrast, the local separation techniques show both an overall better classification rate and a better tolerance to the additional hours for sample X_{7-20} (table 2.6). Especially the fixed parameter methods $\text{NA}_{\alpha,\lambda}$ and Otsu_α outperform all global separators on both test samples and do not degrade on sample X_{7-20}.

Even though most separation techniques have classification rates over 90%, the difference for the classification may be significant. Example 2.4 shows a scene captured at 04. Sept. 2014 at 11:40 which was selected such that different performances of the global separation techniques can be seen. While all global separation techniques except $\mathbf{w}_{\text{con}}$ show comparable results in the separation between vegetation (trees, bushes, etc.) and sky, artificial objects are often misinterpreted as sky patches. The amount to which this happens differs between the several methods as can be seen in example 2.4. In particular $\mathbf{w}_{\text{G}}$, $\mathbf{w}_{\text{UV}}$ and $\mathbf{w}_{\text{con}}$ show several contiguous patches of artificial objects which would be difficult to remove in post-processing steps.

It is more difficult to find differences between the local separation techniques. They show comparable results over the day and differ mainly in the morning and evening under bad lighting conditions. The results of the local separation techniques visualized in example 2.5 for a more challenging scene from the forest/suburban database captured at 02. Sept. 2014 at 8:10 show small differences between the methods. While the separation between vegetation and sky is very sharp with only small errors (best result for $\text{NA}_{100°,2}$), artificial objects appear in the results of the variable local separation methods NA_2 and Otsu. This is due to the effect described in section 2.3.3, when the Otsu criterion does not correspond with the real classification rate (figure 2.14). The additional correction step for the threshold value t in the NA methods reduces misclassification and improves the classification result compared to the results achieved by the Otsu methods.

Example 2.4: Global Separation Techniques

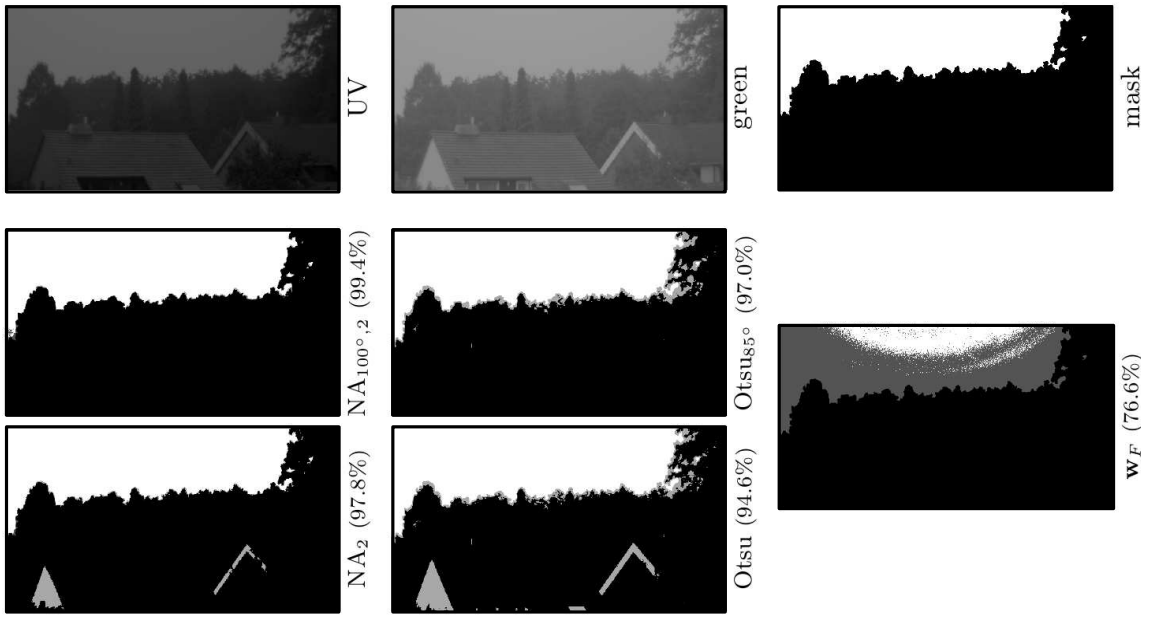


Direct comparison between different global separation methods applied to the same scene from the forest/suburban database on the 04. Sept. 2014 at 11:40. The percentage values show the classification rate of the corresponding separation technique for this single scene (not the average classification rates of table 2.4). Black regions mark correctly labeled ground pixels, white regions correctly labeled sky pixels. Light gray pixels indicate ground pixels which were wrongly classified as sky pixels, dark gray pixels the opposite case. The latter case only occurs for NA_2 in small regions close to the skyline. In the first row the input data (UV- and green-channel images) are shown together with the mask used as ground truth. The last row shows the result of the local separation technique NA_2 for comparison.

2.3.5 Statistical Tests

We use Bootstrapping Efron and Tibshirani (1994) to test whether the mean classification rates of two separation techniques differ significantly (table 2.7). The test set contains the classification rates of the two separation techniques for all HDR image pairs. Bootstrapping now generates a total of 10^4 Bootstrapping sets of the same size as the test set by redrawing from the test set (with repetition allowed). For each Bootstrapping set, the difference between the mean classification rates for the two separation techniques is computed. Over all 10^4 Bootstrapping sets, we obtain a distribution of this difference. For this distribution we test whether zero lies within the confidence interval (two-sided), using significance levels of 99.9%, 99%, and 95%. If this is not the case, the difference of the mean classification rates is significant. The results show that for all separation techniques the classification rates differ significantly, except for the global separators UV, Fisher, and Mask. It can be observed that for the mineral databases (stones, sand, earth), the global separation techniques UV and Fisher do not differ significantly, indicating that the performance of the techniques UV and Fisher are comparable.

Example 2.5: Local Separation Techniques



Direct comparison between different local separation methods applied to the same scene from the forest/suburban database on the 02. Sept. 2014 at 8:10. The notation is the same as described in example 2.4. The most right image shows the result of the global separation technique $\mathbf{w}_F$ for comparison.

		$NA_{90°,\lambda}$	UV	Green	Contrast	Fisher	Mask
Stones	$NA_{90°,\lambda}$		***	***	***	***	***
	UV			***	***	-	-
	Green				***	***	***
	Contrast					***	***
	Fisher						-
	Mask						
Sand	$NA_{90°,\lambda}$		***	***	***	***	***
	UV			***	***	-	***
	Green				***	***	***
	Contrast					***	***
	Fisher						-
	Mask						
Earth	$NA_{90°,\lambda}$		***	***	***	***	***
	UV			***	***	-	**
	Green				***	***	***
	Contrast					***	***
	Fisher						-
	Mask						

Table 2.7: Significance tests were performed on all skyline datasets between 7:00 to 21:00 to compare their classification rates. We used Bootstrapping as statistical test (Efron and Tibshirani, 1994); for details see section 2.3.2. The shown significance values are 99.9% (***), 99% (**), and 95% (*); for the databases *forest/suburban* and *all*, all significance tests are highly significant (99.9%).

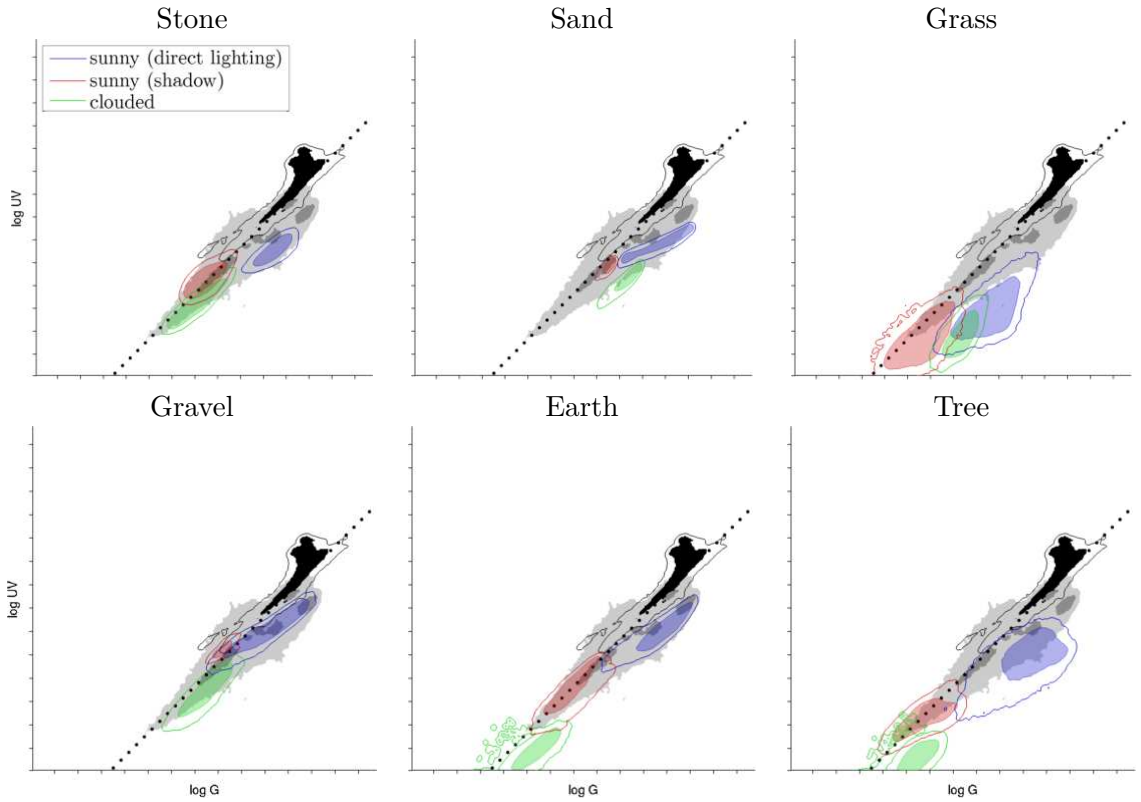


Figure 2.15: A total of 61 different samples of ground objects have been collected. This plot shows 6 different ground objects, each recorded under three different conditions: The object lies in the sun (**blue**) or in the shadow (**red**), both on a sunny day, or the object was recorded on a cloudy day (**green**). For comparison, the pooled data (ground and sky) from figure 2.5 (X_{7-20}) are shown together with the best 1:1 contrast separator ($\mathbf{w}_{\text{con}}$, dotted line). Examples of the log UV/G data used to create the *sand* and *grass* plots are presented in example 2.3. See figure 2.5 for visualization notes.

2.3.6 Records of Ground Objects

The skyline databases cover four specific environments (stones, sand, earth, forest/suburban), however an insect – e.g. an ant navigating through undergrowth – might encounter a wider variety of ground objects. The latter should be covered well by the object databases. To get a visual impression of single ground objects (without sky) under different lighting conditions — compared to the datasets of the mineral skylines collected over the complete days under bright and dry weather conditions — we collected a total of 61 different single samples of ground objects and visualized them together with the data collected over the complete days.

As can clearly be seen from the log UV/G plots of single ground objects (figure 2.15), they differ strongly regarding overall brightness, UV/G contrast, and spread of those values. A strong influence of the overall lighting condition is visible in the data. First, due to the indirect lighting by the blue sky and the high reflectivity for even small wavelengths (UV / near UV, see Sgavetti et al. (2006)), the log UV/G contrast of stones and gravel in shadow is high (a higher amount of UV light compared to green light). In comparison, grass and trees reflect green light around three times better than UV light (Grant et al., 2006) such that the log UV/G contrast is lower. Interestingly, sand was the only mineral which also reflected UV light at a lower rate compared to stones, gravel and earth. Second, ground objects under direct sunlight show a higher proportion of green light compared to UV for all recorded materials. Except for gravel and earth, which show a small overlap with the 1:1 contrast, all other materials are clearly separable from the sky using a linear contrast separator. Third, on cloudy days — where objects were mainly illuminated by

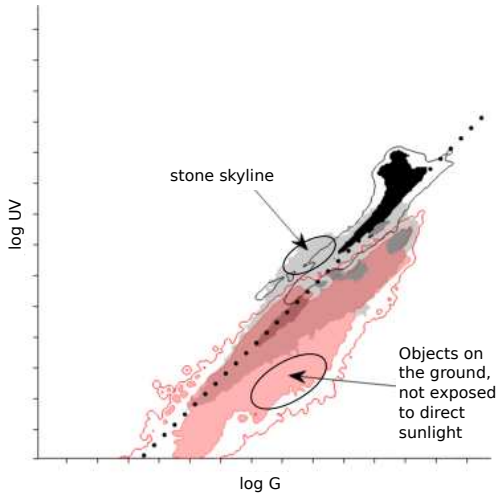


Figure 2.16: Pooled plot of all ground object databases. The wide variety of different objects and lighting conditions (partly in the same image) does not allow a unique distinction of the lighting conditions as in figure 2.15. For comparison, the pooled data from figure 2.5 (X_{8-19}) are shown. The highlighted areas (ellipses) show where differences between the object and skyline databases occur. See figure 2.5 for visualization notes.

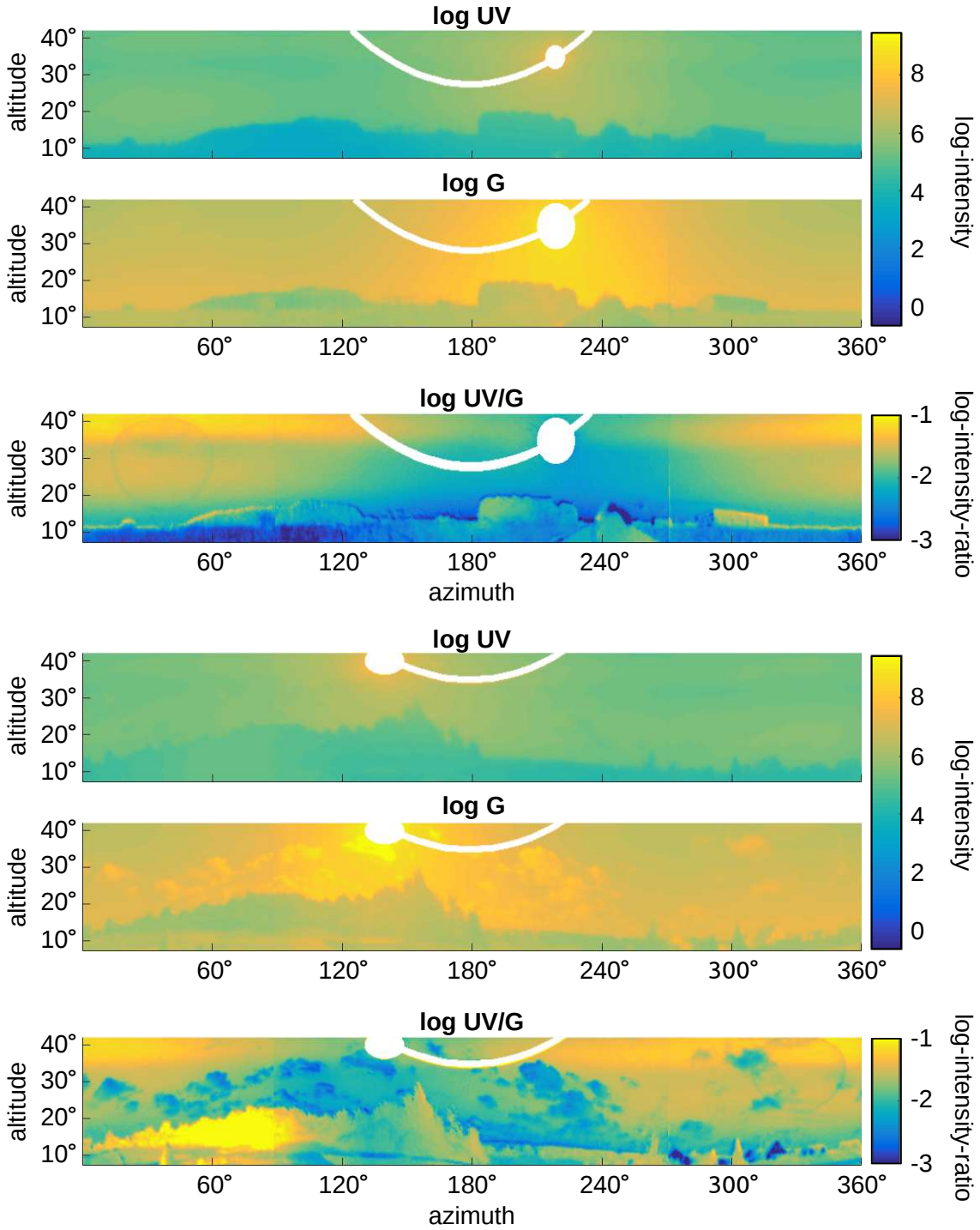
diffuse light reflected from the cloud-covered sky — the collected data show a similar behavior like objects under direct sunlight on sunny days: The data for all materials show a dominance of green light (probably due to a reduction of UV irradiation by the cloud cover), however the overall less strongly lit scenes lead to a strong decrease of the irradiance values on both axes. In summary, diffuse illumination of mineral objects by the blue sky is the most challenging condition for global separators.

To compare the collected log UV/G data from the skyline and object databases, figure 2.16 shows a pooled plot of the complete object databases compared to a pooled plot of the skyline databases. The wide variety of different objects and lighting conditions (partly in the same image) does not allow a unique distinction of the lighting conditions as in figure 2.15, therefore the data of the complete object database is pooled. The log UV/G plot might give the impression that the collected objects have lower UV values compared to the skyline databases. This is due to the overall smaller amount of direct sunlight in most scenes where the object database has been recorded (forest, bushes, etc.) compared to the skyline databases which were mostly recorded in well lit scenes. As can be seen, the log UV/G data of ground objects from the object and the skyline databases intersect mostly except for two areas (section 2.3.6): Under strong illumination, the ground class of the stone skyline database partially extends into the sky class. The ground class of the object database extends into lower log UV/G values (in some cases less strong illumination conditions during the recording, e.g. in forests). However, neither do the log UV/G values of the ground database reach or exceed those of the objects from the stone skyline, nor do we observe an increased overlap of the ground database with the sky class of the skyline databases. Therefore it can be assumed that global separation techniques exhibit the same performance on the ground database as in the collected skyline databases.

2.3.7 Panoramic Images

Example 2.6 shows panoramic images captured on sunny days in the vicinity of Bielefeld university. They show the raw (non-normalized) log UV and log G values obtained by the HDR-algorithm and their ratio $\log \frac{\mathrm{UV}}{\mathrm{G}} = \log \mathrm{UV} - \log \mathrm{G}$, denoted by log UV/G. The blue sky (example 2.6, top) shows the gradient of UV and green light over the sky (Coemans et al., 1994, Rossel and Wehner, 1984), while the clouded sky (example 2.6, bottom) shows the strong influence of clouds to the log UV/G data. Unfortunately for global separation techniques, also the log UV/G contrast

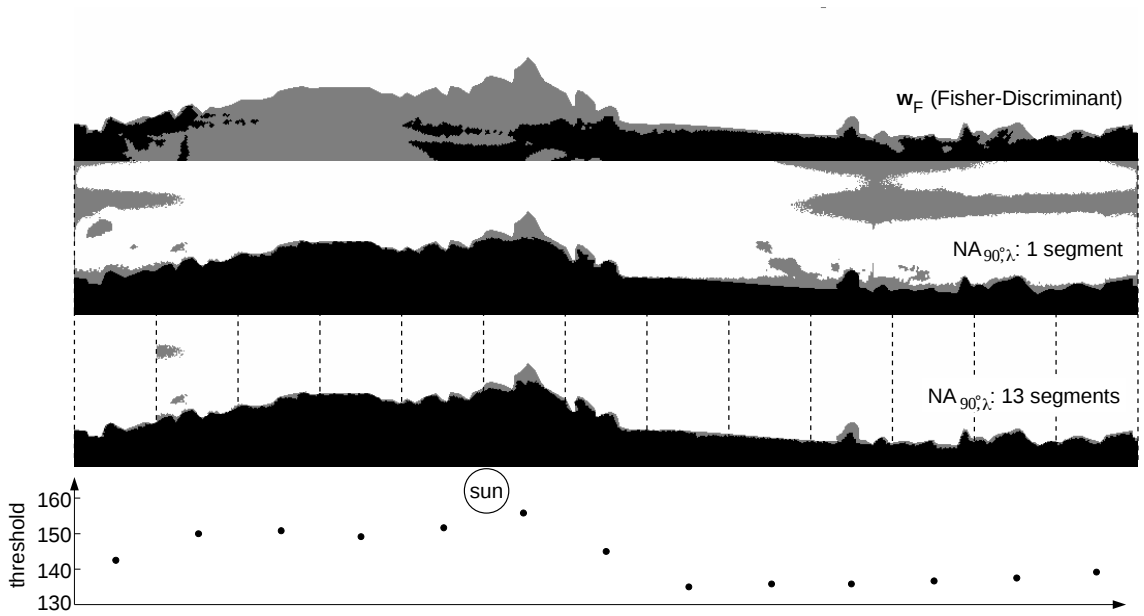
Example 2.6: Panoramic Images



Panoramic images captured on sunny days with ground objects in front of clear blue sky (top: 12. May 2016) and clouded sky (bottom: 27. June 2016) in the vicinity of Bielefeld university. White areas are corrupted by direct sunlight shining onto the sensor and are masked out. Note that while there are many clouds (bottom), the sun itself is *not* covered by clouds. Moreover, clouds which are illuminated from behind appear very bright in the green channel, but are (nearly) not visible in the UV channel.

differs strongly between data points close to the sun compared to data points opposite to the sun, increasing the difficulty of applying a global threshold. The experimental setup used to capture the skyline databases was always pointing into the same direction (mostly north), however, due to the movement of the sun, the appearance of the sky differs strongly over the day. This has nearly no effect on local separation techniques since they can adapt their their thresholds for each image individually, however global separation techniques cannot adapt to this directionality.

Example 2.7: Comparison of Separation Techniques on Panoramic Images



Example of a skyline extraction from the panoramic log UV/G image shown in example 2.6 (bottom) using the separation techniques $\mathbf{w}_F$ (global) and $NA_{90°,\lambda}$. Pixel classified as sky and ground are colored white and black, respectively, while gray pixel indicate misclassifications. The gradient of the log UV and log G data over the sky shown in example 2.6 increases the difficulty for local separation techniques to find a threshold value to separate both classes on the whole image. By splitting the panoramic image into smaller images (bottom, here 30° steps), this problem can be avoided. The graph shows the threshold values for each individual segment. On the contrary, global separation techniques cannot be individually adapted for single segments. The classification rates from top to bottom are 82.1%, 87.4%, and 98.0%.

Example 2.7 shows how the best global separation technique $\mathbf{w}_F$ and the local separation technique $NA_{\alpha,\lambda}$ (with a projection angle $\alpha = 90°$ for UV-only separation) compare on the panoramic image with direction-dependent lighting conditions shown in example 2.6. As can be seen, the global separation technique (classification rate: 82.1%) misclassifies regions which are brightly lit by direct sunlight, while the local separation technique (classification rate: 87.4%) misclassifies large regions of the sky which are opposite of the sun. For the latter, this problem can be avoided by using different thresholds for different parts of the panoramic image. Here we calculated the threshold values for a total of 13 segments (each spanning an azimuthal angle of around 28°) which increases the classification rate to 98.0%. The number of segments has been chosen as a trade-off between too few segments (no adaption to differing lighting conditions) and too many segments (insufficient segment sizes might lead to instable results from Otsu's method). The skyline extraction could furthermore be smoothed by linearizing the threshold values between neighboring segments resulting in an individual threshold for each column.

The classification rates for in total 12 recorded panoramic images can be found in table 2.8.

Separation-technique	Sunlight conditions												Average	
	Covered (C)						Direct (D)						C	D
$\mathbf{w}_F$	99%	97%	98%	97%	95%	67%	97%	82%	81%	79%	78%	82%	92%	83%
$NA_{\alpha,\lambda}$, 1 Seg.	99%	91%	98%	98%	94%	82%	99%	87%	72%	91%	85%	87%	94%	87%
$NA_{\alpha,\lambda}$, 13 Seg.	99%	97%	97%	97%	98%	92%	98%	98%	99%	96%	97%	97%	97%	97%

Table 2.8: We recorded a total of 12 panoramic images and divided them into two groups; one group where direct sunlight is shining onto the sensor (e.g. example 2.6) and one where the sun is covered by clouds. The classification rates for the separation techniques tested on each panoramic image as well as the average classification rate for each group is shown.

We divided them into two groups; one group where direct sunlight is shining onto the sensor (e.g. example 2.6) and one where the sun is covered by clouds. As can be seen, the classification rates are stable for the tested separation techniques as long as the sun is covered by clouds. However, if direct sunlight is shining onto the sensor the classification rates drop for the global separation technique $\mathbf{w}_F$ from 92% to 83% and for the local separation technique $NA_{\alpha,\lambda}$ (without slicing the panoramic image into multiple segments) from 94% to 87%. In contrast, by calculating an individual threshold for multiple segments (here: 13), the classification rate of $NA_{\alpha,\lambda}$ stays stable at around 97%.

2.4 Discussion

In this chapter we explore the question of how insects achieve an illumination-invariant representation of a skyline. The skyline could then be used as input to navigation models. Navigation models which use a skyline representation (Basten and Mallot, 2010, Schwarz et al., 2014) assume that the skyline can easily be extracted by the insect, however only few studies actually suggest methods how to extract the skyline from the visual input. We present several methods for an illumination-invariant skyline extraction — which are based on four different hypotheses — and examine them for two different properties in particular: First, we evaluate the classification rate (section 2.2.7) which is a measure for correctly classified image points. Second, we examine if those methods that need to be trained on a set of training data (i.e. global separation techniques) generalize well to new data which were not part of the training data (appendix B.2).

2.4.1 Skyline Extraction

The hypothesis by Möller (2002) states that log UV/G contrast data can be used to obtain an illumination-invariant separation between sky and ground patches in images. Specifically, Möller (2002) suggests — based on a collection of data in two spectral channels — that insects may use separation mechanisms which are based on (I) fixed thresholds (global separation techniques[2]) on the log UV/G data (dual channel), or (II) on variable thresholds (local separation techniques[3]) on the UV-only data (single channel).

Here we explore these two hypotheses (I) and (II). Additionally, we test global separation techniques on UV-only data (III) and local separation techniques on log UV/G data (IV), such that we are able to compare a total of four possible combinations (table 2.9). We do not list green-only separation explicitly in table 2.9 as single-channel separation, since we found that in our tests green-only separation is always inferior to UV-only separation. All separation techniques presented in this work are based on linear separators or on masks. More sophisticated separation techniques, e.g. support vector machines or quadratic separators, are not tested due to their increased complexity and computational expensiveness. In addition, even complex global separation techniques cannot perform better than the binary masks we tested, which are constructed such that the maximal classification rate for the training data is obtained (section 2.2.8.4).

[2]Supervised learning of separators, trained on a large data set with manually defined skyline masks (section 2.2.8).

[3]Unsupervised separation trained on the given image (section 2.2.8).

	global	local
UV-only (single channel)	(III) $\mathbf{w}_{\mathrm{UV}}$	(II) $\mathrm{NA}_{90°,\lambda}$,$\mathrm{Otsu}_{90°}$
UV/G (dual channel)	(I) $\mathbf{w}_{\mathrm{F}}$, $\mathbf{w}_{\mathrm{con}}$ M_{8-19}, M_{7-20}	(IV) NA_2,Otsu

Table 2.9: Overview over the four different hypotheses (I)-(IV) tested in this work together with their corresponding separation techniques. Even though $\mathbf{w}_{\mathrm{G}}$ would fit to Hypothesis (I), it is not listed since it is a green-only separator.

global separator	class. rate	generaliz.	dusk/dawn tolerance
$\mathbf{w}_{\mathrm{UV}}$	+	○	+
$\mathbf{w}_{\mathrm{G}}$	○	+	○
$\mathbf{w}_{\mathrm{F}}$	+	−	+
$\mathbf{w}_{\mathrm{con}}$	−	+	−
M_{8-19}, M_{7-20}	+	+	+

Table 2.10: Overview of the different global separation techniques tested in this work regarding the classification rate (table 2.4), the generalization to new data (table 2.5) and the tolerance to samples including dusk and dawn (figure 2.10). A coarse rating $(+, ○, -)$ is given to simplify the comparison between different global separation techniques.

2.4.1.1 Global Separation Techniques

We tested different global separation techniques for their ability to distinguish between the terrestrial and celestial parts of images (table 2.4). Furthermore, we examined the generalization of the global separation techniques to new data, which were not part of the training data (table 2.5). Table 2.10 shows a brief overview of the different properties of the global separation techniques tested in this work. The data collected by Möller (2002) and Kollmeier et al. (2007) suggest that a contrast mechanism between the UV and green channel could be exploited to perform this separation. We implement the contrast measure $a \log UV - b \log G$ with $a, b \in \mathbb{R}$ in two variations: By choosing $a = b = 1$, a strict contrast mechanism can be applied to the collected data ($\mathbf{w}_{\mathrm{con}}$). Compared to a classification rate of 50%, which corresponds to random classification, the contrast measure achieves a classification rate slightly over 75%. While this classification rate may convey the feeling that a skyline separation with this strict contrast mechanism could be obtained, visual inspection (e.g. example 2.4) shows that the skyline cannot be extracted reliably. Contrary, the implementation of a less strict contrast measure, where a, b are optimized for the Fisher criterion ($\mathbf{w}_{\mathrm{F}}$) as suggested by Kollmeier et al. (2007), results in the best classification rates achieved by all global linear separators tested in this work. However, $\mathbf{w}_{\mathrm{F}}$ depends strongly on the training data and shows the worst results of all tested techniques with respect to the generalization to unknown data. The best performance was achieved by the non-linear contrast mechanisms which were realized by training a binary mask (M_{7-20}, M_{8-19}). As expected, the masks show the best classification rates with around 90% at times between 8:00-20:00. Furthermore, the masks generalize well to new data sets. While the binary log UV/G masks show the best performance along all global separation techniques, they only perform slightly better than UV-only separation ($\mathbf{w}_{\mathrm{UV}}$). However, the generalization of UV-only separation to new data sets is only mediocre. Since the classification rate for green-only separation $\mathbf{w}_{\mathrm{G}}$ cannot compete with the classification rates of M_{7-20}, M_{8-19} and $\mathbf{w}_{\mathrm{UV}}$, it is not of further interest.

As expected, the results show that the mineral skyline databases have a higher portion in the UV range compared to the forest/suburban skyline database. In brightly lit scenes (e.g. afternoons

on sunny days) the log UV/G values between minerals (except for the stone skyline) and sky differ noticeably and allow a coarse classification by a linear separator (figure 2.6). While this is the same as for the forest/suburban database, for indirectly lighted minerals (objects in shadow, clouds in front of the sun) the classification becomes more difficult. Due to the higher UV reflectivity, ground objects in the mineral skyline databases are generally closer to the sky data. Furthermore, the increased UV intensity (compared to the green light) caused by the indirect lighting of the sky leads to a high amount of data points which have a log UV/G contrast where the UV portion dominates. Especially for the stone skyline, a linear 1:1 contrast separator cannot be used anymore. Instead, the Fisher discriminant ($\mathbf{w}_F$ in figures 2.5, 2.6 and table 2.4) shows that for the stone skyline a best global separation can be achieved for UV-only data. This result is contrary to the hypothesis that a global linear separation can be achieved by using a log UV/G contrast.

Interestingly, we found that under specific circumstances the quality of classification for the sand skyline can be increased noticeably by using log UV/G contrast: As can be seen in table 2.4, the classification rates are increased by $\approx 6\%$ compared to UV-only separation and even more by $\approx 30\%$ compared to G-only separation. The achieved classification rates were only $\approx 2\%$ worse compared to the binary masks, which represent the best global separation. However, the increased classification rates depend strongly on the weather conditions. As shown in figure 2.9, the advantage of using log UV/G data is only prominent if the (dry) sand skyline is visible in front of a cloudy/rainy sky. If instead the sky of a sunny day is used, the UV-only and UV/G contrast separations achieve about the same classification rates.

The classification rate of all global separation techniques decreases if the test sample contains data collected under dim light conditions (sample X_{7-20}). Around 7:00 and 21:00 they drop rapidly to 50% (figure 2.10), which is equivalent to random classification. Overall the masks show the best performance and generalization for all tested global separation techniques, such that hypothesis (I) should be preferred over hypothesis (III). However, we cannot exclude hypothesis (III) as a possible insect mechanism.

2.4.1.2 Local Separation Techniques

Compared to the global separation techniques, all local separation techniques exhibit a better performance on both samples, especially on X_{7-20} which contains data collected under dim light conditions. The worst performance is achieved by those methods without threshold correction, i.e. Otsu and Otsu_α, which both have a classification rate around 96%. The methods with threshold correction, i.e. NA_λ and $\text{NA}_{\alpha,\lambda}$, achieve classification rates over 99%. Interestingly, the performance of the local separators is not affected by the additional data collected under dim light conditions in the morning and evening. Regarding only the classification rate, $\text{NA}_{\alpha,\lambda}$ is the best separation technique tested in this work. Since all methods with fixed projection angles α show the best performance with angles close to 90°, a similar result can be obtained by a local UV-only separation. Therefore hypothesis (IV) could be rejected, such that only hypothesis (II) remains as a likely hypothesis for skyline extraction.

As discussed in section 2.3.1, we have found an increased UV portion for stones, sand, and earth objects in the mineral skyline databases compared to the suburban/forest skyline database. For global separation techniques, the increased UV portion lessens the quality of classification between ground objects and the sky. However, for local separation techniques this effect could not be observed, the classification rates of the local separation techniques are around $96\% - 99\%$ for all databases and tested techniques. Moreover, best classification rates could be obtained by using UV-only separation on all tested databases. This confirms the idea proposed by Wehner (1982) (pp. 96) that a classification between skyline and ground can be obtained by using the UV channel only.

2.4.2 Panoramic Images

As shown by Stone et al. (2014), local UV-only separation can be used to extract the skyline from panoramic images in real time in robot applications using similar computationally-inexpensive

methods (e.g., watershed algorithm). They demonstrate that an agent can use this segmented skyline information to localize itself on a previously-driven track. In section 4, this idea is enhanced by adding rotational invariance using spherical harmonics such that even aggressively-maneuvering robots are able to localize themselves on a previously-driven track. Using the adaption to omnidirectional images by using multiple segments, the quality of the extracted skyline can be increased.

2.4.3 Color Contrast Mechanisms in Insects

In this work we only examine UV/G dual-channel vision as it may be used by the desert ant *Cataglyphis bicolor* (Mote and Wehner, 1980). A significant difference between the vision of insects and the data collected in this work is the way that light is perceived. It is assumed that insects are able to adapt to changing light intensities (section 2.1.3). As a consequence insects are most likely not able to pin down the perceived light intensity directly on a global scale. However, to obtain a database which includes a wide range of light intensities we collected the pixel-wise total light intensity on a global scale (log values).

We can assume that UV and green receptors do not adapt *individually*, as that would eliminate the spectral characteristics. Linear global separation methods would still be applicable if UV and green receptors would *simultaneously* adapt in a way that the projection on the hyperplane $\mathbf{w}_\alpha$ remains unchanged. Unfortunately, to our knowledge there is no information available about simultaneous adaption of receptor cells with different spectral sensitivity in the same ommatidium.

While a linear separation is a simple operation in neural networks, we are doubtful whether binary masks could actually be used by insects. The lookup (of absolute light intensities) would have to be accomplished by the UV and green receptors in each ommatidium, which would require a rather complex classification network. Thus the separation with binary masks should be understood as an exploration of the theoretical limits of the classification rates of global separators.

It is known from behavioral experiments that insects (especially ants) can be trained to use differently colored landmarks (Aksoy and Camlitepe, 2014, Camlitepe and Aksoy, 2010). This suggests that insects can not only see different colors, but are also able to distinguish between them for navigational purposes. Furthermore, insects are still able to navigate under unnatural lighting conditions, for example in artificial training areas. This stands in conflict to hypotheses (II) and (III) which depend on UV-only separation which would fail if the UV portion of the light is strongly reduced. Only mechanisms based on hypothesis (IV) (i.e. Otsu or NA) are able to compensate for strong variances of the UV- and green-portion in the light. Even complete loss of information in one color channel (UV-light or green-light) can be compensated, since the projection plane of the log UV/G diagram can be chosen arbitrarily.

2.5 Future Work

A next step could be to study skyline separation for a combination of the UV channel with a second channel with longer wavelength (such as red or near infrared). According to Kollmeier et al. (2007) such combinations promise improved skyline separation quality in technical applications, specifically robot navigation. However, they are, of course, no longer a model of hymenopteran vision. Furthermore, it could be examined if color boundary detection (Yang et al., 2013), used with the color channels UV and green, improves the illumination-invariant detection of edges compared to a combination of colors from the visible light range.

2.6 Conclusion

We analyzed four different hypotheses for separation methods which can be used to separate terrestrial and celestial parts of images (or equivalently: detect the skyline). These hypotheses were based on all possible combinations of the following two questions (table 2.9): (a) 'Does UV/G contrast vision increase the classification rate of a separation method compared to UV-only vision?' and (b) 'What yields a better classification rate: Separation methods based on a fixed threshold (global separation techniques) or separation methods which adapt the threshold

dependent on the input image (local separation techniques)?'

Regarding question (a), we found that the classification rates of global separation techniques are only marginally increased for UV/G contrast vision in comparison to UV-only vision. Local separation techniques showed best performance for UV-only separation. Furthermore, we could show that the mineral skylines have different log UV/G irradiance characteristics than a forest skyline increasing the difficulty of a global classification based on the log UV/G data. The stone skyline database showed a high irradiance in the UV range such that a distinction from the sky is generally difficult. The same holds for the earth skyline database, however the irradiance in the UV range was less strong compared to the stones. Only for the special case of dry sand in front of a cloudy and/or rainy sky, the classification rate for global linear separators could be improved noticeably by using log UV/G data instead of log UV data only.

For question (b), we found that the difference between local and global separation techniques is pronounced: All local separation techniques yield a better classification rate than global separation techniques for all performed tests. The performance gain increases the classification rates of pixels which are close to the skyline in particular such that the shape of the skyline itself yields sufficient visual information which could be used as landmarks. Independent of the type of skyline (minerals or plants) the best classification rates could be achieved by using the log UV data only. Moreover, we could show that local separation techniques can be easily adapted to work with panoramic images by splitting them into multiple segments. This makes local separation techniques a promising tool for skyline extraction in autonomous navigation.

However, the ability of insects to adapt to strong illumination changes and changes of the light spectrum (e.g. ants in artificial arenas or shaded places like forests) supports the assumption that insects use information from both the UV and green channel. A method which is capable of both adaption to strong changes in the light spectrum and a robust skyline separation is given by the proposed $NA_{\alpha,\lambda}$ method. This method adapts the UV/G contrast vision automatically such that the contrast is maximized regarding the Otsu criterion. Furthermore, $NA_{\alpha,\lambda}$ is simple and we can assume that an appropriate (and for insects plausible) neural network exists.

CHAPTER 3

Spherical Harmonics: Theory & Software Implementation

In this chapter we introduce the basics of Fourier analysis on the rotation group. This generalization of the Fourier transform can be used to perform computations on functions defined on the rotation group or on functions defined on the unit sphere in the frequency domain. We combine many sources from different research areas — including mathematics, physics, and informatics — with the aim of providing the theories and formulas necessary for a computational implementation. A special use-case is the derivation of the Fourier transform of real-valued functions defined on the unit sphere as they can for example be used to describe panoramic images. These functions can be represented using the basis of real spherical harmonics in the frequency domain. Using the example of an agent equipped with a panoramic imaging device, we introduce several techniques required for navigation experiments in ongoing chapters. Besides enhancing some of the existing theory for complex-valued functions to work with real-valued functions, we derive sparsity relations for rotations, formulas to approximate translations in the basis of real spherical harmonics, and show how symmetries and weighting functions can be used to work with hemispherical camera input only. These findings are used in subsequent chapters to define a method to localize robots using the amplitude and bispectrum, a visual 3D compass to rotationally align panoramic images, and a homing method to determine the relative pose between two panoramic images. The theory derived in this chapter has been implemented in a C++ library.

3.1 Introduction

In this section we introduce the basic tools needed to perform computations in the frequency domain on the unit sphere $S^2 = \{\vec{x} \in \mathbb{R}^3 \mid \|\vec{x}\| = 1\}$. Here, the superscript of S^2 denotes the dimension of the spheres surface (i.e. the dimension of its manifold); analogously to the unit sphere, the unit circle is $S^1 = \{\vec{x} \in \mathbb{R}^2 \mid \|\vec{x}\| = 1\}$. At first, we introduce a generalized Fourier transform for functions defined on the rotation group $SO(3)$ using Wigner-D matrices. Afterwards, we show how spherical harmonics (SH) and real spherical harmonics (RSH) — bases for complex- and real-valued functions, respectively, defined on the unit sphere S^2 — can be derived from the Wigner-D matrices. Moreover, the standard Fourier transform naturally extends to functions defined on the unit sphere S^2 which allows us to perform computations on S^2 in the frequency domain. An important point of this work is to bring together the theoretical basics of Fourier analysis as well as the necessary implementation details from many different sources. We extensively use the derived theories in the implementation of our localization algorithm (section 4), visual 3D compass (section 5), and 3D-warping (section 6). In our experiments we only use real-valued functions defined on the unit sphere (i.e. we only work with RSH) to represent omnidirectional camera input, however the more general framework presented in this work allows to transfer the theory also to related problems. An outline of the theory is given in figure 3.1.

SH are used in a wide range of fields, including registration of 3D models (Burel and Henoco, 1995, Shen et al., 2009) and point clouds (Makadia et al., 2006, Dillenseger et al., 2006), visual robot navigation (Friedrich et al., 2008), approximation of diffuse lighting in computer graphics (Sloan et al., 2002), computation of gravitational fields in geodesy (Kaula, 1966) and Gaussian

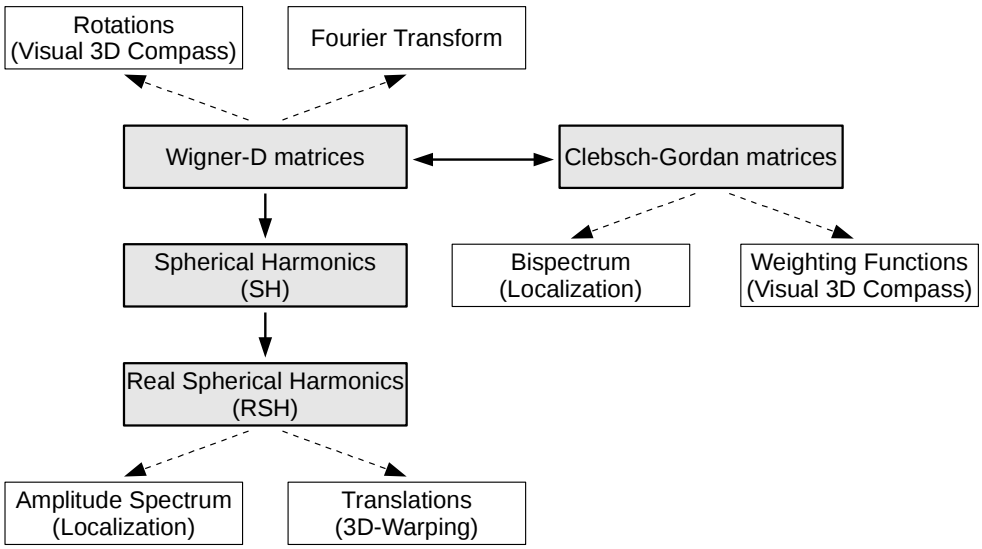


Figure 3.1: The key components of the underlying theory (gray blocks) and their practical applications in robotics (white blocks) used in this chapter are shown. With the Wigner-D matrices, we obtain a tool to Fourier transform functions defined on the rotation group $SO(3)$ (rotations in 3D space). Moreover, Wigner-D matrices can be used to efficiently rotate functions defined on $SO(3)$. A special case of the Wigner-D matrices are the spherical harmonics — and its real-valued analogue — which are a valuable tool to Fourier transform functions defined on the unit sphere. We furthermore show how the amplitude spectrum and translations can be calculated directly in the basis of spherical harmonics. Finally, the Clebsch-Gordan matrices allow us to define point-wise products between spherical harmonics. These point-wise products are used to define the bispectrum and weighting functions in the basis of spherical harmonics.

fields in astronomy (Marinucci and Piccioni, 2004), and wave propagation (Ishimaru, 1991).

The theory in this chapter is mainly based on Marinucci and Peccati (2011) and Chen et al. (2002). Note that we do not describe the underlying (algebraic) theory of harmonic analysis. Instead, we directly use the well-studied application of harmonic analysis for the rotation group $SO(3)$. For an overview of related fields and background information on the algebraic theory of harmonic analysis, we recommend the following sources: Topological and algebraic topics are covered in Hewitt and Ross (1963) and Kakarala (1992). Detailed information on groups and Lie groups are available in Chirikjian and Kyatkin (2001). Information on representation theory and Lie algebras can be found in Erdmann and Wildon (2006).

Besides giving the reader a straight-forward tutorial on how to implement SH, the contributions of this work are the derivation of sparsity relations for real Wigner-D matrices around single axes (theorems 3.23-3.26), the reformulation of Clebsch-Gordan matrices and the bispectrum for RSH (rather than for SH, section 3.9.4), the use of symmetries and weighting functions for RSH to work with hemispherical and non-hemispherical inputs (section 3.6.2 and 3.10.3), the derivation of formulas to approximate translations in the basis of RSH (section 3.8), and details to efficiently implement the theory derived in this chapter (section 3.10). For well-known theorems, references to their proofs are provided, all other proofs were done by the author and are, if not stated otherwise, to the best knowledge of the author novel. Finally, we built a C++ library based on the theory discussed in this chapter (section 3.10); an early prototype of this library was used in Stone et al. (2016).

3.2 Motivation

The RSH form an orthonormal basis for functions defined on the unit sphere. By performing a basis change (Fourier transform), a function f can be represented in the frequency domain instead of the spatial domain. However, since the function f can be represented equivalently in both the

frequency and spatial domain, the question may arise which benefits are obtained by changing into the basis of RSH.

As for the standard Fourier transform — which uses the family e^{ikx} with $k \in \mathbb{N}$ as a basis — the RSH have some essential properties (section 3.5.3). Most importantly, by using the basis of RSH to represent a function f we can easily calculate its amplitude spectrum (section 3.9.2). Moreover, convolutions with other functions — in the spatial domain a costly operation — can be computed more efficiently (Basri and Jacobs, 2003).

For this work, we are especially interested in the relation between RSH and the unit sphere: Panoramic images are used throughout this work to represent the visual scene around a robot, e.g. as captured by a full-spherical panoramic camera. A major drawback of panoramic images is that strong distortion effects appear by mapping the surface of the unit sphere to a plane. These distortions need to be taken care of, for example if integrals over panoramic images are calculated (e.g. image comparisons, section 3.9.1). In contrast, the basis of RSH can be used to represent functions defined on the unit sphere without suffering from distortions. Moreover, during the Fourier transform we can oversample a panoramic image to reduce aliasing artifacts (section 3.10.1).

Another important property of the RSH is their beneficial behavior under rotations. Rotating a function in the basis of RSH can be realized via a sparse matrix-vector multiplication (section 3.7). In contrast, to rotate a panoramic image in the spatial domain, each pixel in the image needs to be shifted — accordingly to the rotation — to a new position in the rotated panoramic image. Since the pixels commonly do not move in integer steps, each pixel of the rotated panoramic image needs to be interpolated. Besides the overhead necessary for interpolation, the quality of the rotated panoramic image degrades with each subsequent rotation. Especially for applications with numerous subsequent rotations (e.g. the visual 3D compass, chapter 5) the quality loss is substantial without intermediate resampling from the original image (example 3.1).

3.3 Rotation Group SO(3)

A rotation in the Euclidean space $\mathbb{R}^n$ can be expressed using matrix transformations which preserve volume and relative orientation (up to reflections) of the basis elements of the vector space. Rotation matrices $\mathbf{R} \in \mathrm{Mat}(n, \mathbb{R})$ have to be orthogonal, i.e. $\langle \mathbf{R}x, \mathbf{R}y \rangle = \langle x, y \rangle$ for all $x, y \in \mathbb{R}^n \setminus \{\vec{0}\}$, and are called **rotation matrices**. A detailed description of rotation matrices together with different rotation parameterizations can be found in Goldstein et al. (2012), chapter 4. The set of the rotation matrices $\mathbf{R} \in \mathrm{Mat}(n, \mathbb{R})$ is denoted by $O(3)$ and called the **orthogonal group**. Note that from the orthogonality of rotation matrices it follows directly that $\mathbf{R}^{-1} = \mathbf{R}^T$ and $\det(\mathbf{R}) = \pm 1$. Rotations can be split into two different sets: Proper rotations $SO(n)$ with $\det(\mathbf{R}) = 1$ and improper rotations $O(n) \setminus SO(n)$ with $\det(\mathbf{R}) = -1$, respectively. In contrast to proper rotations, an improper rotation also performs a reflection. In this work we are especially interested in the proper rotation matrices in $\mathbb{R}^3$ which form the **special orthogonal group** $SO(3)$.

3.3.1 Elementary Rotation Matrices

Rotations around the basis vectors $\vec{X}$, $\vec{Y}$ and $\vec{Z}$ can be expressed by **basic rotation matrices** as

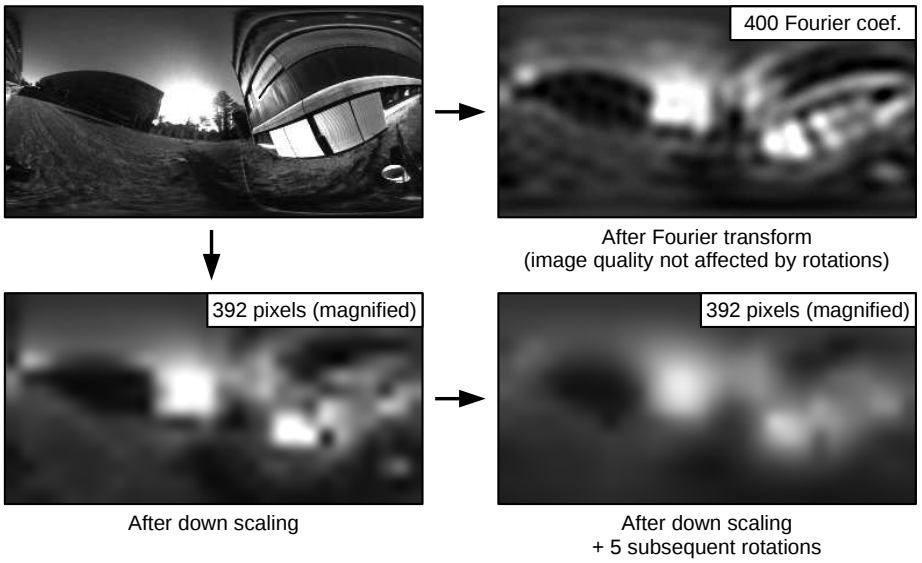
$$\mathbf{R}_{\vec{X},\alpha} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}, \mathbf{R}_{\vec{Y},\alpha} = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix}, \mathbf{R}_{\vec{Z},\alpha} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.1}$$

Each of these basic rotations is an element of $SO(3)$, and it can be shown that, by combining the basic rotation matrices, any rotation matrix $\mathbf{R} \in SO(3)$ can be constructed, i.e. there exist α, β, γ such that

$$\mathbf{R} = \mathbf{R}_{\vec{X},\gamma} \mathbf{R}_{\vec{Y},\beta} \mathbf{R}_{\vec{Z},\alpha}. \tag{3.2}$$

This concatenation of rotations around fixed axes is called a **rotation parameterization** and is denoted by the involved axes of the basic rotations, in this case XYZ. The corresponding rotation

Example 3.1: Spatial versus Frequency Domain



The panoramic image shown with $28 \times 14 = 392$ pixels is represented in the basis of RSH using $L = 20$ bands (frequencies); the resulting Fourier coefficient vector has 400 entries. As can be seen, the panoramic image represented in the basis of RSH appears more detailed than the down scaled image. During the compass search of the visual 3D compass (chapter 5) we apply subsequent rotations to the panoramic image. While these rotations have no influence on the function represented in the basis of RSH (frequency domain), the quality of the panoramic image (spatial domain) is already after 5 rotations noticeably decreased. This is due to the necessity to interpolate the rotated panoramic image after each rotation; alternatively the rotated image can be resampled from the original image which is a computational expensive task. For the subsequent rotations we used bilinear interpolation since it offers a good trade-off between interpolation quality and computational costs. For down scaling and magnification of the panoramic image we used cubic interpolation.

angles are written as tuple (γ, β, α). Rotation parameterizations around three different axes are called **Tait-Bryan** rotations. However it is also possible to express arbitrary rotations using only two axes, e.g.

$$\mathbf{R} = \mathbf{R}_{\vec{Z},\gamma}\mathbf{R}_{\vec{Y},\beta}\mathbf{R}_{\vec{Z},\alpha} \tag{3.3}$$

or short ZYZ. Rotation parallelizations of this type are called **Euler** rotations. Other axis combinations than Tait-Bryan and Euler rotations are possible. In this work we use both Tait-Bryan (XYZ) and Euler rotations (ZYZ). While the first rotation parameterization is more intuitive, the latter is often used in mathematical derivations and formulas.

We need an additional third rotation type in order to deal with metrics defined on $SO(3)$: Any rotation can be represented by a rotation matrix $\mathbf{R}_{\vec{v},\alpha}$ with a (normalized and non-zero) rotation axis $\vec{v}$ and rotation angle α, called the **axis-angle** rotation parameterization.

3.3.2 Tilt Matrices

In the later chapters 4, 5, and 6, we examine visual navigation strategies which have to cope with tilted robots. For example, a wheeled robot moving on a plane would be tilted if one wheel is lifted from the ground (e.g. by moving over a carpet). In contrast to a rotation $\mathbf{R}_{\vec{v},\alpha}$, where the rotation axis $\vec{v}^T = (x, y, z)^T$ can be arbitrary, we refer to a rotation matrix as **tilt matrix** if its rotation axis is limited to the case $z = 0$.

In the following, we parameterize tilt matrices by a tuple (α, β), where the angle α specifies

the tilt angle and β the tilt direction:

$$\mathbf{R}_{(\alpha,\beta)} := \mathbf{R}_{\vec{v},\alpha} \quad \text{with} \quad \vec{v} = \mathbf{R}_{Y,\beta}(0,1,0)^T \tag{3.4}$$

Regarding the coordinate system from example 3.4, the tilt matrix of a robot driving down a ramp with a slope of 5° would be given by $\mathbf{R}_{(5°,0°)}$. Analogously, the tilt matrix of a two wheeled robot, whose right wheel is lifted by 5° from the floor, would be given by $\mathbf{R}_{(5°,90°)}$.

3.3.3 Distance Measure for Rotation Matrices

To compare two 3D rotations we need to define a norm on the rotation group $SO(3)$ to obtain a **rotational distance measure**. While distances of vectors in an Euclidean space are commonly intuitive and vivid (e.g. the Euclidean norm or absolute norm), this is not the case for the rotation group $SO(3)$. Several suggestions have been made to define a norm on $SO(3)$, an overview can be found in Huynh (2009). In this work, we use the smallest rotation angle necessary to represent the rotation matrix $\mathbf{R}_1^T\mathbf{R}_2$ using the axis-angle rotation parameterization. Therefore we define our rotational distance measure as

$$d(\mathbf{R}_1, \mathbf{R}_2) = \text{acos}\left(\frac{\text{trace}(\mathbf{R_1^T R_2}) - 1}{2}\right) \tag{3.5}$$

using the formulas from Huynh (2009). The rotational distance measure is for example used in chapter 5 to calculate the rotational alignment error between panoramic images.

3.4 Fourier-Transform and Spectra

A common task in signal processing is to extract meaningful information from signals. For example, the decomposition of a sound signal into its phase and amplitude information can be used for visualization in a music system or voice recognition. To obtain these information, we first have to transform the signal from the spatial domain into the frequency domain. This transformation is called **Fourier transform** and is discussed in the following for complex-valued functions (section 3.4). A detailed introduction to Fourier analysis on complex functions can be found in Gonzalez and Woods (1992), sections 3.2 (Fourier transform) and 3.3.8 (correlations).

The Fourier transform of complex-valued functions $f : \mathbb{R} \to \mathbb{C}$ is calculated by changing the standard basis of a function f to the orthonormal basis $\mathbf{b} = \left\{ e^{ilx} \;\middle|\; l \in \mathbb{Z} \right\}$. The index l (equivalent to the frequency) is in the following called **band**.

Definition 3.1 (Fourier Transform)**.** *Let $f : \mathbb{R} \mapsto \mathbb{C}$ be an integrable function. Then the Fourier transform of f is defined as*

$$\mathcal{F}_f(l) = \int_{\mathbb{R}} f(x) e^{-ilx} dx. \tag{3.6}$$

Definition 3.2 (Inverse Fourier Transform)**.** *Let $f : \mathbb{R} \mapsto \mathbb{C}$ be an integrable function and $g(l) := \mathcal{F}_f(l)$ the Fourier transformed of f. Then the inverse Fourier transform is defined as*

$$\mathcal{F}_g^{-1}(x) = \frac{1}{2\pi} \int_{\mathbb{R}} g(l) e^{ixl} dl. \tag{3.7}$$

In practical applications, the function $f : [a, b] \to \mathbb{C}$ is bounded and discrete. The task is to find a finite number of linear coefficients $\vec{c} = (c_0, c_1, c_1', \ldots, c_{L-1}, c_{L-1}')^T$ for L bands, called **Fourier coefficients**, to represent f as accurately as possible in the basis $\mathbf{b}$. The calculation of these coefficients is called **discrete Fourier transform** (DFT). An algorithm which decreases the computation time of the DFT was developed in its original form by Cooley and Tukey (1965) and is called **fast Fourier transform** (FFT). The FFT applied to RSH is discussed in section 3.10.2.

Now let $\vec{c}$ be the Fourier coefficient vector of f, then the **phase spectrum** and **amplitude spectrum** can be calculated as $\arg(c_l)$ and $|c_l|^2 = c_l\overline{c_l}$, respectively. Interestingly, a phase shift (in this context often called a *translation*) of the signal f does not affect the amplitude spectrum. The phase and amplitude spectra are therefore called **translation-variant** and **translation-invariant**, respectively. However, both spectra only contain a fraction of the information of f and are therefore — each on its own — not capable to restore the function f.

The amplitude spectrum is a first-order spectrum. As discussed in Mendel (1991), higher-order spectra as for example the **bispectrum** can be used to get both the phase and amplitude information of a function f while the spectrum still remains translation-invariant. Note that the zero phase $\varphi_f(0)$ can never be restored from a translation-invariant spectrum since it contains the unknown translation. For continuous functions f, the amplitude spectrum can be calculated as

$$\mathcal{AS}_f(l) = \mathcal{F}_f(l)\overline{\mathcal{F}_f(l)}. \tag{3.8}$$

and the bispectrum as

$$\mathcal{BS}_f(l_1, l_2) = \mathcal{F}_f(l_1)\mathcal{F}_f(l_2)\overline{\mathcal{F}_f(l_1 + l_2)}. \tag{3.9}$$

Higher-order spectra (e.g. the tri-spectrum) exist, however with an increasing order, the calculation of higher-order spectra becomes more complex. Therefore it is uncommon to use spectra with order greater than two in practical applications.

3.5 Fourier Analysis on SO(3)

The Fourier transform as discussed in section 3.4 is commonly applied to periodic, bounded, and complex-valued functions, i.e. functions of the form $f : [0, 2\pi) \to \mathbb{C}$. Instead of defining f on the domain $[0, 2\pi)$, we can also find a mapping between $\alpha \in [0, 2\pi)$ and the rotation matrices $\mathbf{R}_\alpha \in SO(2)$ where α is the rotation angle. This allows us to reformulate the Fourier transform for complex-valued functions defined on the rotation group $SO(2)$. This generalization suggests that the Fourier transform can be generalized to work with various other groups which is, in fact, the basic idea of Fourier analysis[1].

In this section we describe how the Fourier transform can be generalized to functions defined on the rotation group $SO(3)$ by introducing the Wigner-D matrices (section 3.5.1). To calculate Kronecker products of Wigner-D matrices we furthermore introduce Clebsch-Gordan matrices and show how they can be calculated (section 3.5.2). The Clebsch-Gordan matrices are later used to calculate point-wise products and the bispectrum of functions defined on the unit sphere. Functions defined on the unit sphere can be represented in the basis of spherical harmonics (SH) and are a special case of Wigner-D matrices (section 3.5.3). We will use the SH extensively to represent panoramic images as they could be captured by a robot in the frequency domain. The necessary Fourier transform is introduced and a brief summary of important properties of SH is given. Most importantly, we show how rotations of functions defined in the basis of SH can be realized by a simple (and computationally efficient) matrix-vector multiplication with an appropriate Wigner-D matrix. Finally, we present an alternative formulation of SH harmonics which gives us explicit formulas to calculate SH without the necessity of calculating Wigner-D matrices (section 3.5.4)

3.5.1 Wigner-D matrices

From a technical point of view, Wigner-D matrices are the irreducible representations of the rotation group $SO(3)$ (Chen et al., 2002). Loosely speaking, this means that each Wigner-D matrix is a representation of a rotation matrix $\mathbf{R} \in SO(3)$ and behaves — by design — in the same way as the rotation matrix. Most importantly, the Wigner-D matrices allow us to Fourier

[1] It can be shown that the Fourier transform can be defined on Hausdorff locally compact topological groups (Chirikjian and Kyatkin, 2001). A famous and well-studied family of groups which fulfill these conditions are the Lie-groups which have various applications, for example in robotics Chirikjian and Kyatkin (2001), visual odometry Engel et al. (2014), and machine learning Cohen and Welling (2016).

transform a function f defined on the rotations group $SO(3)$. This gives us the possibility to use methods on f which only work in the frequency domain. Before we can define the Wigner-D matrices, we first need to define the direct sum of matrices.

Definition 3.3 (Direct Sum)**.** *Given two matrices* $\mathbf{A} \in \mathrm{Mat}(m \times n, \mathbb{C})$ *and* $\mathbf{B} \in \mathrm{Mat}(p \times q, \mathbb{C})$*, their direct sum is defined as*

$$\mathbf{A} \oplus \mathbf{B} = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{pmatrix} \in \mathrm{Mat}(m+p \times n+q, \mathbb{C}), \tag{3.10}$$

where $\mathbf{0}$ *are matrices filled with zeros.*

Note that the direct sum can be applied to both matrices and vectors. We use the definition of the Wigner-D matrices from Marinucci and Peccati (2011), section 3.3.2, for the explicit calculation of the Wigner-D matrices. This definition uses the ZYZ rotation parameterization of section 3.3.

Definition 3.4 (Wigner-D matrix)**.** *The Wigner-D matrices are defined as* $\mathbf{D}^l = (D^l_{mn})_{mn} \in \mathrm{Mat}(2l+1, \mathbb{C})$*, where the indices for* $l \in \mathbb{N}$ *take values* $m, n \in \{-l, -l+1, \dots, l\}$ *and are ordered as follows:*

$$\mathbf{D}^l = \begin{pmatrix} D^l_{-l,-l} & D^l_{-l,-l+1} & \cdots & D^l_{-l,l} \\ D^l_{-l+1,-l} & D^l_{-l+1,-l+1} & \cdots & D^l_{-l+1,l} \\ \vdots & \vdots & \ddots & \vdots \\ D^l_{l,-l} & D^l_{l,-l+1} & \cdots & D^l_{l,l} \end{pmatrix} \tag{3.11}$$

Using the ZYZ parameterization from section 3.3, the entries itself are defined as

$$D^l_{mn}(\mathbf{R}) = D^l_{mn}(\gamma, \beta, \alpha) = e^{-in\gamma} \lambda^l_{mn}(\beta) e^{-im\alpha} \tag{3.12}$$

with

$$\begin{aligned} \lambda^l_{mn}(\beta) &:= (-1)^{m-n} \sqrt{(l+m)!(l-m)!(l+n)!(l-n)!} \\ &\quad \cdot \sum_s \left[\frac{(-1)^s \left(\cos\frac{\beta}{2}\right)^{2l-m+n-2s} \left(\sin\frac{\beta}{2}\right)^{m-n+2s}}{(l+n-s)!s!(m-n+s)!(l-m-s)!} \right] \\ &= \lambda^l_{nm}(-\beta). \end{aligned} \tag{3.13}$$

The index s *runs through all values* $s \in \mathbb{N}$ *for which the faculties are well defined (greater equal zero). For integer values* $l \in \mathbb{N}$*, we denote by*

$$\mathbf{D}^L = \bigoplus_{l=0}^{L-1} \mathbf{D}^l \tag{3.14}$$

the direct sum of all Wigner-D matrices with integer band l *up to degree* $L-1$*. This notation is ambiguous if we choose a concrete value for* l *or* L*; in this case we always refer to the lowercase, e.g.* $\mathbf{D}^3$ *means* $l = 3$*.*

We use the notation of lowercase 'l' (individual objects) and uppercase 'L' (direct sum of objects) not only for Wigner-D matrices but also for other objects like for example spherical harmonics (Y^l and Y^L) and Fourier coefficient vectors ($\vec{A}^l$ and $\vec{A}^L$). The annotation from the definition of Wigner-D matrices (definition 3.4) holds true in all of these cases.

In the following we write $\mathbf{D}^l(\mathbf{R})$ to describe a rotation $\mathbf{D}^l(\gamma, \beta, \alpha)$, where $\mathbf{R}$ is the rotation matrix observed by a ZYZ rotation around the angles (γ, β, α). An example for indices used in Wigner-D matrices can be found in example 3.2. As mentioned before, a Wigner-D matrix $\mathbf{D}^l(\mathbf{R})$

Example 3.2: Indexing of Wigner-D matrices

The Wigner-D matrix $\mathbf{D}^1$ is indexed as follows:

$$\mathbf{D}^1 = \begin{pmatrix} D^1_{-1,-1} & D^1_{-1,0} & D^1_{-1,1} \\ D^1_{0,-1} & D^1_{0,0} & D^1_{0,1} \\ D^1_{1,-1} & D^1_{1,0} & D^1_{1,1} \end{pmatrix} \tag{3.15}$$

The element $D^l_{0,0}$ is always in the center.

behaves in the same way as the rotation matrix $\mathbf{R}$. Mathematically, this is ensured through the relation

$$\mathbf{D}^l(\mathbf{R}_1\mathbf{R}_2) = \mathbf{D}^l(\mathbf{R}_1)\mathbf{D}^l(\mathbf{R}_2). \tag{3.16}$$

Moreover, the entries of Wigner-D matrices form an orthogonal basis

$$\left\langle D^l_{mn}(\mathbf{R}), D^{l'}_{m'n'}(\mathbf{R}) \right\rangle = \int_{SO(3)} D^l_{mn}(\mathbf{R})\overline{D^{l'}_{m'n'}(\mathbf{R})}\, d\mathbf{R} = \frac{8\pi^2}{2l+1}\delta_{l,l'}\delta_{m,m'}\delta_{n,n'} \tag{3.17}$$

which can be shown using **Schur's orthogonality relation** (Marinucci and Peccati, 2011, Th. 2.33). In section 3.5.3 it can be seen that this property is inherited by the spherical harmonics and is used in section 3.9 to calculate the integral squared error.

As described in Kostelec and Rockmore (2008), a two-times integrable function $f : SO(3) \to \mathbb{C}$ can be approximated for L bands by the sum

$$f(\mathbf{R}) \approx \sum_{l=0}^{L-1} \frac{2l+1}{8\pi} \sum_{m,n=-l}^{l} A^l_{mn} D^l_{mn}(\mathbf{R}), \tag{3.18}$$

where the coefficients A^l_{mn} are obtained by

$$A^l_{mn} = \int_{SO(3)} f(\mathbf{R}) D^l_{mn}(\mathbf{R}) d\mathbf{R}. \tag{3.19}$$

This is also called the **Fourier transform on SO(3)**. Note that the Fourier transform on $SO(3)$ is linear.

3.5.2 Clebsch-Gordan Matrices

To calculate the bispectrum (section 3.9.3) and the point-wise product (section 3.5.3) of spherical harmonics, we need to introduce the concept of **Clebsch-Gordan matrices**. Clebsch-Gordan matrices appear in the computation of **Kronecker products**[2]between Wigner-D matrices, which is defined as follows:

Definition 3.5 (Kronecker Product)**.** *Let* $\mathbf{A} \in \mathrm{Mat}(m \times n, \mathbb{R})$ *and* $\mathbf{B} \in \mathrm{Mat}(p \times q, \mathbb{R})$*, then the Kronecker product is defined as:*

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \dots & a_{1,n}\mathbf{B} \\ a_{2,1}\mathbf{B} & a_{2,2}\mathbf{B} & \dots & a_{2,n}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1}\mathbf{B} & a_{m,2}\mathbf{B} & \dots & a_{m,n}\mathbf{B} \end{pmatrix} \in \mathrm{Mat}(mp \times nq) \tag{3.20}$$

The resulting matrix is of dimension $mp \times nq$*.*

[2] The Kronecker product is a special case of the tensor product, i.e. for homomorphisms between vector spaces with fixed bases which can be represented by matrices. In general, the tensor product is an abstract generalization of the outer product between vectors and an essential concept for the study of groups, rings, and similar.

It can be shown that the Kronecker product between Wigner-D matrices can be calculated as

$$\mathbf{D}^{l_1} \otimes \mathbf{D}^{l_2} = \mathbf{C}_{l_1,l_2} \left[\bigoplus_{i=|l_2-l_1|}^{l_2+l_1} \mathbf{D}^i \right] \mathbf{C}_{l_1,l_2}^\dagger, \tag{3.21}$$

where the matrices $\mathbf{C}_{l_1,l_2}$ are the Clebsch-Gordan matrices (Marinucci and Peccati, 2011, Eq. 3.55). The Clebsch-Gordan matrices are crucial to calculate point-wise products as shown in section 3.5.5. The entries of Clebsch-Gordan matrices are called **Clebsch-Gordan coefficients** and have many applications in **quantum mechanics**, especially angular momentum coupling. An introduction to Clebsch-Gordan coefficients as used in physics can be found in Chen et al. (2002), section 3.15. In this section, we address two problems: First, the calculation of Clebsch-Gordan coefficients and, second, the ordering of Clebsch-Gordan coefficients in Clebsch-Gordan matrices.

We denote the Clebsch-Gordan coefficients by $c^{l,m}_{l_1,m_1,l_2,m_2}$ with $l, m, l_1, m_1, l_2, m_2 \in \mathbb{N}$. Explicit formulas (Rudnicki-Bujnowski, 1975) allow direct calculation of the Clebsch-Gordan coefficients. For example the **Racah formula** states

$$c^{l,m}_{l_1,m_1,l_2,m_2} = \sum_s \frac{(-1)^s \sqrt{\triangle(l,l_1,l_2)(2l+1)(l_1+m_1)(l_1-m_1)(l_2+m_2)(l_2-m_2)(l+m)(l-m)}}{[s!(l-l_2+s+m_1)!(l-l_1+s-m_2)!(l_1+l_2-l-s)!(l_1-s-m_1)!(l_2-s+m_2)!]}, \tag{3.22}$$

where

$$\triangle(l,l_1,l_2) = \frac{(l_1+l_2-l)!(l_2+l-l_1)!(l+l_1-l_2)!}{(l_1+l_2+l+1)!}. \tag{3.23}$$

Note that the index s runs through all values $s \in \mathbb{N}$ for which all faculties are well-defined (greater equal zero). The Clebsch-Gordan coefficients are real-valued and due to the symmetry of Clebsch-Gordan coefficients given by

$$c^{l,m}_{l_1,m_1,l_2,m_2} = (-1)^{L-l_1-l_2} c^{l,m}_{l_2,m_2,l_1,m_1}, \tag{3.24}$$

where L is the maximum number of bands, it is sufficient to calculate all Clebsch-Gordan coefficients with $l_2 \geq l_1$ (Homeier and Steinborn, 1996). In the following we always assume that $l_2 \geq l_1$. The number of non-zero coefficients $c^{l,m}_{l_1,m_1,l_2,m_2}$ is restricted by the **triangle condition** (Marinucci and Peccati, 2011, Remark 3.38)

$$l \leq l_1 + l_2 \ \vee \ l_1 \leq l + l_2 \ \vee \ l_2 \leq l_1 + l \quad \Rightarrow \quad c^{l,m}_{l_1,m_1,l_2,m_2} = 0 \tag{3.25}$$

and — as stated by Straub (2014) — by

$$m_1 + m_2 \neq m \quad \Rightarrow \quad c^{l,m}_{l_1,m_1,l_2,m_2} = 0. \tag{3.26}$$

The Clebsch-Gordan coefficients can be calculated using — as usual in the context of Wigner-D matrices — several recurrence relations. Algorithms based on recurrence relations can be found in Zuo et al. (1993) and Straub (2014). So far, we are able to calculate the Clebsch-Gordan coefficients, however their location (i.e. row and column indices) in Clebsch-Gordan matrices — as for example stated by the implicit ordering for the rows and columns depending on the indices l, m, l_1, m_1, l_2, m_2 from Marinucci and Peccati (2011), Remark 3.40 — is not intuitively clear. To efficiently create Clebsch-Gordan matrices, we derived a closed form solution for the row and column indices for each Clebsch-Gordan coefficient:

Lemma 3.6. *Let $c^{l,m}_{l_1,m_1,l_2,m_2}$ be a Clebsch-Gordan coefficient. Then the row and column indices of their entries in the Clebsch-Gordan matrix $\mathbf{C}_{l_1,l_2}$ are given by*

$$row(c^{l,m}_{l_1,m_1,l_2,m_2}) = (l_1+m_1)(2l_2+1) + l_2 + m_2 + 1 \tag{3.27}$$

$$col(c^{l,m}_{l_1,m_1,l_2,m_2}) = l^2 - (l_2-l_1)^2 + l + m + 1 \tag{3.28}$$

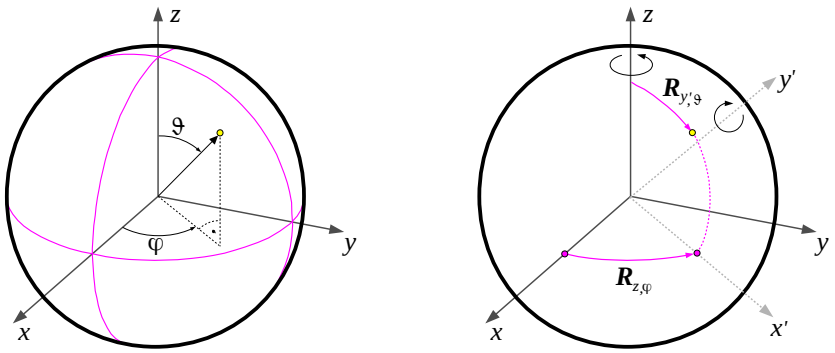


Figure 3.2: Left: The spherical coordinates identify each point (yellow dot) on S^2 with two angles, the azimuth angle φ and the altitude angle ϑ. Rotations along the X, Y, and Z axis would rotate a point on the sphere parallel to the corresponding orbits (magenta lines). Right: A point on the sphere can be placed in an arbitrary position on the sphere by applying a rotation around the Z axis followed by a rotation around the new Y' axis.

Clebsch-Gordan matrices are indexed normally, i.e. the indices for rows and columns start by 1. Note that these formulas are only valid for Clebsch-Gordan coefficient which fulfill the triangle conditions, i.e. equation (3.25) and (3.26).

Proof. The proof can be found in appendix A.2. □

An example of the row and column indices in a Clebsch-Gordan matrix is given in example 3.3. Now we can finally define the Clebsch-Gordan matrix as follows:

Definition 3.7 (Clebsch-Gordan Matrix)**.** *Let $l_1, l_2 \in \mathbb{N}$ with $l_2 \geq l_1$. Then the matrix $\mathbf{C}_{l_1,l_2} \in \mathrm{Mat}(n, \mathbb{R})$ with $n = (2l_1 + 1)(2l_2 + 1)$, the orderings defined in lemma 3.6, and the corresponding Clebsch-Gordan coefficients is called **Clebsch-Gordan matrix**.*

Our implementation for the computation of Clebsch-Gordan matrices is shown in appendix C.1; to compute the Clebsch-Gordan coefficients, we used the algorithm suggested by Straub (2014).

Example 3.3: Clebsch-Gordan Matrix

The Clebsch-Gordan matrix $\mathbf{C}_{1,1}$, is given by

$$\begin{pmatrix}
c^{0,0}_{1,-1,1,-1} & c^{1,-1}_{1,-1,1,-1} & c^{1,0}_{1,-1,1,-1} & c^{1,1}_{1,-1,1,-1} & c^{2,-2}_{1,-1,1,-1} & c^{2,-1}_{1,-1,1,-1} & c^{2,0}_{1,-1,1,-1} & c^{2,1}_{1,-1,1,-1} & c^{2,2}_{1,-1,1,-1} \\
c^{0,0}_{1,-1,1,0} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
c^{0,0}_{1,-1,1,1} & \cdots & \cdots & \cdots & c^{2,-2}_{1,-1,1,1} & \cdots & \cdots & \cdots & c^{2,2}_{1,-1,1,1} \\
c^{0,0}_{1,0,1,-1} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
c^{0,0}_{1,0,1,0} & \cdots & c^{1,0}_{1,0,1,0} & \cdots & c^{2,-2}_{1,0,1,0} & \cdots & c^{2,0}_{1,0,1,0} & \cdots & c^{2,2}_{1,0,1,0} \\
c^{0,0}_{1,0,1,1} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
c^{0,0}_{1,1,1,-1} & \cdots & \cdots & \cdots & c^{2,-2}_{1,1,1,-1} & \cdots & \cdots & \cdots & c^{2,2}_{1,1,1,-1} \\
c^{0,0}_{1,1,1,0} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
c^{0,0}_{1,1,1,1} & \cdots & c^{1,0}_{1,1,1,1} & \cdots & c^{2,-2}_{1,1,1,1} & \cdots & c^{2,0}_{1,1,1,1} & \cdots & c^{2,2}_{1,1,1,1}
\end{pmatrix},$$

where $c^{l,m}_{l_1,m_1,l_2,m_2}$ are the Clebsch-Gordan coefficients.

3.5.3 Spherical Harmonics

Spherical harmonics (SH) are a tool frequently used to describe functions on the unit sphere S^2. The set of SH forms an orthonormal basis which can be used to map functions defined on S^2 into the basis of SH (Fourier transform). The basic idea to derive the SH is the following: We

construct a mapping between the unit sphere S^2 and the rotation group $SO(3)$ which allows us to define the SH as a subset of the Wigner-D matrices. Since the SH are a subset of the Wigner-D matrices, we can finally derive many properties directly from the Wigner-D matrices, especially the Fourier transform for SH. This section is based on Marinucci and Peccati (2011), section 3.4.

To identify S^2 with the rotation group $SO(3)$, we first have to introduce spherical coordinates.

Definition 3.8 (Spherical Coordinates)**.** *Let $\vec{p} = (x, y, z)^T \in S^2$ be a vector of the unit sphere in Cartesian coordinates, then we denote the tuple*

$$(\vartheta, \varphi) = (\text{acos}(z), \text{atan2}(y, x)) \tag{3.29}$$

*as **spherical coordinates**. Spherical coordinates can be mapped back to Cartesian coordinates via*

$$(x, y, z)^T = (\sin\vartheta\cos\varphi, \sin\vartheta\sin\varphi, \cos\vartheta)^T. \tag{3.30}$$

Note that by setting $\varphi = 0$ if either $\vartheta = 0$ or $\vartheta = \pi$, the tuple $(\vartheta, \varphi) \subseteq [0, \pi] \times [0, 2\pi)$ can uniquely be identified with $\vec{p}$.

In the following we write by abuse of notation elements of the unit sphere as vectors $\vec{p} \in S^2$, as spherical coordinate (ϑ, φ), or as rotation matrix $\mathbf{R}$. The last notation — writing an element of the unit sphere as a rotation matrix — is justified by the mapping (3.32). Figure 3.2 illustrates the tuple (ϑ, φ) and the relation between S^2 and the rotation group $SO(3)$. As can be seen, any point $\vec{p} \in S^2$ on the sphere with spherical coordinates (ϑ, φ) can be expressed as a rotation

$$\vec{p} = (\vartheta, \varphi) = \mathbf{R}_{Y,\vartheta-\frac{\pi}{2}}\mathbf{R}_{Z,\varphi}(1, 0, 0)^T. \tag{3.31}$$

Therefore we can define a mapping

$$\phi : S^2 \to SO(3) \quad \text{with} \quad \phi(\vec{p}) = \mathbf{R}_{Y,\vartheta}\mathbf{R}_{Z,\varphi}. \tag{3.32}$$

which identifies each position $\vec{p} \in S^2$ with a ZYZ rotation (γ, β, α), where $\alpha = \varphi$, $\beta = \vartheta$ and γ is an arbitrary but fixed angle; in the following we use $\gamma = 0$. Now we can define the spherical harmonics as a subset of the Wigner-D matrices as follows:

Definition 3.9 (Spherical Harmonics)**.** *The SH $Y_m^l : S^2 \to \mathbb{C}$ are defined as*

$$Y_m^l(\vartheta, \varphi) := \sqrt{\frac{2l+1}{4\pi}}\bar{D}_{m0}^l(\phi(\vartheta, \varphi)) \overset{(3.32)}{=} \sqrt{\frac{2l+1}{4\pi}}\bar{D}_{m0}^l(\varphi, \vartheta, 0). \tag{3.33}$$

*with $l \in \mathbb{N}$ and $m \in \{-l, \dots, l\}$. The index l is called **band** and corresponds to the frequency of the SH. Note that we use the mapping ϕ from equation (3.32) to map a point $(\vartheta, \varphi) \in S^2$ on the sphere to a rotation $\phi(\vartheta, \varphi) = (\varphi, \vartheta, 0) \in SO(3)$. We denote by*

$$\vec{Y}^l := (Y_{-l}^l, \dots, Y_l^l)^T$$

the vector containing the SH of band l and by

$$\vec{Y}^L = \bigoplus_{l=0}^{L-1} \vec{Y}^l = (Y_0^0, Y_{-1}^1, Y_0^1, Y_1^1, \dots, Y_{L-1}^{L-1})^T$$

the vector containing all SH for L bands.

Since SH form a subset of Wigner-D matrices, several properties can be derived from the Wigner-D matrices: First, the SH form an orthonormal basis, i.e.

$$\left\langle Y_m^l, Y_{m'}^{l'} \right\rangle = \int_{S^2} Y_m^l(\vec{p})\overline{Y_{m'}^{l'}(\vec{p})}d\vec{p} = \delta_{l,l'}\delta_{m,m'}. \tag{3.34}$$

Second, rotations are calculated band-wise such that rotations can be realized via sparse matrix-vector multiplications (section 3.7). Third, as the following theorem shows, using SH the Fourier transform can be defined for functions f defined on the unit sphere:

Theorem 3.10 (Peter-Weyl Theorem for Spherical Harmonics)**.** *Let f be a complex-valued function defined on S^2. Then f can be approximated in the basis of SH as*

$$f(\vartheta, \varphi) \approx \sum_{l=0}^{L-1} \sum_{m=-l}^{l} A_m^l Y_m^l(\vartheta, \varphi), \tag{3.35}$$

where the Fourier coefficients are given by

$$A_m^l = \int_{S^2} f(\vec{p}) \bar{Y}_m^l(\vec{p}) d\vec{p} \; = \int_{\varphi=0}^{2\pi} \int_{\vartheta=0}^{\pi} f(\vartheta, \varphi) \bar{Y}_m^l(\vartheta, \varphi) \sin\vartheta d\vartheta d\varphi. \tag{3.36}$$

Note that the sine term in equation (3.36) is a consequence of the coordinate transform (integrating in spherical coordinates). For an increasing value of L, the Fourier series converges quadratically to the function f.

Proof. See Marinucci and Peccati (2011), proposition 3.29. □

In the following, we use the same notation as in definition 3.9 to denote the stacked vectors $\vec{A}^L = \bigoplus_{l=0}^{L-1} \vec{A}^l$. The Peter-Weyl theorem for SH generalizes the standard Fourier transform on S^1 onto the sphere S^2. As can be seen in example 3.4, the approximation of a function f converges quickly towards the original function f with increasing band L. However, in comparison to the standard 1D Fourier transform, the number of Fourier coefficients increases quadratically, i.e. a Fourier coefficient vector of band L has L^2 entries, limiting the maximal number of bands which can be determined from the signal f in practical applications.

Now we discuss how spherical harmonics can be rotated using Wigner-D matrices (adapted from Marinucci and Peccati (2011), section 3.4.2). Let $\vec{p} \in S^2$ be an arbitrary point on the unit sphere and $\mathbf{R} \in SO(3)$ a rotation matrix. Then it is a natural choice to define the rotation of a spherical harmonic as

$$\mathbf{R} \circ Y_m^l(\vec{p}) := Y_m^l(\mathbf{R}\vec{p}). \tag{3.37}$$

We use a trick to calculate the right side of the equation: The new coordinate of the rotated point is given by $\vec{q} = \mathbf{R}\vec{p}$. Using the mapping ϕ from equation (3.32), which maps elements from S^2 to $SO(3)$, we have

$$\phi(\vec{q}) = \phi(\mathbf{R}\vec{p}) = \mathbf{R}\phi(\vec{p}). \tag{3.38}$$

Here we use that it is not important if we rotate the point $\vec{p}$ first and then map it into $SO(3)$ or vice versa. From definition 3.9 we have that the spherical harmonics are defined as entries of Wigner-D matrices an by applying equation (3.38) we obtain:

$$Y_m^l(\mathbf{R}\vec{p}) \overset{\text{Def. 3.9}}{=} \sqrt{\frac{2l+1}{4\pi}} \bar{\mathbf{D}}_{m0}^l(\phi(\mathbf{R}\vec{p})) \overset{(3.38)}{=} \sqrt{\frac{2l+1}{4\pi}} \bar{\mathbf{D}}_{m0}^l(\mathbf{R}\phi(\vec{p})) \tag{3.39}$$

From equation (3.16) we have $\mathbf{D}^l(\mathbf{R}\phi(\vec{p})) = \mathbf{D}^l(\mathbf{R})\mathbf{D}^l(\phi(\vec{p}))$. For a product of matrices, the value of each element with indices p, q in the resulting matrix can be calculated as a sum. In this case we have

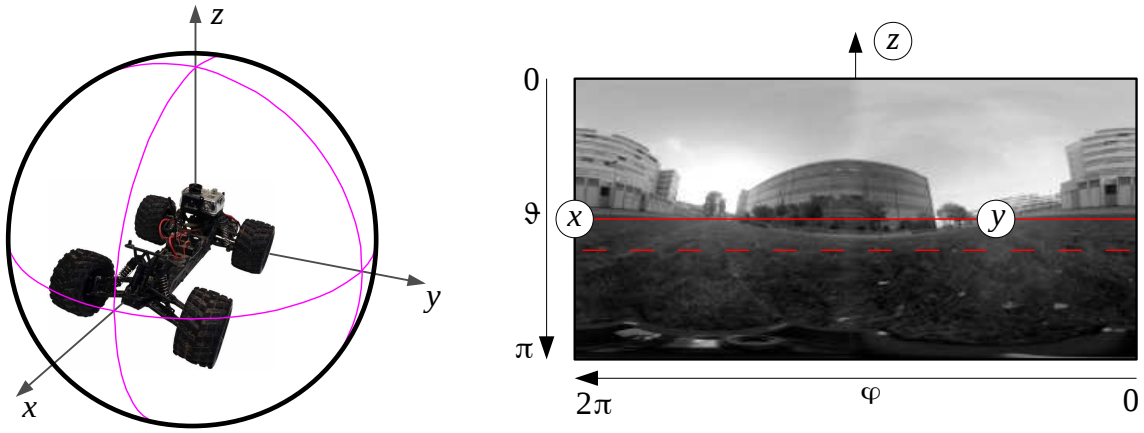
$$\left(\mathbf{D}^l(\mathbf{R})\mathbf{D}^l(\phi(\vec{p}))\right)_{p,q} = \sum_{n=-l}^{l} \mathbf{D}_{pn}^l(\mathbf{R}) \mathbf{D}_{nq}^l(\phi(\vec{p})) \tag{3.40}$$

which can be substituted into equation (3.39) to obtain

$$= \sqrt{\frac{2l+1}{4\pi}} \sum_{n=-l}^{l} \bar{\mathbf{D}}_{mn}^l(\mathbf{R}) \bar{\mathbf{D}}_{n0}^l(\phi(\vec{p})) \overset{\text{Def. 3.9}}{=} \sum_{n=-l}^{l} \bar{\mathbf{D}}_{mn}^l(\mathbf{R}) Y_n^l(\vec{p}). \tag{3.41}$$

Example 3.4: Fourier Series on the Sphere (Spherical Harmonics)

Let f be a function over S^2 used to describe the full-spherical panoramic image as captured by an omnidirectional camera mounted on an agent:



The left image shows the how the axes are aligned with an agent. We always assume that the agent is heading into the direction of the X-axis and mounted with a camera aligned with the Z-axis, completely determining the right-handed coordinate system. The right image shows how spherical coordinates are used to represent the visual field (panoramic image) as seen by the agent. The axes of the underlying coordinate system are marked by white circles. Using a fish-eye objective, not the complete visual scene can be observed; wide-angle fish-eye lenses have an angle of view between 180° (solid red line) and 220° (dashed red line). Therefore a function f describing the visual scene contains only information about the upper hemisphere, while it is completely (180°) or partially (220°) undefined on the lower hemisphere. We treat the problem of hemispherical functions in the sections 3.6.2 and 3.10.3. By applying the Fourier transformation as stated in equation (3.36) to approximate f for a maximum of L bands, we obtain a band-limited representation of f in the frequency domain:

$L = 5$ $L = 10$ $L = 15$

$L = 20$ $L = 30$ $L = 40$

The images show the resulting approximation of f for an increasing band L. Throughout this work we visualize Fourier transformed functions f by applying the inverse Fourier transform afterwards to obtain a representation of it in the spatial domain.

In the last step, we resubstituted the spherical harmonic for the Wigner-D matrix. Summarized, we get the rotated spherical harmonic

$$\mathbf{R} \circ Y_m^l(\vec{p}) = \sum_{n=-l}^{l} \bar{\mathbf{D}}_{mn}^l(\mathbf{R}) Y_n^l(\vec{p}) \tag{3.42}$$

or written in matrix form

$$\mathbf{R} \circ \vec{Y}^l(\vec{p}) := \vec{Y}^l(\mathbf{R}\vec{p}) \overset{(3.42)}{=} \bar{\mathbf{D}}^l(\mathbf{R}) \vec{Y}^l(\vec{p}). \tag{3.43}$$

The following lemma gives us some more general information about rotations of SH and Wigner-D matrices. Since the results are also true for the real Wigner-D matrices and real spherical harmonics (RSH) introduced later in section 3.6, we state the lemma for both cases.

Lemma 3.11. *Let Y_m^l, Y_n^l be SH, $\mathbf{R} \in SO(3)$ be a rotation matrix, and $\mathbf{D}^l$ be the corresponding Wigner-D matrix of band l. Alternatively, let y_m^l, y_n^l be RSH and $\mathbf{D}^l$ be the corresponding real Wigner-D matrix. Then we have:*

(i) $\left\langle \mathbf{R} \circ Y_m^l, Y_n^l \right\rangle = \left\langle Y_m^l, \mathbf{R}^T \circ Y_n^l \right\rangle$ *and* $\left\langle \mathbf{R} \circ y_m^l, y_n^l \right\rangle = \left\langle y_m^l, \mathbf{R}^T \circ y_n^l \right\rangle$

(ii) $\left[\mathbf{D}^l(\mathbf{R})\right]^\dagger = \mathbf{D}^l(\mathbf{R}^T)$ *and* $\left[\mathbf{d}^l(\mathbf{R})\right]^T = \mathbf{d}^l(\mathbf{R}^T)$

Proof. The proof can be found in appendix A.5. □

We use this lemma in the proof of theorem 3.24 (sparsity of Y-axis rotations) and to calculate rotations of functions on S^2: Let f be a complex-valued function on S^2 and $\vec{A}^L$ its Fourier coefficient vector, then we have from definition 3.9 and theorem 3.10 that $f = \sum_{l=0}^{L-1} \left[\vec{Y}^l\right]^T \vec{A}^l$. By defining the rotation on a function f as before, we have

$$\begin{aligned} \mathbf{R} \circ f(\vec{p}) \;&:=\; f(\mathbf{R}\vec{p}) = \sum_{l=0}^{L-1} \left[\vec{Y}^l(\mathbf{R}\vec{p})\right]^T \vec{A}^l \\ &\overset{(3.43)}{=} \sum_{l=0}^{L-1} \left[\bar{\mathbf{D}}^l(\mathbf{R}) \vec{Y}^l(\vec{p})\right]^T \vec{A}^l = \sum_{l=0}^{L-1} \left[\vec{Y}^l(\vec{p})\right]^T \left[\mathbf{D}^l(\mathbf{R})\right]^\dagger \vec{A}^l \\ &\overset{\text{Le. } 3.11}{=} \sum_{l=0}^{L-1} \left[\vec{Y}^l(\vec{p})\right]^T \mathbf{D}^l(\mathbf{R}^T) \vec{A}^l. \end{aligned} \tag{3.44}$$

In the last step we use lemma 3.11 to get rid of the complex conjugate. Now we are able to calculate the rotation of a Fourier transformed function f on S^2 by rotating its associated Fourier coefficient vector:

Definition 3.12 (Rotation of Spherical Harmonics)**.** *Let f be a complex-valued function on S^2 and $\vec{A}^L$ its Fourier coefficient vector, then the Fourier coefficient vector of the rotated function $\mathbf{R} \circ f$ is given by*

$$\mathbf{R} \circ \vec{A}^l := \mathbf{D}^l(\mathbf{R}^T) \vec{A}^l, \tag{3.45}$$

which is a simple matrix-vector multiplication.

In section 3.7 we show that Wigner-D matrices for rotations around the X/Y/Z-axes are sparse and that their rotations can be implemented computationally cheap.

3.5.4 Alternative Formulation of Spherical Harmonics

The SH can be calculated using orthogonal polynomials, i.e. **associated Legendre polynomials**. For practical applications it is often advantageous to use these formulas since they unveil many symmetry properties which cannot directly be seen from the equations in definition 3.4 or from recursive formulas (section 3.6.1). A derivation of these formulas can be found in common harmonic analysis books, e.g. Folland (1992), section 6.3, and Byerly (1893), chapter 6.

Definition 3.13 (Associated Legendre Polynomial)**.** *Polynomials of the form*

$$P_m^l(x) = \frac{(-1)^m}{2^l l!}(1-x^2)^{\frac{m}{2}}\frac{d^{l+m}}{dx^{l+m}}\left[(x^2-1)^l\right] \tag{3.46}$$

with $0 \leq l \in \mathbb{N}$ *and* $m \in \mathbb{N}$ *with* $|m| \leq l$ *are called associated Legendre polynomials.*

In section 3.6.1, a recursive algorithm is shown to effectively compute the values P_m^l. The alternative formulation of SH originates from their classical derivation[3]. In terms of associated Legendre polynomials we can rewrite these solutions (i.e. the SH) as

$$Y_m^l(\vartheta,\varphi) = \begin{cases} K_m^l e^{im\varphi} P_m^l(\cos\vartheta) & m \geq 0 \\ (-1)^m \bar{Y}_{-m}^l(\vartheta,\varphi) & m < 0, \end{cases} \tag{3.47}$$

where

$$K_m^l = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}. \tag{3.48}$$

As can be seen, the SH consist of three parts: First, the normalization factor K_m^l ensures that the set of SH forms an orthonormal basis. Second, the behavior of the SH around lines parallel to the equator (i.e. ϑ is constant) is given by the functions $e^{im\varphi}$. Third, the behavior of the SH towards the poles (i.e. φ is constant) is defined by the associated Legendre polynomials $P_m^l(\cos\vartheta)$. In section 3.6 we use this alternative formulation of the SH to define the **real spherical harmonics** (RSH).

3.5.5 Point-Wise Products

In chapter 5, we introduce weighting functions to calculate the weighted squared integral error between two functions f, g in the basis of RSH (section 3.9). This calculation requires the computation of the **point-wise product** between two functions in the basis of SH (example 3.5): Let f, g be two functions defined on the unit sphere S^2, then the point-wise product is defined as $(f \cdot g)(\vec{p}) = f(\vec{p})g(\vec{p})$. Let $\vec{A}^{L_1}$ and $\vec{B}^{L_2}$ be the Fourier coefficients of the functions f, g, then the point-wise product is given by

$$(f \cdot g)(\vec{p}) = \left(\sum_{l=0}^{L_1-1}\sum_{m=-l}^{l} A_m^l Y_m^l(\vec{p})\right)\left(\sum_{l'=0}^{L_2-1}\sum_{m'=-l'}^{l'} B_{m'}^{l'} Y_{m'}^{l'}(\vec{p})\right) \tag{3.49}$$

$$= \sum_{l=0}^{L_1-1}\sum_{m=-l}^{l}\sum_{l'=0}^{L_2-1}\sum_{m'=-l'}^{l'} A_m^l B_{m'}^{l'} Y_m^l(\vec{p}) Y_{m'}^{l'}(\vec{p}). \tag{3.50}$$

Now two questions arise: 'What is each product $Y_m^l(\vec{p})Y_{m'}^{l'}(\vec{p})$?' and 'What is the resulting Fourier coefficient vector $\vec{A}^{L_1} \cdot \vec{B}^{L_2}$ of the function $(f \cdot g)(\vec{p})$?'. In the following we answer both of these questions and finally give a closed form solution to calculate point-wise products directly in the basis of SH (equation (3.54)).

[3] The proof is complicated, a detailed explanation can be found in Mohlenkamp (2016). The key idea is the following: It can be shown that the spherical Laplace operator is self-adjoint. As a consequence, the eigenfunctions of the solution of the spherical Laplace operator are orthogonal for different eigenvalues. These eigenfunctions are the desired SH.

Example 3.5: Point-Wise Products

$\vec{A}^{L_1}$ $\vec{B}^{L_2}$ $\vec{A}^{L_1} \cdot \vec{B}^{L_2}$

Let f, g be two functions defined on the unit sphere, then the point-wise product is defined as $(f \cdot g)(\vec{p}) = f(\vec{p})g(\vec{p})$. The point-wise product can also be calculated in terms of spherical harmonics. Let $\vec{A}^{L_1}$ and $\vec{B}^{L_2}$ be the Fourier coefficients of the functions f, g, then the point-wise product $f \cdot g$ can again be expressed using SH. The resulting Fourier coefficients $\vec{A}^{L_1} \cdot \vec{B}^{L_2}$ of the point-wise product $f \cdot g$ can be calculated using theorem 3.14 and equation (3.54). For both the theorem and the equation, the computation of Clebsch-Gordan matrices is required.

As shown in Hanyk (1999), appendix A.4, Clebsch-Gordan matrices (section 3.5.2) are directly related to the point-wise product: By defining **coupling coefficients**

$$\tilde{c}^{l,m}_{l_1,m_1,l_2,m_2} := \sqrt{\frac{(2l_1+1)(2l_2+1)}{4\pi(2l+1)}} c^{l,0}_{l_1,0,l_2,0} c^{l,m}_{l_1,m_1,l_2,m_2}, \tag{3.51}$$

based on the Clebsch-Gordan coefficients $c^{l,m}_{l_1,m_1,l_2,m_2}$, the point-wise product between two SH can be calculated as

$$Y^{l_1}_{m_1}(\vec{p}) Y^{l_2}_{m_2}(\vec{p}) = \sum_{l=|l_2-l_1|}^{l_2+l_1} \sum_{m=-l}^{l} \tilde{c}^{l,m}_{l_1,m_1,l_2,m_2} Y^{l}_{m}(\vec{p}). \tag{3.52}$$

Recall from the definition of the SH (definition 3.9) that SH differ from entries of the Wigner-D matrices by a scalar factor which only depends on the band l. These scalar factors ensure that the SH form an orthonormal basis. For point-wise products, the usage of coupling coefficients instead of Clebsch-Gordan coefficients is again necessary to maintain the orthonormality of the SH (Homeier and Steinborn, 1996).

Now we define the **coupling matrix** $\tilde{\mathbf{C}}_{l_1,l_2}$ in the same way as the Clebsch-Gordan matrix $\mathbf{C}_{l_1,l_2}$, but with the coefficients $c^{l,m}_{l_1,m_1,l_2,m_2}$ replaced by $\tilde{c}^{l,m}_{l_1,m_1,l_2,m_2}$. This allows us to compute point-wise products directly in the basis of SH as shown by the following theorem:

Theorem 3.14 (Point-Wise Product)**.** *Let f_1, f_2 be complex-valued functions on S^2 with Fourier coefficients $\vec{A}^{L_1}$ and $\vec{B}^{L_2}$. Then the point-wise product between band l_1 of $\vec{A}^{L_1}$ and band l_2 of $\vec{B}^{L_2}$ is given by*

$$\vec{A}^{l_1} \cdot \vec{B}^{l_2} := \tilde{\mathbf{C}}^{\dagger}_{l_1,l_2} \left[\vec{A}^{l_1} \otimes \vec{B}^{l_2} \right] \tag{3.53}$$

where $\tilde{\mathbf{C}}_{l_1,l_2}$ is the coupling matrix. Note that the resulting vector contains only the non-zero entries for the bands $|l_2 - l_1|, \ldots, l_2 + l_1$ (triangle condition, section 3.5.2); the result is commonly not *a Fourier coefficient vector.*

Proof. See Hanyk (1999), appendix A.4. □

As can be seen from equation (3.52), the point-wise product $\vec{A}^{l_1} \cdot \vec{B}^{l_2}$ between the bands l_1 and l_2 is a vector with entries for all bands l which fulfill $|l_2 - l_1| \leq l \leq l_2 + l_1$. Therefore the result is commonly *not* a Fourier coefficient vector but only a segment of it; more precisely the entries for the bands $0 \leq l < |l_2 - l_1|$ and $l_2 + l_1 < l < L_1 + L_2 - 1$ are missing. This can simply

	complex	real
Spherical harmonics	Y_m^l	y_m^l
Fourier coefficients	A_m^l	a_m^l
Wigner-D matrices	$\mathbf{D}^l$	$\mathbf{d}^l$
Clebsch-Gordan matrices	$\mathbf{C}_{l_1,l_2}$	$\mathbf{c}_{l_1,l_2}$
Coupling matrices	$\tilde{\mathbf{C}}_{l_1,l_2}$	$\tilde{\mathbf{c}}_{l_1,l_2}$

Table 3.1: Overview of names used in the basis of SH and their real counterparts involved in various calculations throughout this work.

be fixed by adding leading and trailing zero-vectors to $\vec{A}^{l_1} \cdot \vec{B}^{l_2}$. With this approach, we are able to compute the point-wise product between Fourier coefficient vectors as follows: Let $\vec{I}_{l_1,l_2}$ and $\vec{J}_{l_1,l_2}$ be vectors with $(l_2 - l_1)^2$ and $(L_1 + L_2 - 1)^2 - (l_2 + l_1)^2$ zero entries, respectively. Then we can write the product between two Fourier coefficient vectors $\vec{A}^{L_1} \cdot \vec{B}^{L_2}$ as

$$\vec{A}^{L_1} \cdot \vec{B}^{L_2} = \sum_{l_1=0}^{L_1-1} \sum_{l_2=0}^{L_2-1} \left[\vec{I}_{l_1,l_2} \oplus \left(\vec{A}^{l_1} \cdot \vec{B}^{l_2} \right) \oplus \vec{J}_{l_1,l_2} \right]. \tag{3.54}$$

The resulting Fourier coefficient of $\vec{A}^{L_1} \cdot \vec{B}^{L_2}$ has a maximum band of $L = L_1 + L_2 - 1$.

3.6 Real Spherical Harmonics

Up to this point we discussed the general case of complex-valued functions defined on the rotation group $SO(3)$ or the sphere S^2. However, in our applications (e.g. the visual 3D compass, section 5.2) we only use real-valued functions as input. By splitting equation (3.47) into its real and imaginary part, we can reformulate the SH Y_m^l as **real spherical harmonics** (RSH) denoted by y_m^l:

Definition 3.15 (Real Spherical Harmonics)**.** *The RSH $y_m^l : S^2 \to \mathbb{R}$ are defined as*

$$y_m^l(\vartheta, \varphi) = \begin{cases} \sqrt{2} K_m^l \cos(m\varphi) P_m^l(\cos\vartheta), & m > 0; \\ \sqrt{2} K_m^l \sin(-m\varphi) P_{-m}^l(\cos\vartheta), & m < 0; \\ K_0^l P_0^l(\cos\vartheta), & m = 0. \end{cases} \tag{3.55}$$

In the following we use the same notations as for the spherical harmonics, e.g. the vector $\vec{y}^l$ contains all RSH of band l.

In this section we show that all calculations we discussed for SH can be reformulated to work with RSH using appropriate transformation matrices (definitions 3.16, 3.17, and 3.18). Table 3.1 gives an overview of the real and complex definitions.

Analogously to the SH, the RSH form an orthonormal basis. Furthermore, the Peter-Weyl theorem for SH (theorem 3.10) can be applied by replacing Y_m^l with y_m^l. Contrary to the SH, we denote the corresponding Fourier coefficients of the RSH by a_m^l instead of A_m^l. Most important, by using RSH, the coefficients a_m^l for a real-valued function f are *real-valued*. This simplifies the implementation and halves the computational time for algorithms which work with real-valued instead of complex-valued functions. As usual in mathematics, we exploit the complex numbers in order to deal with RSH[4], especially we use the complex-valued Wigner-D/Clebsch-Gordan matrices to calculate the **real Wigner-D matrices/Clebsch-Gordan matrices**. In this case, the term *real* does not imply that these matrices are real-valued (even though this is true), but are defined for use in the basis of RSH. We now define the **transformation matrix** which maps Fourier coefficients calculated in the basis of (complex-valued) SH into Fourier coefficients calculated in the basis of (real-valued) RSH.

[4] *'The shortest path between two truths in the real domain passes through the complex domain.'* — Jacques Hadamard

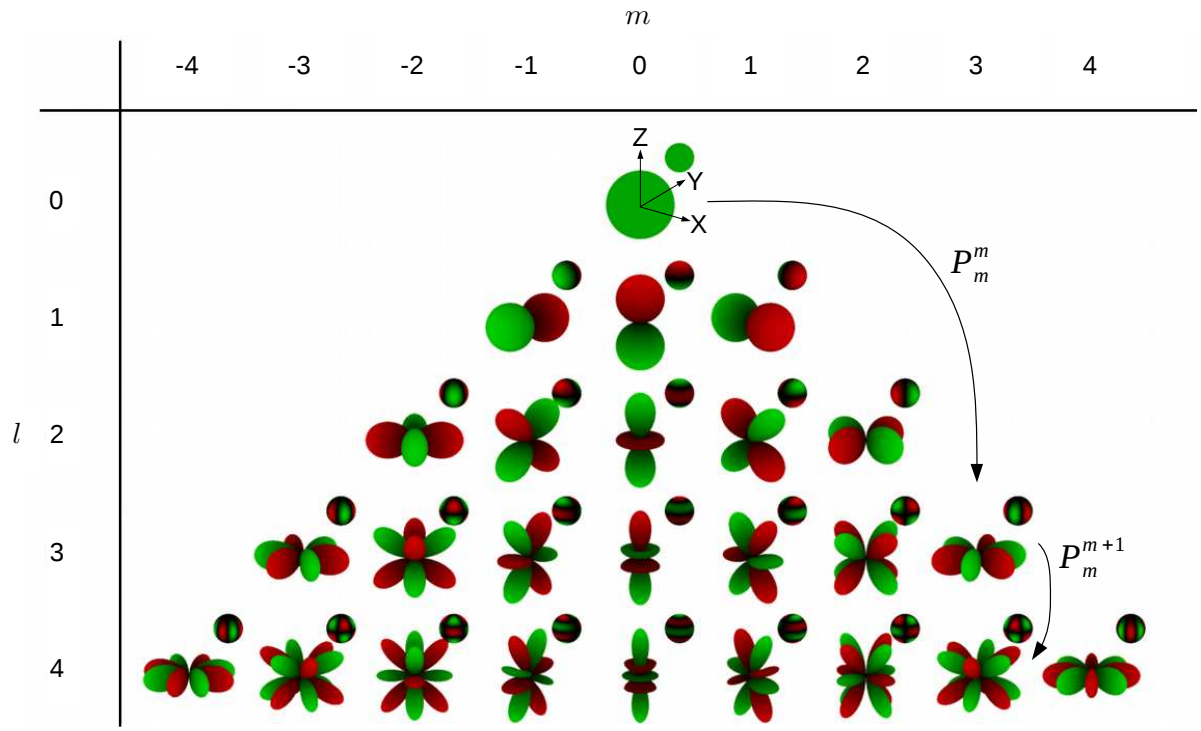


Figure 3.3: The RSH are shown vertically sorted by their band l and horizontally sorted by their index $m \in \{-l, \dots, l\}$. Each plot shows the values of y_m^l on the sphere S^2 by color (green positive, red negative) and shape (small extents correspond to small values). The arrows indicate the recurrence relations established by equation (3.64) (i.e. P_m^m) and the equations (3.65) and (3.66) (i.e. P_m^{m+1}).

Definition 3.16 (Transformation Matrix)**.** *The basis transform from SH to RSH can be expressed in matrix form as (Blanco et al., 1997):*

$$\mathbf{T}_l = \frac{1}{\sqrt{2}} \begin{pmatrix} i & 0 & \cdots & 0 & \cdots & 0 & -i(-1)^l \\ 0 & i & \cdots & 0 & \cdots & -i(-1)^{l-1} & 0 \\ \vdots & \vdots & \ddots & \vdots & \iddots & \vdots & \vdots \\ 0 & 0 & \cdots & \sqrt{2} & \cdots & 0 & 0 \\ \vdots & \vdots & \iddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \cdots & 0 & \cdots & (-1)^{l-1} & 0 \\ 1 & 0 & \cdots & 0 & \cdots & 0 & (-1)^l \end{pmatrix} \tag{3.56}$$

It fulfills $\vec{y}^l = \mathbf{T}_l \vec{Y}^l$ and is unitary, i.e. $\mathbf{T}_l^{-1} = \mathbf{T}_l^\dagger$. Therefore we also have $\mathbf{T}_l^\dagger \vec{y}^l = \vec{Y}^l$.

Let f be a function on the unit sphere with Fourier coefficient vector $\vec{A}^l$. We use the transformation matrix $\mathbf{T}_l$ to transform a Fourier coefficient vector $\vec{A}^l$ in the basis of SH to a Fourier coefficient vector $\vec{a}^l$ in the basis of RSH.

$$f = \sum_{l=0}^{L} \left[\vec{Y}^l\right]^T \vec{A}^l \overset{\text{Def. 3.16}}{=} \sum_{l=0}^{L} \left[\mathbf{T}_l^\dagger \vec{y}^l\right]^T \vec{A}^l = \sum_{l=0}^{L} \left[\vec{y}^l\right]^T \bar{\mathbf{T}}_l \vec{A}^l \tag{3.57}$$

By defining the Fourier coefficient vector in the basis of RSH as

$$\vec{a}^l := \bar{\mathbf{T}}_l \vec{A}^l \quad \text{we have} \quad f = \sum_{l=0}^{L} \left[\vec{y}^l\right]^T \vec{a}^l \quad \text{and} \quad \vec{A}^l = \left[\mathbf{T}_l\right]^T \vec{a}^l. \tag{3.58}$$

Furthermore, we have that for a real-valued function f defined on the unit sphere, the Fourier coefficient vector $\vec{a}^l$ is also real-valued (Blanco et al., 1997).

In section 3.5.3 we showed that a function f can be rotated in the basis of SH by performing a band-wise matrix multiplication of its Fourier coefficient vector with a Wigner-D matrix. This allows us to perform rotations in the basis of RSH in three steps: First, we transform the Fourier coefficient vector $\vec{a}^l$ from the basis of RSH into the basis of SH. Second, we rotate the function in the basis of SH. Third, we transform the result back into the basis of RSH. Summarizing this concatenation of transformations into a single transformation, we obtain the **real Wigner-D matrices**:

Definition 3.17 (Real Wigner-D matrix)**.** *The real Wigner-D matrices are defined as*

$$\mathbf{d}^l := \bar{\mathbf{T}}_l \mathbf{D}^l \mathbf{T}_l^T \in \mathrm{Mat}(2l+1, \mathbb{R}), \tag{3.59}$$

where $\mathbf{T}_l$ is the transformation matrix defined in definition 3.16. The matrices $\mathbf{d}^l$ are real-valued as can be seen in theorem 3.21.

Therefore we have, analogously to definition 3.12, that the rotation of a function f can be calculated directly in the basis of RSH as

$$\mathbf{R} \circ \vec{a}^l = \mathbf{d}^l(\mathbf{R}^T)\vec{a}^l \tag{3.60}$$

The advantage of using real Wigner-D matrices is that the expense of applying transformation matrices can be factored out into the precalculation of $\mathbf{d}^l$.

As for rotations in the basis of RSH, we need the point-wise product in the basis of RSH. Similar to the real Wigner-D matrices, we propose **real Clebsch-Gordan matrices** (definition 3.18) to compute point-wise products and the bispectrum directly in the basis of RSH. Since the derivation of the bispectrum in Kakarala (1992) and Kakarala and Mao (2010) only deals with the case of SH, the real Clebsch-Gordan matrices allow us to reformulate the bispectrum directly in the basis of RSH (section 3.9.4).

Definition 3.18 (Real Clebsch-Gordan Matrix)**.** *Let $\mathbf{C}_{l_1,l_2}$ be a Clebsch-Gordan matrix with $l_1, l_2 \in \mathbb{N}, l_2 \geq l_1$ and $\mathbf{T}_i$ be transformation matrices. Then we define*

$$\mathbf{c}_{l_1,l_2} := \overline{[\mathbf{T}_{l_1} \otimes \mathbf{T}_{l_2}]} \mathbf{C}_{l_1,l_2} \left[\bigoplus_{i=l_2-l_1}^{l_2+l_1} \mathbf{T}_i \right]^T \in \mathrm{Mat}(n, \mathbb{R}), \tag{3.61}$$

with $n = (2l_1+1)(2l_2+1)$ as real Clebsch-Gordan matrix.

Additionally to the Clebsch-Gordan matrices $\mathbf{C}_{l_1,l_2}$, we defined in section 3.5.5 — by scaling the coefficients of the Clebsch-Gordan matrix — the coupling matrices $\tilde{\mathbf{C}}_{l_1,l_2}$. Analogously to the real Clebsch-Gordan matrices, we define the **real coupling matrix** as follows:

$$\tilde{\mathbf{c}}_{l_1,l_2} := \overline{[\mathbf{T}_{l_1} \otimes \mathbf{T}_{l_2}]} \tilde{\mathbf{C}}_{l_1,l_2} \left[\bigoplus_{i=l_2-l_1}^{l_2+l_1} \mathbf{T}_i \right]^T \in \mathrm{Mat}(n, \mathbb{R}) \tag{3.62}$$

Now it is a simple task to show that the point-wise product for SH from section 3.5.5 can directly be used to calculate the point-wise product for RSH.

Theorem 3.19 (Real Point-Wise Product)**.** *Let f_1, f_2 on S^2 be real-valued functions with Fourier coefficients $\vec{a}_1^{L_1}$ and $\vec{a}_2^{L_2}$. Then the point-wise product between band l_1 of $\vec{a}_1^{L_1}$ and band l_2 of $\vec{a}_2^{L_2}$ is given by*

$$\vec{a}_1^{l_1} \cdot \vec{a}_2^{l_2} = \tilde{\mathbf{c}}_{l_1,l_2}^{\dagger} \left[\vec{a}_1^{l_1} \otimes \vec{a}_2^{l_2} \right]. \tag{3.63}$$

Proof. The proof can be found in appendix A.3. □

As for real Wigner-D matrices, for real Clebsch-Gordan matrices the expense of applying transformation matrices can be factored out into the precalculation of $\tilde{\mathbf{c}}^l$. We use the point-wise product for RSH in section 3.9.3 to derive the bispectrum in the basis of RSH and in section 3.10.3 to calculate the weighted integral squared error. The latter is excessively used for the visual 3D compass in chapter 5 to calculate the rotational offset between two functions.

3.6.1 Recurrence Relations

Up to this point we only used explicit formulas to calculate the values of SH and Wigner-D matrices. In this section we present different recursive algorithms to calculate the SH and RSH as well as the real Wigner-D matrices. The following theorem allows an effective calculation of the associated Legendre polynomials required to calculate SH and RSH.

Theorem 3.20. *The associated Legendre polynomial $P_m^l(x)$ with $m, n \in \mathbb{N}$ can be calculated using the following recurrence relations*

$$P_m^m(x) = (-1)^m(2m-1)!!(1-x^2)^{\frac{m}{2}} \tag{3.64}$$

$$P_m^{m+1}(x) = x(2m+1)P_m^m(x) \tag{3.65}$$

$$P_m^l(x) = \frac{x(2l-1)P_m^{l-1}(x) - (l+m-1)P_m^{l-2}(x)}{(l-m)}, \tag{3.66}$$

where $n!!$ denotes the combinatoric ***double factorial****.*

Proof. See Press et al. (1992). □

Regarding figure 3.3, equation (3.64) corresponds to moving diagonally downward-right in the diagram, while equations (3.65) and (3.66) correspond to moving vertically downwards. Using the equations from theorem 3.20, an arbitrary associated Legendre polynomial $P_m^l(x)$ with $m \geq 0$ can be evaluated as follows:

1. Apply equation (3.64) repeatedly until the value $P_m^m(x)$ is obtained.
2. If $m < l$, apply equation (3.65) repeatedly until the desired value $P_m^l(x)$ is obtained.

While equations (3.64) and (3.65) are sufficient to calculate the value $P_m^l(x)$, equation (3.66) is numerically more stable and should be preferred as soon as (3.65) would be called more than once.

The calculation of real Wigner-D matrices is more complicated. Several formulas can be found in literature, for example in Ivanic and Ruedenberg (1996), Blanco et al. (1997), and Choi et al. (1999).

Theorem 3.21. *Let $\mathbf{R} \in SO(3)$ be a rotation matrix with corresponding real Wigner-D matrix $\mathbf{d}^l(\mathbf{R})$. Then the entries of $\mathbf{d}^l(\mathbf{R})$ can be calculated as*

$$d_{mn}^l(\mathbf{R}) = u_{mn}^l U_{mn}^l + v_{mn}^l V_{mn}^l + w_{mn}^l W_{mn}^l, \tag{3.67}$$

where the functions u_{mn}^l, v_{mn}^l, and w_{mn}^l are given by

	$\lvert n\rvert < l$	$\lvert n\rvert = l$
u_{mn}^l	$\sqrt{\frac{(l+m)(l-m)}{(l+n)(l-n)}}$	$\sqrt{\frac{(l+m)(l-m)}{(2l)(2l-1)}}$
v_{mn}^l	$\frac{1}{2}(1-2\delta_{m,0})\sqrt{\frac{(1+\delta_{m,0})(l+\lvert m\rvert-1)(l+\lvert m\rvert)}{(l+n)(l-n)}}$	$\frac{1}{2}(1-2\delta_{m,0})\sqrt{\frac{(1+\delta_{m,0})(l+\lvert m\rvert-1)(l+\lvert m\rvert)}{(2l)(2l-1)}}$
w_{mn}^l	$-\frac{1}{2}(1-\delta_{m,0})\sqrt{\frac{(l-\lvert m\rvert-1)(l-\lvert m\rvert)}{(l+n)(l-n)}}$	$-\frac{1}{2}(1-\delta_{m,0})\sqrt{\frac{(l-\lvert m\rvert-1)(l-\lvert m\rvert)}{(2l)(2l-1)}}$

(3.68)

and the functions U_{mn}^l, V_{mn}^l, and W_{mn}^l are given by

	$m = 0$	$m > 0$	$m < 0$
U_{mn}^l	${}_0P_{0,n}^l$	${}_0P_{m,n}^l$	${}_0P_{m,n}^l$
V_{mn}^l	${}_1P_{1,n}^l + {}_{-1}P_{-1,n}^l$	${}_1P_{m-1,n}^l\xi_1 - {}_{-1}P_{-m+1,n}^l\zeta_1$	${}_1P_{m+1,n}^l\zeta_{-1} + {}_{-1}P_{-m-1,n}^l\xi_{-1}$
W_{mn}^l	0	${}_1P_{m+1,n}^l + {}_{-1}P_{-m-1,n}^l$	${}_1P_{m-1,n}^l - {}_{-1}P_{-m+1,n}^l$

(3.69)

with $\xi_a := \sqrt{1+\delta_{m,a}}$ *and* $\zeta_a := 1-\delta_{m,a}$. *Finally, the function* ${}_iP^l_{m,n}$ *yielding the recurrence relation is given by:*

	$\lvert n\rvert < l$	$n = l$	$n = -l$
${}_iP^l_{m,n}$	$d^1_{i,0}d^{l-1}_{m,n}$	$d^1_{i,1}d^{l-1}_{m,l-1} - d^1_{i,-1}d^{l-1}_{m,-l+1}$	$d^1_{i,1}d^{l-1}_{m,-l+1} + d^1_{i,-1}d^{l-1}_{m,l-1}$

(3.70)

The matrix $\mathbf{d}^1(\mathbf{R})$ *can directly be calculated from* $\mathbf{R}$ *using a ZYZ parameterization with angles* (γ, β, α) *as*

$$\mathbf{d}^1(\gamma,\beta,\alpha) = \begin{pmatrix} \cos\alpha\cos\gamma \\ -\sin\alpha\cos\beta\sin\gamma & \sin\alpha\sin\beta & \cos\alpha\sin\gamma \\ & & +\sin\alpha\cos\beta\cos\gamma \\ \sin\beta\sin\gamma & \cos\beta & -\sin\beta\cos\gamma \\ -\cos\alpha\cos\beta\sin\gamma & & \cos\alpha\cos\beta\cos\gamma \\ -\sin\alpha\cos\gamma & \cos\alpha\sin\beta & \text{- }\sin\alpha\sin\gamma \end{pmatrix}, \tag{3.71}$$

providing the starting point of the recurrence relation.

Proof. See Ivanic and Ruedenberg (1996); additions and corrections to their first proof can be found in Ivanic and Ruedenberg (1998). Note that in the additions and corrections a sign error was introduced in the term V^l_{mn} with $m < 0$ which has been corrected in this theorem. □

The implementation of the recurrence relations stated in the theorems 3.20 and 3.21 is straightforward. While these algorithms are merely bearable for calculations in real time, they can be used to precalculate lookup tables. The visual 3D compass (chapter 5) is designed to subsequently rotate a function for a predefined set of rotations $\mathbf{R} \in SO(3)$. In this case, we can precalculate all necessary rotations. By choosing a suitable rotation parameterization, mostly Z-axis rotations are used. A single Z-axis rotation is extremely fast such that the calculation time for each single search step is small.

In contrast to Z-axis rotations, arbitrary rotations are comparably slow (e.g. X/Y-axis rotations, section 3.7). In applications where rotations cannot be precalculated, the formulas presented in theorem 3.21 are computationally expensive. To calculate arbitrary rotations on the fly, specialized algorithms can be found, e.g. Lessig et al. (2012) (approximative algorithm using kernels), Nowrouzezahrai et al. (2012) (using a decomposition in zonal RSH), and Gimbutas and Greengard (2009) (specialized for high bands and parallel computing). Since our implementation of the visual 3D compass mainly relies on computationally cheap Z-axis rotations, it is unnecessary to use these sophisticated algorithms.

3.6.2 Symmetries

In practical applications, often not the complete function f defined on the sphere S^2 is known; for example if f is the input of a hemispherical fish-eye lens (example 3.4). In this section we derive a **hemispherical continuation** which can be used to automatically fill in the missing information of the lower hemisphere if only information of the upper hemisphere is available. We use the hemispherical continuation in section 3.10.3 to fill the panoramic image captured by a hemispherical camera. To obtain these symmetries, we first define the following operations for RSH to obtain symmetries between the upper and lower hemisphere:

$$\begin{aligned} \kappa_R(y^l_m(\vartheta,\varphi)) &= y^l_m(\vartheta,\varphi+\pi) && \text{(azimuthal rotation)} \\ \kappa_M(y^l_m(\vartheta,\varphi)) &= y^l_m(\pi-\vartheta,\varphi) && \text{(mirroring)} \\ \kappa_N(y^l_m(\vartheta,\varphi)) &= -y^l_m(\vartheta,\varphi) && \text{(negation)} \end{aligned} \tag{3.72}$$

The geometrical interpretation of κ_R and κ_M can be seen in figure 3.4. The idea is the following: We want to find subsets of the RSH basis functions which are invariant under κ_M.

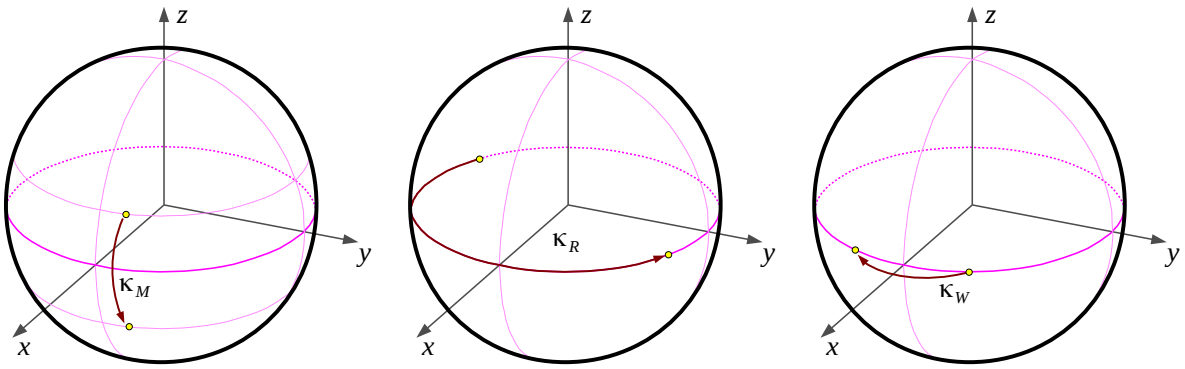


Figure 3.4: Left: The operation κ_M mirrors a point on the upper hemisphere to the lower hemisphere. Middle: The operation κ_R rotates a point around the equator (or on a parallel orbit) to the opposite side of the sphere. Right: The operation κ_W mirrors a point by the plane spanned by the X-axis and Z-axis (needed for the proof of theorem 3.26).

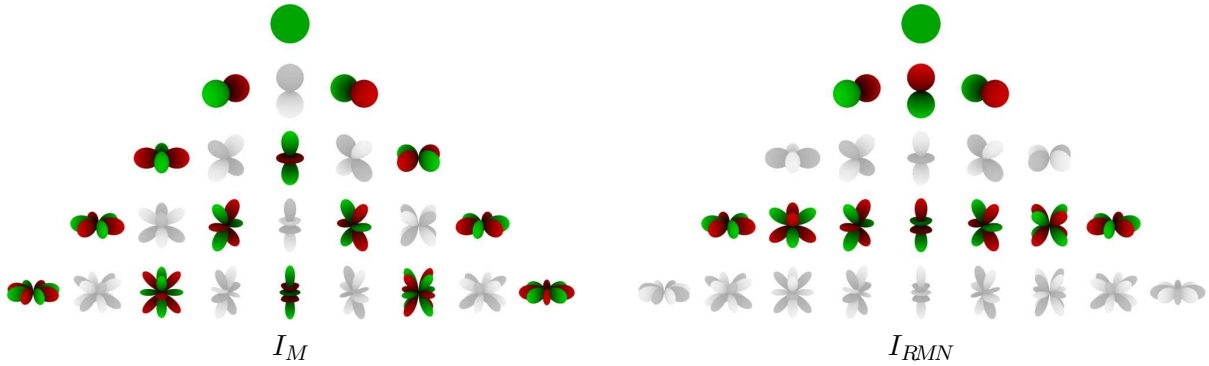


Figure 3.5: The subsets I_M and I_{RMN} (colored) of the RSH are shown for the first five bands (compare figure 3.3). Each subset is invariant to the operation κ_M (left) and κ_{RMN} (right), respectively, and was determined using theorem 3.22.

Since κ_M automatically establishes a connection between the upper and the lower hemisphere, we are able to automatically fill in the missing information in the lower hemisphere with available information of the upper hemisphere. For example we could simply fill the lower hemisphere by mirroring each pixel of the upper hemisphere to the lower hemisphere (κ_M). This is a trivial operation in the spatial domain (e.g. on the camera image), however the question arises if the complete set of SH is required to represent this image or if — due to the symmetries of the SH — only a subset of the SH is required. The following theorem shows which subsets of the SH are required to represent images which are invariant under any combination of the operations κ_R, κ_M, and κ_N.

Theorem 3.22 (Symmetries)**.** *Let y_m^l be a RSH with $l \geq 1$, then it is invariant under the following operations:*

	l even	*l odd*
m even	κ_R, κ_M, κ_{RM}	κ_R, κ_{MN}, κ_{RMN}
m odd	κ_{RM}, κ_{RN}, κ_{MN}	κ_M, κ_{RN}, κ_{RMN}

Except for the operation κ_N alone, $y_0^0(\vartheta, \varphi)$ is invariant under all combinations of κ_R, κ_M, and κ_N.

Proof. The proof can be found in appendix A.4. □

There is a total of 8 possible combinations (concatenations) of the operations $\kappa_R, \kappa_M, \kappa_N$. Since only κ_M establish a relation between the points in the upper and the lower hemisphere, only four of these combinations are suitable for hemispherical continuation (κ_M, κ_{RM}, κ_{MN} and κ_{RMN}). Using our symmetry theorem 3.22 we can now select a subset of the RSH which is invariant to one of these combinations and denote it by

$$I_X := \left\{ y_m^l \;\middle|\; \kappa_X(y_m^l) = y_m^l \right\}, \quad X \in \{M, MN, RM, RMN\}. \tag{3.73}$$

For visualization, the subsets I_M and I_{RMN} are shown in figure 3.5.

Each of these combinations allows us to calculate the Fourier transform on the subset I_X using points of the upper hemisphere only[5]. We show this explicitly for the case I_M (the remaining cases are calculated analogously) by using that $y_m^l(\vartheta, \varphi) = y_m^l(\pi - \vartheta, \varphi)$ for all $y_m^l \in I_M$. We calculate the Fourier coefficient a_m^l using theorem 3.10 (adapted for RSH, compare section 3.6) as before but split the integral into two integrals, one for the upper and one for the lower hemisphere:

$$a_m^l = \int_0^{2\pi} \int_0^{\pi} f(\vartheta, \varphi) y_m^l(\vartheta, \varphi) \sin\vartheta \, d\vartheta d\varphi \tag{3.74}$$

$$= \int_0^{2\pi} \int_0^{\frac{\pi}{2}} f(\vartheta, \varphi) y_m^l(\vartheta, \varphi) \sin\vartheta \, d\vartheta d\varphi + \int_0^{2\pi} \int_{\frac{\pi}{2}}^{\pi} f(\vartheta, \varphi) y_m^l(\vartheta, \varphi) \sin\vartheta \, d\vartheta d\varphi \tag{3.75}$$

By applying integration by substitution ($\vartheta' = \pi - \vartheta$) and use that $y_m^l(\pi - \vartheta, \varphi) = y_m^l(\vartheta, \varphi)$ (i.e. $y_m^l \in I_M$) we obtain:

$$= \int_0^{2\pi} \int_0^{\frac{\pi}{2}} f(\vartheta, \varphi) y_m^l(\vartheta, \varphi) \sin\vartheta \, d\vartheta d\varphi - \int_0^{2\pi} \int_{\frac{\pi}{2}}^{0} f(\pi - \vartheta, \varphi) y_m^l(\pi - \vartheta, \varphi) \sin(\pi - \vartheta) d\vartheta d\varphi$$

$$= \int_0^{2\pi} \int_0^{\frac{\pi}{2}} \left[f(\vartheta, \varphi) + f(\pi - \vartheta, \varphi) \right] y_m^l(\vartheta, \varphi) \sin\vartheta \, d\vartheta d\varphi \tag{3.76}$$

Since f should only be expressed by SH which are invariant under κ_M, we also assume that f is invariant under κ_M, i.e. $f \in I_M$ and therefore $f(\pi - \vartheta, \varphi) = f(\vartheta, \varphi)$, and we finally obtain

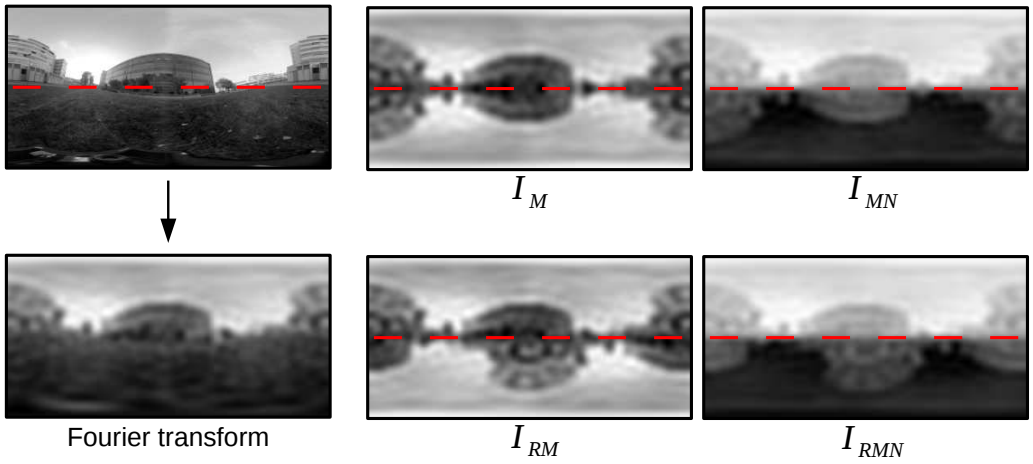
$$= 2 \int_0^{2\pi} \int_0^{\frac{\pi}{2}} f(\vartheta, \varphi) y_m^l(\vartheta, \varphi) \sin\vartheta \, d\vartheta d\varphi. \tag{3.77}$$

As can be seen, equation (3.74) allows us to perform the Fourier transform on the upper hemisphere only while simultaneously obtaining a hemispherical continuation for the lower hemisphere (example 3.6). Moreover, the computation time is halved if only the upper hemisphere is Fourier transformed.

The subsets I_{RM} and I_{RMN} of the RSH basis functions are of special interest: From our symmetry theorem 3.22 we have that $I_{RM} = \left\{ y_m^l \;\middle|\; l \text{ even} \right\}$ and $I_{RMN} = \left\{ y_m^l \;\middle|\; l \text{ odd} \vee l = 0 \right\}$. For a function f — which is Fourier-transformed using one of the sets I_{RM} or I_{RMN} — this means that half of the Fourier coefficient vectors $\vec{a}^l$ (depending on the band l) are zero. Calculations which are performed band-wise benefit from this property: For example, rotations (section 3.7) are only computed for odd (I_{RMN}) or even (I_{RM}) bands. As a consequence, the computation time can approximately be halved using hemispherical continuation.

[5] Note that for each subset $I_{M,MN,RM,RMN}$ only panoramic images can be Fourier transformed which are assumed to be invariant under the corresponding operation. For hemispherical panoramic images, information in the lower hemisphere is obviously lost, however using fish-eye objectives with an opening angle of 180° this information is not available in the first place.

Example 3.6: Hemispherical Continuation



Hemispherical continuation using the subsets I_M, I_{MN}, I_{RM}, and I_{RMN} and only the upper hemisphere of the panoramic image. For comparison the Fourier transform of the complete panoramic image is shown Note that the visualizations of I_{MN} and I_{RMN} are normalized: Positive values are white and negative values are black.

3.7 Rotations

In section 3.6 we showed that rotations in the basis of RSH are calculated band-wise:

$$\mathbf{R} \circ \vec{a}^l = \mathbf{d}^l(\mathbf{R}^T)\vec{a}^l \tag{3.78}$$

In this section, we examine the sparsity of the Wigner-D rotation matrices $\mathbf{d}^l(\mathbf{R})$ for arbitrary as well as axis-specific rotations. Visualizations of rotations around the X-axis and Y-axis are shown in example 3.7.

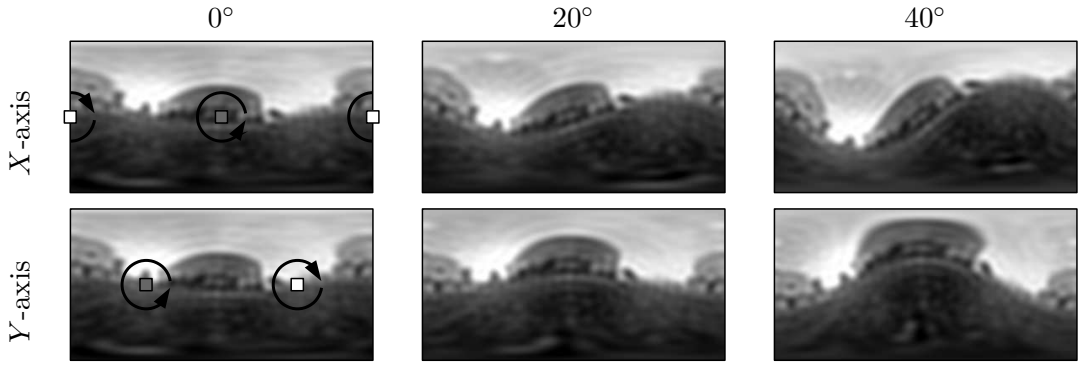
We have from equation (3.14) that real Wigner-D matrices $\mathbf{d}^L$ consist of block matrices $\mathbf{d}^L$ and have the following form:

$$\left(\begin{array}{c|ccc|ccccc|c} X & & & & & & & & & \\ \hline & X & X & X & & & & & & \\ & X & X & X & & & & & & \\ & X & X & X & & & & & & \\ \hline & & & & X & X & X & X & X & \\ & & & & X & X & X & X & X & \\ & & & & X & X & X & X & X & \\ & & & & X & X & X & X & X & \\ & & & & X & X & X & X & X & \\ \hline & & & & & & & & & \ddots \end{array}\right) \tag{3.79}$$

They contain only $\sum_{l=0}^{L-1}(2l+1)^2 = \frac{4L^3-L}{3}$ non-zero entries (marked by 'X') instead of L^4.

In the following we present several theorems which reveal more detailed sparsity information for rotation matrices around the X-, Y-, and Z-axis in the basis of RSH. These sparsity information can be used to implement rotations more efficiently by exploiting the structure of the single matrices $\mathbf{d}^l(\mathbf{R})$, e.g. rotations around a single axis can be implemented more efficiently. Furthermore, we can use them to find a rotation parameterization — especially useful for the visual 3D compass, section 5.2.2) — with a low computation time. In the following theorems 3.23, 3.24, and 3.25

Example 3.7: Rotating Spherical Harmonics



Theorem 3.21 states formulas to compute arbitrary rotations in the basis of SH and RSH. The images show the rotation around the X-axis and Y-axis (top and bottom, respectively) with an increasing rotation angle of 0°, 20°, and 40° (from left to right). The white squares point in the direction of the rotation axis (the grey squares point in the opposite direction) and the arrows indicate the direction of the rotation.

we derive sparsity relations for real Wigner-D matrices for rotations around the X-, Y-, and Z-axis. Note that it is sufficient to show that $d^l_{mn}(\mathbf{R}_{\vec{v},\alpha}) = 0$ for all rotations $\alpha \in \mathbb{R}$ in order to show that also $d^l_{nm}(\mathbf{R}_{\vec{v},\alpha}) = 0$. This is a direct consequence of lemma 3.11 (ii) which states that $d^l_{mn}(\mathbf{R}_{\vec{v},\alpha}) = d^l_{nm}(\mathbf{R}_{\vec{v},-\alpha})$. Moreover, this relation allows us to calculate the Wigner-D matrices for pairs of rotation angles $\pm\alpha$ simultaneously.

Theorem 3.23 (Z-Axis Rotations)**.** *Let $\mathbf{R} := \mathbf{R}_{Z,\alpha} \in SO(3)$ be a rotation matrix around the Z-axis by angle α. Then a rotated RSH has the form*

$$y^l_m(\mathbf{R}\vec{p}) = \begin{cases} \cos(m\alpha)y^l_m(\vec{p}) + \sin(m\alpha)y^l_{-m}(\vec{p}), & m > 0 \\ \cos(m\alpha)y^l_m(\vec{p}) - \sin(m\alpha)y^l_{-m}(\vec{p}), & m < 0 \\ y^l_m(\vec{p}), & m = 0. \end{cases} \tag{3.80}$$

For the corresponding real Wigner-D matrix $\mathbf{d}^l := \mathbf{d}^l(\mathbf{R})$ of band l we have

$$d^l_{mn} = \begin{cases} \cos(|m|\alpha), & n = m \\ \sin(n\alpha), & n = -m \\ 0, & |m| \neq |n|, \end{cases} \tag{3.81}$$

for all $\mathbf{R} \in SO(3)$.

Proof. The proof can be found in appendix A.5.1. □

In contrast to Z-axis rotations, where it is a simple task to calculate explicit formulas for rotated RSH $y^l_m(\mathbf{R}\vec{p})$ directly from their definition 3.9, this is not the case for X/Y-axis rotations (compare theorem 3.21).

Theorem 3.24 (Y-Axis Rotations)**.** *Let $\mathbf{R} := \mathbf{R}_{Y,\alpha} \in SO(3)$ be a rotation matrix around the Y-axis by angle α and $\mathbf{d}^l := \mathbf{d}^l(\mathbf{R})$ the corresponding real Wigner-D matrix of band l. Then it holds*

$$d^l_{mn} = \begin{cases} d^l_{nm}, & m+n \text{ is even} \\ -d^l_{nm}, & m+n \text{ is odd.} \end{cases} \tag{3.82}$$

Furthermore we have

$$m \geq 0 \text{ and } n < 0 \;\Rightarrow\; \mathbf{d}^l_{mn} = \mathbf{d}^l_{nm} = 0. \tag{3.83}$$

Proof. The proof can be found in appendix A.5.2. □

Theorem 3.25 (X-Axis Rotations)**.** *Let* $\mathbf{R} := \mathbf{R}_{X,\alpha} \in SO(3)$ *be a rotation matrix around the* X*-axis by angle* α *and* $\mathbf{d}^l := \mathbf{d}^l(\mathbf{R})$ *the corresponding real Wigner-D matrix of band* l*. Then it holds*

$$d^l_{mn} = \begin{cases} d^l_{nm}, & \text{if } m+n \text{ is even} \\ -d^l_{nm}, & \text{if } m+n \text{ is odd} \end{cases} \tag{3.84}$$

Furthermore, if one of the following conditions holds

	$m < 0$	$m \geq 0$
$n < 0$	$m+n$ *odd*	$m+n$ *even*
$n \geq 0$	$m+n$ *even*	$m+n$ *odd*

we have that $d^l_{mn} = 0$*.*

Proof. The proof can be found in appendix A.5.3. □

The theorems derived are valid for arbitrary rotations around a single axis. As we will see later (corollary 3.27), Wigner-D matrices for rotations around the Y/X-axes have — in comparison to rotations around the Z-axis — many non-zero elements, increasing the computational costs. A common strategy (e.g. Green (2003)) is to replace these rotations by rotations of the form

$$\mathbf{R}_{Y,\beta} = \mathbf{R}_{X,90°}\mathbf{R}_{Z,\beta}\mathbf{R}_{X,-90°} \quad \text{and} \quad \mathbf{R}_{X,\gamma} = \mathbf{R}_{Y,-90°}\mathbf{R}_{Z,\gamma}\mathbf{R}_{Y,90°} \tag{3.85}$$

such that only X/Y-axes rotations of $\pm 90°$ would be necessary to perform arbitrary rotations. Fortunately, rotations of $\pm 90°$ are highly sparse as we show in the following theorem.

Theorem 3.26. *Let* $\mathbf{R} \in SO(3)$ *be a rotation matrix around the* X*,*Y*, or* Z*-axis by* $\pm 90°$ *and* $\mathbf{d}^l := \mathbf{d}^l(\mathbf{R})$ *the corresponding real Wigner-D matrix of band* l*. Depending on the rotation axis we have* $d^l_{mn} = 0$ *if one of the following conditions is fulfilled:*

X-axis:
- *(i)* $n < 0$ and $l+m$ *even*
- *(ii)* $n \geq 0$ and $l+m$ *odd*

Y-axis:
- *(i)* $n < 0$ and $l+m+n$ *even*
- *(ii)* $n \geq 0$ and $l+m+n$ *odd*

Z-axis:
- *(i)* $mn < 0$ and m *even*
- *(ii)* $mn \geq 0$ and m *odd*

These conditions hold additionally to the results of the theorems 3.23, 3.24, and 3.25.

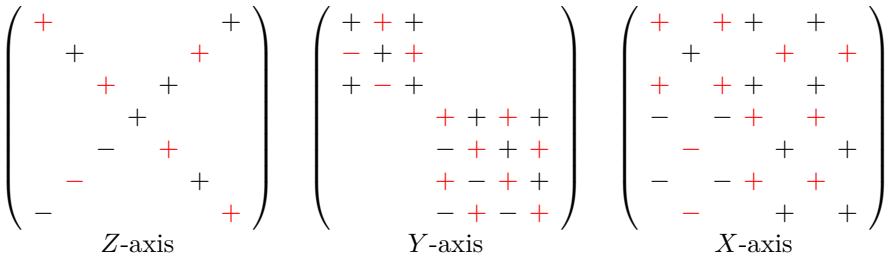
Proof. The proof can be found in appendix A.5.4. □

Corollary 3.27. *Let* $\mathbf{R} \in SO(3)$ *be a rotation matrix,* $\mathbf{d}^l := \mathbf{d}^l(\mathbf{R})$ *be the corresponding real Wigner-D matrix of band* l*, and* $\mathbf{d}^L := \mathbf{d}^L(\mathbf{R})$ *be the real Wigner-D matrices for* L *bands. Then the maximal number of non-zero entries, based on the type of the rotation matrix* $\mathbf{R}$*, is given by:*

	$\mathbf{R}$	$\mathbf{R}_{X,\alpha}$,$\mathbf{R}_{Y,\alpha}$	$\mathbf{R}_{Z,\alpha}$	$\mathbf{R}_{X,90°}$,$\mathbf{R}_{Y,90°}$	$\mathbf{R}_{Z,90°}$
$\mathbf{d}^l$	$(2l+1)^2$	$l^2+(l+1)^2$	$4l+1$	l^2+l+1	$2l+1$
$\mathbf{d}^L$	$\frac{1}{3}(4L^3-L)$	$\frac{1}{3}(2L^3+L)$	$2L^2-L$	$\frac{1}{3}(L^3+2L)$	L^2

Proof. The corollary directly follows from the theorems 3.23, 3.24, 3.25, and 3.26: For each case we know the sparsity relations of $\mathbf{d}^l(\mathbf{R})$ from the corresponding theorems and can simply count (band-wise) the number of non-zero entries for each $\mathbf{d}^l(\mathbf{R})$. Afterwards, we sum up the number of non-zero entries over all bands to obtain the total number of non-zero entries for each $\mathbf{d}^L(\mathbf{R})$. Since both tasks are simple, but tedious and non-constructive, we do not explicitly present them. □

Example 3.8: Sparsity of Rotation Matrices



Z-axis $\quad$ Y-axis $\quad$ X-axis

The real Wigner-D matrices $\mathbf{d}^3$ for the three basic rotations around the Z, Y, and X-axis are presented. Each figure shows three properties: First, zero entries are represented by empty cells in the matrix. Second, a plus in the lower triangle indicates that the transposed element is the same as the original element. Contrary, a minus indicates that the transposed element is the negative of the original element. Note that the signs do *do not* state if the entry itself is positive or negative. Third, the red markings denote the entries which vanish for rotations around $\pm 90°$.

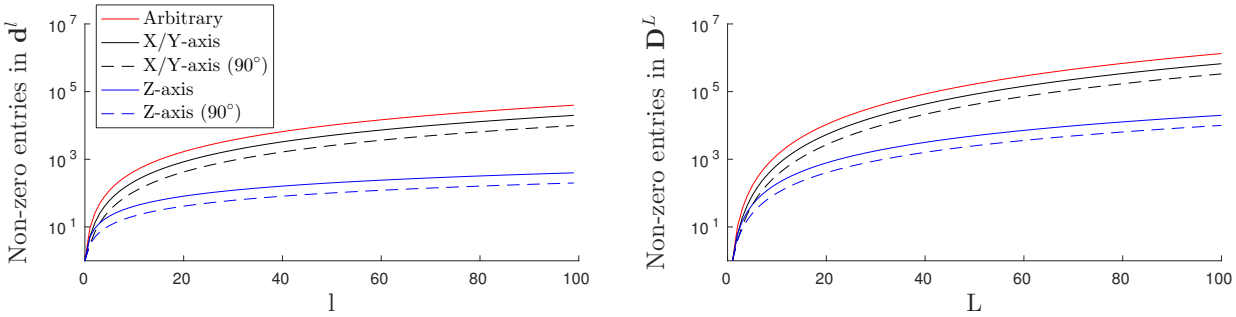


Figure 3.6: Logarithmic plot (base 10) of the maximal number of non-zero entries in the matrices $\mathbf{d}^l$ and $\mathbf{d}^L$ for different types of rotations. The number of non-zero entries can be calculated using corollary 3.27.

The results of this section are summarized in figure 3.6. The real Wigner-D matrices $\mathbf{d}^3$ for the three basic rotations around the Z, Y, and X-axis are presented in example 3.8. The relation between the transposed elements in the matrices can be used to halve the computation time needed to precalculate the real Wigner-D matrices. Furthermore, the sparsity of the real Wigner-D matrices can be used to reduce the computation time for rotations around the basis axes.

3.8 Translations

As depicted in example 3.4, we can describe a visual scene by a function f defined on the unit sphere S^2. In section 3.7 we derived formulas to rotate the function f. This can also be interpreted as a rotation (in opposite direction) of the agent and can be used to predict the visual scene assuming that the agent is rotated. In this section we derive additional formulas to simulate the effect of translations of the agent. Interpreting RSH as unit spheres S^2 on which the visual scene of a spherical camera is projected, we can define translations for RSH by simulating a shift of the cameras position inside the sphere. More specifically, we are interested in the behavior of RSH under a translation $\vec{t} \in \mathbb{R}^3$ of the 'center'

$$\vec{t} \circ y_m^l(\vec{p}) := y_m^l\left(\frac{\vec{p}+\vec{t}}{\|\vec{p}+\vec{t}\|}\right), \tag{3.86}$$

where $\vec{p} \in \mathbb{R}^3$ represents a spherical coordinate (ϑ, φ) on the unit sphere S^2. In this work we imagine a RSH function as a sphere with a given radius $0 < r \in \mathbb{R}$, i.e. an agent which is translated in a scene where all objects have the same distance r (**equal-distance assumption**). Explicit formulas to calculate $\vec{t} \circ y_m^l$ can be derived (van Gelderen, 1998, Danos and Maximon, 1965), however these formulas are limited to translations $\|\vec{t}\| < r$ inside the sphere. Therefore we define the translation operation slightly differently (figure 3.7).

Definition 3.28 (Translation of Spherical Harmonics)**.** *The translation of a RSH y_m^l by a vector $\vec{t} \in \mathbb{R}^3$ is defined as*

$$\vec{t} \circ y_m^l(\vec{p}) := \sum_{\gamma \in \Gamma} \lambda_\gamma y_m^l \left(\frac{\vec{t} + \gamma \vec{p}}{r} \right), \tag{3.87}$$

where $\Gamma = \left\{ \gamma \in \mathbb{R} \;\middle|\; \vec{t} + \gamma \vec{p} = r \right\}$ is the ascending ordered set of values γ such that $\vec{t} + \gamma \vec{p}$ intersects the sphere. The weighting term λ_γ can be chosen dependent on the use-case, e.g. to simulate the effect on the visual scene as perceived by a translated agent (section 3.8.3).

Figuratively speaking, $\vec{p}$ represents a viewing direction (or ray) from the translated position $\vec{t}$ and we are interested in the intersection of $\vec{t} + \gamma \vec{p}$ with the sphere. We address the problem of evaluating the set Γ in section 3.8.1 as well as the reformulation of translations as matrix-vector multiplications in the basis of RSH. In section 3.8.2 we show that translation along the Z-axis can be expressed using sparse matrices. Furthermore, we examine various interpretations of 'translations' by using different weighting terms λ_γ in section 3.8.3: First, we show how translations affect the visual scene as perceived by a translated agent. Second, we examine how point clouds could be translated in the basis of RSH. For this purpose, we show in section 3.8.4 how multiple spheres with differing radii r_i can be used to work with depth information if available. Note that the second interpretation of translations as well as using multiple spheres with differing radii are only suggestions to make RSH applicable for other tasks (e.g. alignment of point clouds), but are not tested or discussed in depth.

3.8.1 Approximation of Translation Matrices

To make use of translations as stated in definition 3.28, we need to evaluate the set Γ. For a unit sphere with radius r, Γ is the ascending ordered set of values γ such that $\vec{t} + \gamma \vec{p}$ with $\vec{t}, \vec{p} \in \mathbb{R}^3$ intersects the sphere. While translations can be calculated for arbitrary directions, we cover in the following only Z-axis translations, i.e. we have $\vec{t} = (0, 0, t)^T$. For arbitrary directions, we express translations as a concatenation of rotations and Z-axis translations (section 3.8.2). Now we need to solve $\|\vec{t} + \gamma \vec{p}\| = r$ for γ to find the solutions

$$\gamma_1 = -tz - \sqrt{t^2(z^2 - 1) + r^2} \quad \text{and} \quad \gamma_2 = -tz + \sqrt{t^2(z^2 - 1) + r^2}. \tag{3.88}$$

The set Γ can therefore contain up to 2 elements for each intersection as sketched in figure 3.7. This allows us to calculate the translation of a single point on the sphere such that we can approximate translations in the basis of RSH for L bands as follows: For a given Fourier coefficient vector $\vec{a}^L$ the translation

$$\vec{t} \circ \vec{a}^L := \sum_{l=0}^{L-1} \sum_{m=-l}^{l} a_m^l \left(\vec{t} \circ y_m^l \right) \tag{3.89}$$

is calculated as the sum of the translations applied to each RSH independently. We calculate for each RSH y_m^l the translated RSH $t \circ y_m^l$ as stated in definition 3.28: We have explicit formulas to calculate each value $y_m^l(\vec{p})$ (definition 3.15). For a set of sampling points on the sphere (section 3.10.1), we apply equation (3.88) to calculate the translation $t \circ y_m^l$ in the spatial domain. Afterwards, we Fourier transform $t \circ y_m^l$ from the spatial domain into frequency domain. The result is a Fourier coefficient vector $\vec{a}^L$ which fulfills

$$\vec{t} \circ y_m^l = \left[\vec{y}^L \right]^T \vec{a}^L. \tag{3.90}$$

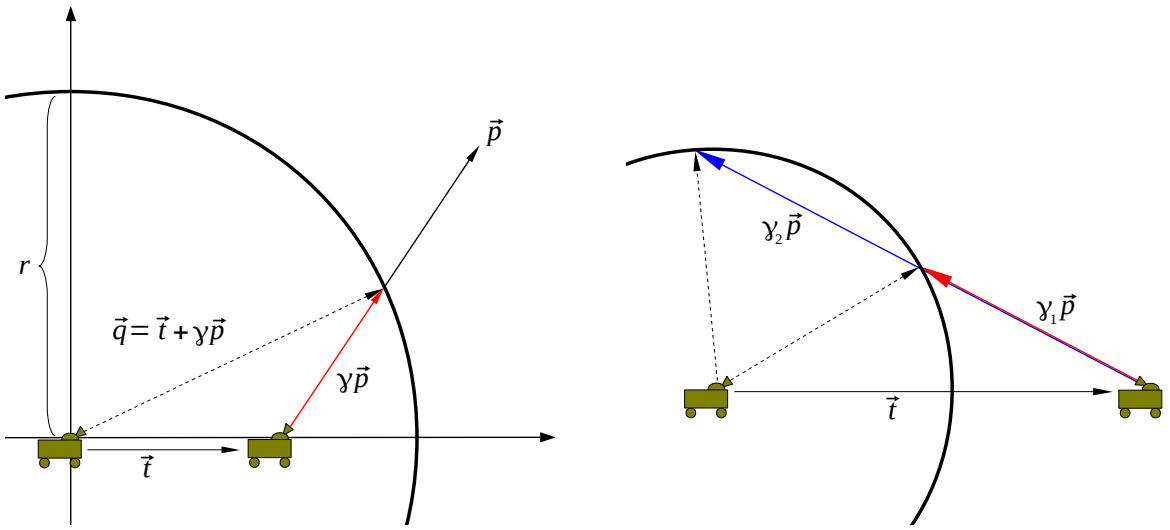


Figure 3.7: Left: Sketch of a translation of the 'center' by $\vec{t}$ with $\|\vec{t}\| < r$. As sketched, a use-case is to calculate the visual scene of a translated agent. For a given viewing direction $\vec{p}$ as seen from the translated position $\vec{t}$, we need the corresponding viewing direction $\vec{q}$ as seen from the original center. This can be achieved by calculating the distance γ at which $\vec{q} = \vec{t} + \gamma\vec{p}$ intersects the sphere. Right: If the translation $\|\vec{t}\| > r$ is outside the sphere the number of intersections can range between 0 and 2.

We calculate $\vec{t} \circ y_m^l$ for each basis element RSH y_m^l and stack the resulting Fourier coefficient vectors into a transformation matrix (i.e. the i-th column is the image of the i-th — here implicitly given by the indices l, m — basis element). We denote the resulting transformation matrix as **translation matrix** $\mathbf{T}_{\vec{t}}$. For the translation matrix $\mathbf{T}_{\vec{t}} = (T^{l_1,l_2}_{m_1,m_2})_{l_1,l_2,m_1,m_2}$ we use the following indexing scheme

$$\mathbf{T}_{\vec{t}} = \begin{pmatrix} T^{0,0}_{0,0} & T^{0,1}_{0,-1} & T^{0,1}_{0,0} & T^{0,1}_{0,1} & T^{0,2}_{0,-2} & T^{0,2}_{0,-1} & T^{0,2}_{0,0} & T^{0,2}_{0,1} & T^{0,2}_{0,2} & \dots \\ T^{1,0}_{-1,0} & T^{1,1}_{-1,-1} & T^{1,1}_{-1,0} & T^{1,1}_{-1,1} & T^{1,2}_{-1,-2} & T^{1,2}_{-1,-1} & T^{1,2}_{-1,0} & T^{1,2}_{-1,1} & T^{1,2}_{-1,2} & \\ T^{1,0}_{0,0} & T^{1,1}_{0,-1} & T^{1,1}_{0,0} & T^{1,1}_{0,1} & T^{1,2}_{0,-2} & T^{1,2}_{0,-1} & T^{1,2}_{0,0} & T^{1,2}_{0,1} & T^{1,2}_{0,2} & \\ T^{1,0}_{1,0} & T^{1,1}_{1,-1} & T^{1,1}_{1,0} & T^{1,1}_{1,1} & T^{1,2}_{1,-2} & T^{1,2}_{1,-1} & T^{1,2}_{1,0} & T^{1,2}_{1,1} & T^{1,2}_{1,2} & \\ T^{2,0}_{-2,0} & T^{2,1}_{-2,-1} & T^{2,1}_{-2,0} & T^{2,1}_{-2,1} & T^{2,2}_{-2,-2} & T^{2,2}_{-2,-1} & T^{2,2}_{-2,0} & T^{2,2}_{-2,1} & T^{2,2}_{-2,2} & \\ T^{2,0}_{-1,0} & T^{2,1}_{-1,-1} & T^{2,1}_{-1,0} & T^{2,1}_{-1,1} & T^{2,2}_{-1,-2} & T^{2,2}_{-1,-1} & T^{2,2}_{-1,0} & T^{2,2}_{-1,1} & T^{2,2}_{-1,2} & \\ T^{2,0}_{0,0} & T^{2,1}_{0,-1} & T^{2,1}_{0,0} & T^{2,1}_{0,1} & T^{2,2}_{0,-2} & T^{2,2}_{0,-1} & T^{2,2}_{0,0} & T^{2,2}_{0,1} & T^{2,2}_{0,2} & \\ T^{2,0}_{1,0} & T^{2,1}_{1,-1} & T^{2,1}_{1,0} & T^{2,1}_{1,1} & T^{2,2}_{1,-2} & T^{2,2}_{1,-1} & T^{2,2}_{1,0} & T^{2,2}_{1,1} & T^{2,2}_{1,2} & \\ T^{2,0}_{2,0} & T^{2,1}_{2,-1} & T^{2,1}_{2,0} & T^{2,1}_{2,1} & T^{2,2}_{2,-2} & T^{2,2}_{2,-1} & T^{2,2}_{2,0} & T^{2,2}_{2,1} & T^{2,2}_{2,2} & \\ \vdots & & & & & & & & & \ddots \end{pmatrix} \tag{3.91}$$

which is an enhancement of the indices used for the Wigner-D matrices $\mathbf{D}^l$ (red colored entries, compare definition 3.4). Due to its construction, the translation matrix fulfills

$$\vec{t} \circ \vec{a}^L = \mathbf{T}_{\vec{t}} \vec{a}^L. \tag{3.92}$$

and can be used to calculate translations directly in the basis of RSH.

3.8.2 Z-Axis Translation Matrices

Since the calculation of translation matrices $\mathbf{T}_{\vec{t}}$ is computationally expensive (depending on the number of bands and sample points), it is not convenient to calculate translations for arbitrary directions. As shown in this section, transformation matrices of Z-axis translations are sparse[6]

[6] We also investigated the sparseness of X/Y-axis translations, however their sparseness relations are complicated and their derivation is tedious. Since they are not necessary for our implementation of translations via equation

(theorem 3.29). Moreover, translations in the opposite direction $-\vec{t}$ (which are *not* the inverse matrices) can be calculated easily (theorem 3.30). Therefore we use Z-axis translations only and combine them with rotations to perform translations in arbitrary directions (**3D-warping**) as follows: First, an arbitrary translation $\vec{t} = (x, y, z)^T$ can be decomposed into a direction (ϑ, φ) (spherical coordinates) and a distance d. Second, we align the Z-axis with the translation axis given by the spherical coordinates (ϑ, φ). Third, we apply a translation in direction of the Z-axis (which is at this point aligned with the required translation direction). Finally, we undo the rotations. Taken together, we have

$$\mathbf{T}_{\vec{t}} = \mathbf{R}^T \mathbf{T}_{(0,0,d)^T} \mathbf{R}, \quad \text{with} \quad \mathbf{R} = \mathbf{R}_{Y,\vartheta} \mathbf{R}_{Z,\varphi}. \tag{3.93}$$

Note that this approach is not feasible for hemispherical continuation; in that case each translation matrix $\mathbf{T}_{\vec{t}}$ has to be calculated individually. The following theorem shows that for translations along the Z-axis the transformation matrix $\mathbf{T}_{(0,0,d)^T}$ is sparse.

Theorem 3.29. *Let $\vec{t} = (0, 0, z)^T$ be a translation along the Z-axis and $\mathbf{T}_{\vec{t}}$ be the corresponding translation matrix in the basis of RSH for L bands, then we have*

$$m_i \neq m_j \;\Rightarrow\; T^{l_1,l_2}_{m_1,m_2} = 0. \tag{3.94}$$

The corresponding translation matrix $\mathbf{T}_{\vec{t}}$ has $\frac{L(2L^2+1)}{3}$ non-zero entries.

Proof. See Wang et al. (2006). □

For a translation along the Z-axis, the following matrix shows the location of the non-zero entries (marked by 'X') as stated by theorem 3.29 for $L = 3$ bands:

$$\mathbf{T}_{(0,0,z)^T} = \left(\begin{array}{c|ccc|ccccc} X & & X & & & & X & & \\ \hline & X & & & & X & & & \\ X & & X & & & & X & & \\ & & & X & & & & X & \\ \hline & & & & X & & & & \\ & X & & & & X & & & \\ X & & X & & & & X & & \\ & & & X & & & & X & \\ & & & & & & & & X \end{array}\right) \tag{3.95}$$

Applying theorem 3.29, the calculation of translations in arbitrary directions using equation (3.93) consists of sparse matrices only. Besides translations as defined in equation (3.86), the proof of Wang et al. (2006) is also valid for translations using weightings (definition 3.28) as long as the weighting function is symmetric around the Z-axis[7]. This is true for the translation interpretations introduced in section 3.8.3 such that these can also be expressed using sparse matrices, reducing the computation times significantly.

If the translation matrix $\mathbf{T}_{(0,0,z)^T}$ is known, the following theorem allows us to calculate from it directly the transformation matrices $\mathbf{T}_{(0,0,-z)^T}$ in the opposite direction. Since the approximation of translation matrices can be time-consuming, this is especially useful to calculate the tangent distance from equation (3.136) efficiently.

Theorem 3.30. *Let $\vec{t} = (0, 0, z)^T$ be a translation along the Z-axis and $\mathbf{T} := \mathbf{T}_{\vec{t}}$ be the corresponding translation matrix in the basis of RSH for L bands. Define σ^{l_1,l_2}_m as given by the following table:*

(3.93), we decided to not invest more time examining the sparseness of X/Y-axis translations.

[7] In fact, they use a weighting function themselves: Their use-case is the calculation of environmental lighting and they use a weighting for energy preservation (Wang et al., 2006).

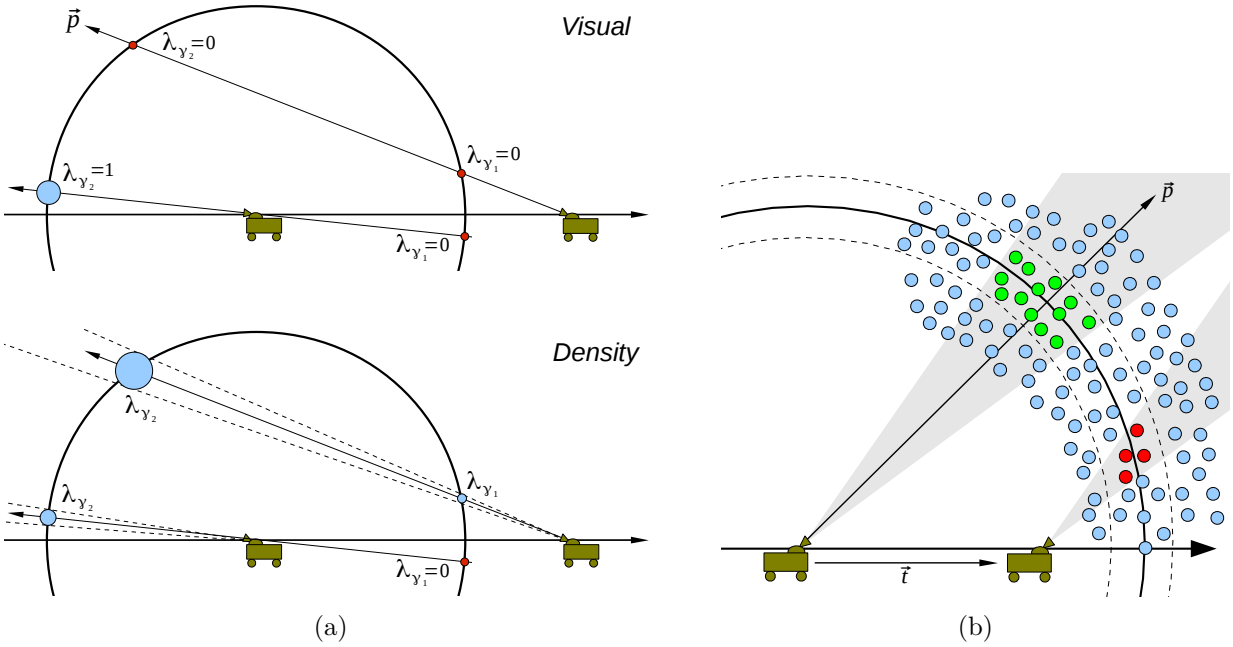


Figure 3.8: (a) Two different approaches for choosing the weights λ_γ (section 3.8.3). Blue circles show intersections which have a non-zero weight (the radius indicates the size of the weight) and red circles show intersections with a zero weight. The interpretation *visual* (top) assumes that the sphere represents the visual scene around an agent. It only takes the closest intersection in viewing direction ($\gamma > 0$) into account, weights for intersections 'behind' the viewing direction ($\gamma < 0$) are set to zero. Note that translations outside the sphere are undefined, in this case we set all weights to zero. The interpretation *density* (bottom) assumes that each sphere represents a point cloud. As depicted in (b), the translation of the agent affects the estimated number of points visible in viewing direction $\vec{p}$, we approximate this effect by choosing the weights as stated in equation (3.98). (b) Visualization of the interpretation *density*: Depending on the translation $\vec{t}$ and the viewing angle $\vec{p}$ the expected number of points observed by an agent changes.

$\sigma_m^{l_1,l_2}$	$l_1 + l_2$ *even*	$l_1 + l_2$ *odd*
$m \geq 0$	$+1$	-1
$m < 0$	-1	$+1$

Then the translation in the opposite direction $\mathbf{T}' := \mathbf{T}_{-\vec{t}}$ *can be obtained from the entries of* $\mathbf{T}$ *as*

$$\mathbf{T}'^{l_1,l_2}_{m,m} = \sigma_m^{l_1,l_2} \mathbf{T}^{l_1,l_2}_{m,m}. \tag{3.96}$$

For better readability we use that $m := m_1 = m_2$ *have to be equal, otherwise we have from theorem 3.29 that* $\mathbf{T}^{l_1,l_2}_{m_1,m_2} = \mathbf{T}'^{l_1,l_2}_{m_1,m_2} = 0$ *are zero.*

Proof. The proof can be found in appendix A.6. □

3.8.3 Interpreting Translations

In section 3.8.1 we described how to calculate the translation matrix $\mathbf{T}_{\vec{t}}$, however we did not discuss which values λ_γ should be used as weights in definition 3.28. Colloquially said, the weights change the resulting 'look' of a translation. Depending on the use-case, different interpretations of translations are reasonable. For example, a function f can be used to describe the visual scene as perceived by an agent. Assuming that all objects are equally far away from the agent (equal-distance assumption), a translation in the basis of RSH can be used to simulate the expected visual appearance at a new location. For this interpretation of translations, the object represented by each point $f(\vec{p})$ does not change its appearance. Therefore the weight is either $\lambda_\gamma = 1$ (visible) or $\lambda_\gamma = 0$ (not visible). However, for other interpretations — as for example the previously

mentioned energy preservation for environmental lighting (Wang et al., 2006) — the value of each point $f(\vec{p})$ is changed depending on the translation $\vec{t}$ and the viewing angle $\vec{p}$. The weight λ_γ can be chosen appropriately to achieve this effect. In the following we discuss two different weighting attempts for translations in detail; the interpretations *visual* and *density* (figure 3.8).

1. *Visual*: The function f describes the visual scene as perceived by an agent. Under the assumption that all objects are equally far away from the agent (equal-distance assumption), we can simulate the expected visual appearance at a new location by using the following weights (change in the pixel opening angle, figure 3.8, top left):

$$\lambda_{\gamma_1} = 0 \quad \text{and} \quad \lambda_{\gamma_2} = \begin{cases} 1 & \text{if } t \leq r \\ 0 & \text{if } t > r \end{cases} \tag{3.97}$$

   Note that for translations inside the sphere $\|\vec{t}\| < r$ the value γ_1 is always negative and corresponds to the intersection with the sphere opposite to the viewing direction. Contrary, the weight $\lambda_{\gamma_2} = 1$ corresponds to the intersection with the sphere in viewing direction and is set to a constant value.
2. *Density*: The function f is the projection of a point cloud on the unit sphere, i.e. each point $\vec{p} \in S^2$ equals the number of points of the point cloud in this viewing direction. As sketched in figure 3.8 (b), the expected number of points in a viewing direction changes depending on the translation $\vec{t}$ and the viewing angle $\vec{p}$: We assume that the points are approximately evenly distributed on the unit sphere. By approximating the viewing angle $\vec{p}$ as an infinitesimal small cone, the intersection of the cone with the sphere is roughly a disk. The area of the disk — which we choose as weight — only depends on the distance and can be used to calculate the weights

$$\lambda_{\gamma_1} = \frac{\gamma_1^2}{r^2} \quad \text{and} \quad \lambda_{\gamma_2} = \frac{\gamma_2^2}{r^2}. \tag{3.98}$$

   Note that the term r^2 is important if multiple unit spheres with differing radii $r_i \neq 1$ are used (slices, section 3.8.4).

An example for both interpretations *visual* and *density* can be found in example 3.9 and 3.10, respectively. In chapter 6, we use the interpretation *visual* to estimate home vectors (section 1.2.2) for visual robot navigation. We do not examine the interpretation *density* in detail, however — to the best knowledge of the author — the idea of estimating translations of point clouds in the basis of RSH is novel and could be used as an alternative technique for alignment of 3D laser scans, 3D models, or similar.

3.8.4 Slices

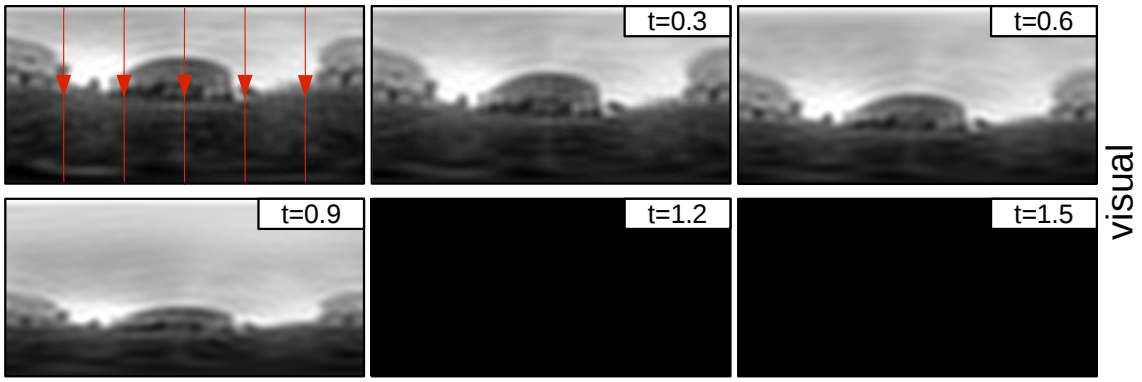
Up to this point, we discussed translations in a single unit sphere with radius r. However, using a single sphere we are not able to use (if available) depth information. For example, by projecting a point cloud onto a sphere we loose the distance information of each point which might add valuable information for the estimation of translations.

In order to make usage of distance information for translations, we suggest to use multiple spheres with varying radii, in the following called **slices**. Using the example of a point cloud, we can project each point in the most inner slice onto the most inner slice and each point outside the most outer slice onto the most outer slice. The remaining points are — if not exactly located on a single slice in which case the projection is trivial — located between two adjacent slices. These points can be projected, for example using a linear weight, on its adjacent slices. The latter two projections (outside and between two slices) are depicted in figure 3.9 (the same projection is later used to approximate the translation matrices).

Now we enhance translations to work with multiple slices with differing radii $r_i, i \in \{1, \ldots, n\}$ with $n \geq 2$ as follows: We have n source slices at the original center (**source slices**). To sustain

Example 3.9: Translation: Visual

By combining the results from the sections 3.8.1, 3.8.2, and 3.8.3 we are able to simulate translation of functions defined on the unit sphere directly in the basis of RSH. The images show a translation of $\vec{t} = (0, 0, t)^T$ along the Z-axis. The red arrows represent the flow field for the performed translation, indicating that pixel in the image move along a line extending from top to bottom.



Let a function f defined on the unit sphere S^2 describe the visual scene as seen by a camera. By assuming that the distance to each point in the visual scene is equal (equal-distance assumption), we can simulate the translation of the camera from the center of the sphere in direction $\vec{t}$ using the interpretation *visual*. For translations outside the sphere ($t \geq 1$), the visual appearance of the scene is undefined.

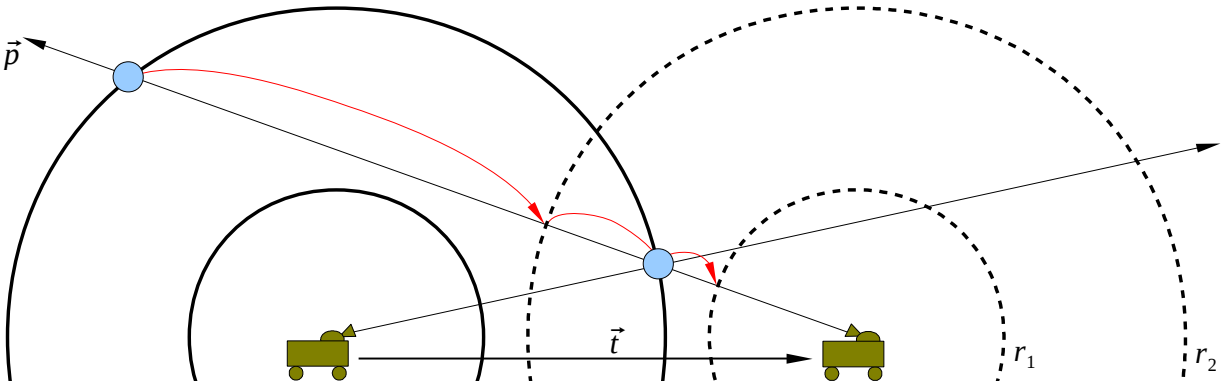
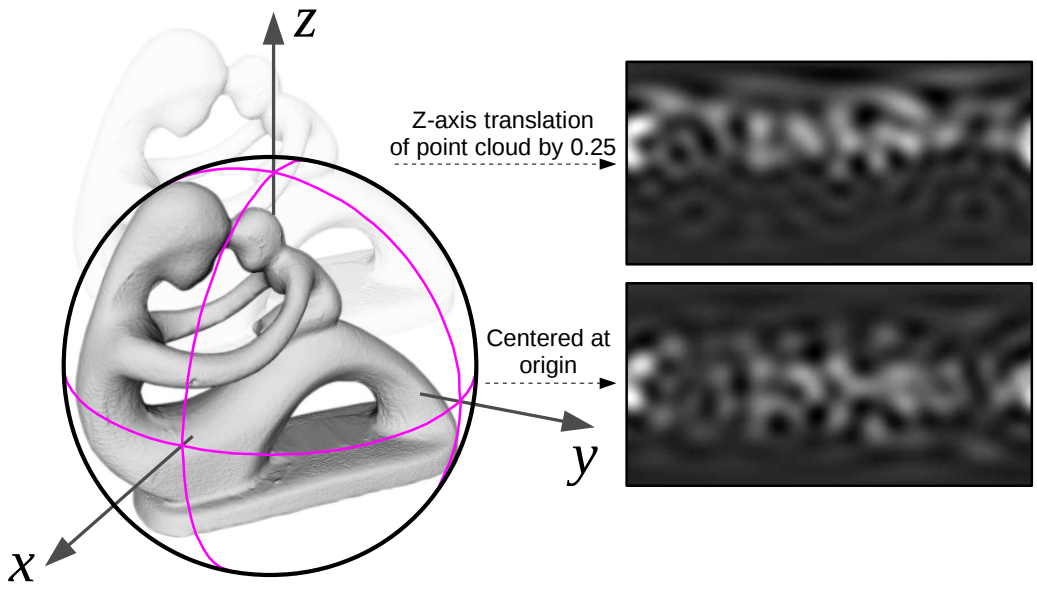


Figure 3.9: A translation with two source slices with radii r_1 and r_2. The intersection points are projected (linearly weighted, red arrows) on the closest target slices (dashed lines). The offset between the center of the source slices (solid lines) and target slices is given by the translation vector $\vec{t}$.

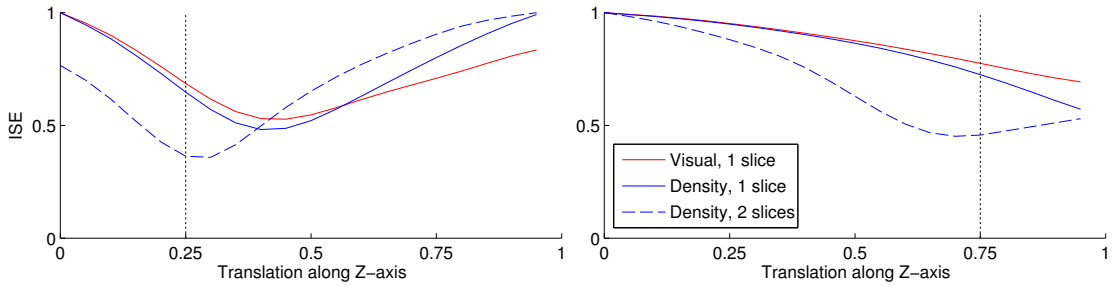
the distance information after translation, we also use n slices with the same radii r_i at the translated position (**target slices**). The basic calculations remain the same as stated in the sections 3.8.1, 3.8.2, and 3.8.3. The only difference to translations as described in the previous sections is that points are commonly located between adjacent target slices and therefore need to be mapped adequately. Therefore we need to calculate for each pair of source and target slices a transformation matrix. As a consequence, each translation is not represented by a single translation matrix as before but by a total of n^2 translation matrices.

As can be seen, the computational costs increase quadratically with the number of slices. If the conservation of distance information *after* the translation is not necessary, a single target slice could be used alternatively. This would reduce the number of required translation matrices to n.

Example 3.10: Translation: Density



The figure shows a 3D scan from the Aim@Shape repository (model 'Fertility', http://visionair.ge.imati.cnr.it/ontologies/shapes/viewgroup.jsp?id=670-fertility_-_watertight, Falcidieno (2004)). We took the low-polygon representation with 143 vertices (the coordinates of all vertices are used as a point cloud), centered it at the center of mass, and scaled it such that the most distant point had a distance of 1. Afterwards, we translated the point cloud by a distance of 0 (no translation), 0.25, and 0.75 along the Z-axis. For all three cases, we projected the points on the unit sphere S^2 and Fourier transformed it into the basis of RSH. The Fourier transformed point clouds without translation (bottom right) and with translation 0.25 (top right) are shown. To estimate the Z-axis translation between the different point clouds, we translate the projections directly in the basis of RSH using the formulas derived in section 3.8.



The plots show the normalized integral squared error (Y-axis, ISE, section 3.9.1) between the untranslated point cloud under varying translation estimations (X-axis) and the point clouds translated by 0.25 (left) and 0.75 (right). The estimated translation is then the translation which minimizes the ISE. Note that for two slices (with radii 0.5 and 1) we sum up the ISE for both pairs of slices. The results show that using the interpretation *density* with two slices noticeably improves the estimation quality compared to using either the interpretation *visual* or *density* with only one slice (with radius 1).

	Rotation-variant	Rotation-invariant
Phase-variant	ISE	BS
Phase-invariant		AS

Table 3.2: Overview of the different measures for functions $f, g : S^2 \to \mathbb{C}$ stated in this section: Integral squared error (ISE), amplitude spectrum (AS), and bispectrum (BS).

3.9 Distance Measures for Spherical Harmonics

For our localization algorithm (chapter 4), for the visual 3D compass (chapter 5), and for 3D-warping (chapter 6) we represent the visual scene at the location of an agent using a function f defined on the unit sphere. All of these algorithms have in common that the current view is compared with a set of snapshots, each again represented by a function g defined on the unit sphere. To decide if those functions are similar or different we need reliable similarity measures for functions on S^2. In this section three measures for SH and RSH are presented: The **integral squared error** (ISE), the **amplitude spectrum** (AS), and the **bispectrum** (BS); see table 3.2. The most basic measure is the ISE which integrates the squared difference between f and g over the whole sphere S^2. This measure is **rotation-variant**, i.e. it is sensitive to rotations. However, sometimes a measure which is not influenced by rotations — called **rotation-invariant** — might be advantageous: For example, our localization algorithm is designed to locate an agent on a previously driven training run independently of the orientation of the agent in the test run. This can be obtained using a rotation-invariant distance measure.

In the following we define the rotation-invariant amplitude spectrum and bispectrum (section 3.4) for functions defined on the unit sphere S^2. The amplitude spectrum *and* bispectrum have in common that they compare the amplitudes in the frequency domain, however *only* the bispectrum takes also phase information into account. Therefore we call the amplitude spectrum **phase-invariant**, while the bispectrum is **phase-variant**.

3.9.1 Integral Squared Error

Before we define the integral squared error (ISE), we examine the scalar product in the basis of SH first. Let $\vec{A}^L$ and $\vec{B}^L$ be the Fourier coefficient vectors of f and g for L bands, respectively, then we have

$$\langle f, g \rangle = \int_{S^2} f(s)\overline{g(s)}ds \tag{3.99}$$

$$= \int_{S^2} \left(\sum_{l=0}^{L-1} \sum_{m=-l}^{l} A_m^l Y_m^l(s) \right) \overline{\left(\sum_{l=0}^{L-1} \sum_{m=-l}^{l} B_m^l Y_m^l(s) \right)} ds \tag{3.100}$$

By reordering the terms

$$= \int_{S^2} \sum_{l=0}^{L-1} \sum_{m=-l}^{l} \sum_{l'=0}^{L-1} \sum_{m'=-l'}^{l'} A_m^l \overline{B_{m'}^{l'}} Y_m^l(s) \overline{Y_{m'}^{l'}(s)} ds \tag{3.101}$$

$$= \sum_{l=0}^{L-1} \sum_{m=-l}^{l} \sum_{l'=0}^{L-1} \sum_{m'=-l'}^{l'} A_m^l \overline{B_{m'}^{l'}} \int_{S^2} Y_m^l(s) \overline{Y_{m'}^{l'}(s)} ds \tag{3.102}$$

we can apply that the SH are orthonormal (equation (3.34)). Therefore most terms (all terms where $l \neq l'$ or $m \neq m'$) vanish and we obtain

$$\overset{(3.34)}{=} \sum_{l=0}^{L-1} \sum_{m=-l}^{l} A_m^l \overline{B_m^l} \underbrace{\int_{S^2} Y_m^l(s) \overline{Y_m^l(s)} ds}_{=1}. \tag{3.103}$$

$$= \left\langle \vec{A}^L, \vec{B}^L \right\rangle \tag{3.104}$$

As can be seen, the scalar product between two functions f, g can be calculated in the basis of SH as the scalar product of the Fourier coefficient vectors:

$$\langle f, g \rangle = \left\langle \vec{A}^L, \vec{B}^L \right\rangle \tag{3.105}$$

Using our previous results, we can now define the ISE between two functions $f, g : S^2 \to \mathbb{C}$ as follows (Friedrich et al., 2008):

Definition 3.31 (Integral Squared Error)**.** *Let $f, g : S^2 \to \mathbb{C}$ be functions on the sphere and denote by $\vec{A}^L$ and $\vec{B}^L$ the Fourier coefficients of the Fourier transforms of f and g for L bands. Then*

$$\mathrm{ISE}(f, g) := \|f - g\|^2 \overset{(3.105)}{=} \|\vec{A}^L - \vec{B}^L\|^2 \tag{3.106}$$

is called the integral squared error between f and g on S^2.

3.9.2 Amplitude Spectrum

The amplitude spectrum as discussed in section 3.4 can be generalized for functions $f : S^2 \to \mathbb{C}$ on the unit sphere represented by a Fourier coefficient vector $\vec{A}^L$. The derivation of the amplitude spectrum in the basis of SH can be found in Kazhdan et al. (2003) and is given band-wise for l by

$$\mathcal{AS}_f(l) = [\vec{A}^l]^\dagger \vec{A}^l = \|\vec{A}^l\|^2. \tag{3.107}$$

This allows us to define a rotation-invariant distance measure based on the amplitude spectrum as follows.

Definition 3.32 (Amplitude Spectrum)**.** *Let $f, g : S^2 \to \mathbb{C}$ be functions on the sphere. The amplitude spectrum for L bands (in vector form) is given by*

$$\vec{\mathcal{AS}}_f = (\mathcal{AS}_f(0), \ldots, \mathcal{AS}_f(L-1))^T$$

where each entry is the amplitude spectrum of f for band $l = 0, \ldots, L-1$ (analogously for g). Then we denote by

$$\mathrm{AS}(f, g) = \|\vec{\mathcal{AS}}_f - \vec{\mathcal{AS}}_g\| \tag{3.108}$$

the measure based on the amplitude spectrum.

3.9.3 Bispectrum

As for the amplitude spectrum, we have that the bispectrum as discussed in section 3.4 can be generalized for functions $f : S^2 \to \mathbb{C}$ on the unit sphere represented by a Fourier coefficient vector $\vec{A}^L$. As shown in Kakarala and Mao (2010), the bispectrum on the sphere S^2 can be calculated — using our notation of the point-wise product as defined in section 3.5.5 — as

$$\mathcal{BS}_f(l_1, l_2, i) = \underline{\vec{A}}_i^\dagger \left[\vec{A}^{l_1} \cdot \vec{A}^{l_2} \right] \in \mathbb{C} \tag{3.109}$$

with

$$\underline{\vec{A}}_i = (\underbrace{0, \ldots, 0}_{i^2-(l_2-l_1)^2}, \underbrace{A_{-i}^i, \ldots, A_i^i}_{2i+1}, \underbrace{0, \ldots, 0}_{(l_2+l_1+1)^2-(i+1)^2})^T. \tag{3.110}$$

Moreover, they show that $\mathcal{BS}_f(l_1, l_2, i) = \lambda \mathcal{BS}_f(l_2, l_1, i)$, where $\lambda \in \mathbb{C}$ is some constant independent of the function f. Therefore it is sufficient to calculate the entries $\mathcal{BS}_f(l_1, l_2, i)$ with $l_2 \geq l_1$. Finally, they show that the amplitude spectrum and the bispectrum are related:

$$\mathcal{BS}_f(l, 0, l) = A_0^0 \mathcal{AS}_f(l) \tag{3.111}$$

As can be seen, the amplitude spectrum is a special case of the bispectrum. Note that not all values $\mathcal{BS}_f(l_1, l_2, i)$ can be calculated since the sum index $i = l_2 + l_1$ may exceed the maximal band $L - 1$. Now we are able to define a rotation-invariant but phase-variant measure as follows:

Definition 3.33 (Bispectrum)**.** *Let $f, g : S^2 \to \mathbb{C}$ be functions on the sphere. Stacking all entries of the bispectrum of f (analogously for g)*

$$\vec{\mathcal{BS}}_f = (\mathcal{BS}_f(l_1, l_2, i))^T_{l_1, l_2, i},$$

we can express the bispectrum by a single vector $\vec{\mathcal{BS}}_f$. Then we denote by

$$\mathrm{BS}(f, g) = \|\vec{\mathcal{BS}}_f - \vec{\mathcal{BS}}_g\| \tag{3.112}$$

the measure based on the bispectrum.

3.9.4 Distance Measures for Real Spherical Harmonics

The measures derived in the sections 3.9.1, 3.9.2, and 3.9.3 are defined in the basis of SH. In this section we show how the measures can be adapted to work with real-valued functions $f : S^2 \to \mathbb{R}$ using RSH.

Fortunately, the basis transformation matrix $\mathbf{T}^l$ between SH and RSH is unitary, i.e. $\|\mathbf{T}^l \vec{A}^l\| = \|\vec{A}^l\|$ (definition 3.16). As a consequence, the ISE and the amplitude spectrum are the same in the basis of SH and RSH. For the bispectrum we show that it can be calculated directly in the basis of RSH using our definition of the real point-wise product (definition 3.19).

Theorem 3.34. *Let f be a real-valued function with Fourier coefficient vector $\vec{a}^L$. Then the bispectrum can be calculated as*

$$\mathcal{BS}_f(l_1, l_2, i) = \underline{\vec{a}}_i^\dagger \left[\vec{a}^{l_1} \cdot \vec{a}^{l_2}\right] \in \mathbb{R} \tag{3.113}$$

with

$$\underline{\vec{a}}_i = (\underbrace{0\ ,\ \ldots\ ,\ 0}_{i^2-(l_2-l_1)^2},\ \underbrace{a^i_{-i}\ ,\ \ldots\ ,\ a^i_i}_{2i+1},\ \underbrace{0\ ,\ \ldots\ ,\ 0}_{(l_2+l_1+1)^2-(i+1)^2})^T. \tag{3.114}$$

Proof. The proof can be found in appendix A.7. □

As can be seen, using real Clebsch-Gordan matrices the bispectrum can be calculated directly in the basis or RSH. The Fourier coefficient vectors $\vec{a}^L$ in the basis of RSH could also be transformed into the Fourier coefficient vector $\vec{A}^L$ in the basis of SH using appropriate transformation matrices (equation (3.58)), however this would have to be done for each Fourier coefficient vector $\vec{a}^L$. Using theorem 3.34, this transformation is implicitly performed using the (precalculated) real Clebsch-Gordan matrices.

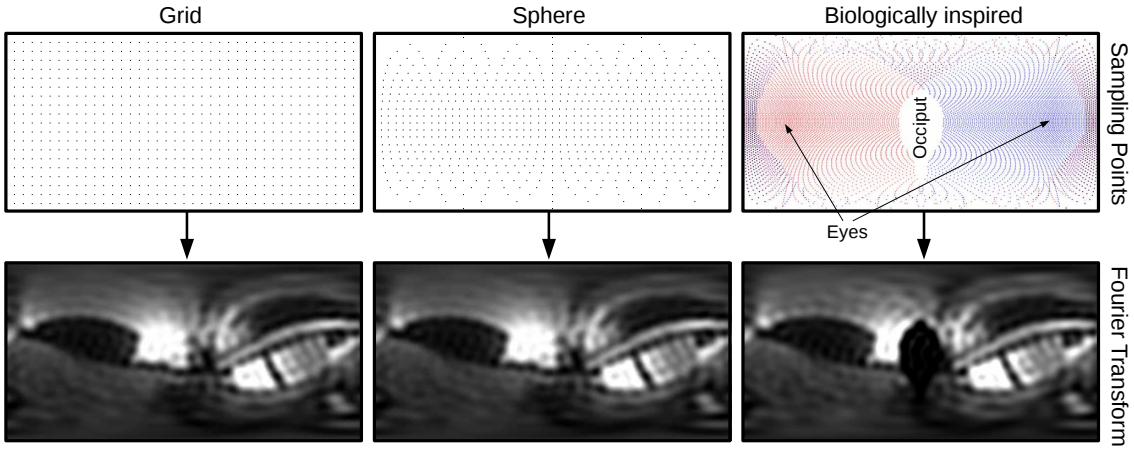
3.10 Implementation Details

In this section we present different strategies to implement the theory on SH derived before in a computationally efficient way. For high-precision calculations using SH, other libraries are available (e.g. Moore (2008) and Wieczorek (2015)). We focus on specialized functionality — especially for visual navigation tasks as performed by autonomous agents — optimized for speed instead of accuracy. We developed a C++ library called **libSHC**, which supports SIMD vector instructions (SSE, ARM NEON, etc.) via the linear algebra library 'Eigen' (Gaël et al., 2010) and FFT (Fast Fourier transform, section 3.10.2) for RSH via the library 'kissFFT' (Borgerding, 2006).

3.10.1 Sampling Points

Example 3.11: Sampling Points

The following panoramic images show exemplary distributions of sampling points on S^2 used to perform a Fourier transform. For the *biologically inspired* sampling point distribution, the color of the sampling points depicts to which eye they belong.



While the evenly distributed sampling points on the *sphere* cover the surface of S^2 well, the sampling points on a *grid* do not represent the surface of S^2 accurately (especially close to the poles). The *biologically inspired* sampling point distribution shows a high density of sampling points at the center of the eyes and none at the backside of the head (occiput). However, the visualization of the Fourier-transformed panoramic image is only slightly worse compared to those resulting from the sampling points evenly distributed on the *grid* and *sphere*. By filling the missing information with noise and/or applying a weighting function (section 3.10.3) it can be used for methods based on camera input as our localization algorithm (section 4), visual 3D compass (section 5), and drifting correction (section 6).

A function $f : S^2 \to \mathbb{R}$ can be Fourier-transformed into the basis of SH and RSH using the Peter-Weyl theorem (theorem 3.10). In a software implementation, a set of sampling points on the sphere S^2 has to be chosen to discretize the involved integrals. The **Nyquist–Shannon sampling theorem** states that for a band-limited function f with maximal band L (the Fourier coefficients a_m^l of f are zero for $l \geq L$) a total of $n = 2L$ sampling points on the sphere is sufficient to uniquely determine the Fourier coefficients of f via a discrete Fourier transform (Maslen, 1996). In practical applications, the band-limit L of a function is commonly unknown or has a high value. If a function with band-limit L is sampled with $n = 2L'$ sampling points and $L' < L$ it is called **undersampling** and artifacts might occur in the Fourier-transformed function. There are two approaches to deal with the problem of undersampling: First, the input signal f can be low-pass filtered in order to eliminate high frequencies. However, implementations of low-pass filters for S^2 are comparably slow since they have to cope with the topology of S^2 (Bülow, 2001). Second, the function can be **oversampled**, i.e. the number of sampling points $n \gg 2L$ is chosen larger than the number of sampling points stated by the Nyquist–Shannon sampling theorem. The second approach — which is used in our implementation — is especially useful if camera images are used as input due to the large number of pixels available.

The discrete Fourier transform (DFT) for RSH on S^2 can be calculated by approximating the integral in the Peter-Weyl theorem (theorem 3.10) with a finite sum. Using a total of n sampling

points $(\vartheta_i, \varphi_i) \in S^2$ with $i = 1, \ldots, n$, we can calculate each Fourier coefficient as

$$\sum_{i=1,\ldots,n} w_{\vartheta_i,\varphi_i} f(\vartheta_i, \varphi_i) y_m^l(\vartheta_i, \varphi_i). \tag{3.115}$$

Here, $w_{\vartheta_i,\varphi_i}$ is a weighting term which corresponds to the number of points per area, i.e. the sum of all weighting terms should equal the surface area of the unit sphere $\sum_{i=1}^{m} w_{\vartheta_i,\varphi_i} = 4\pi$. Depending on the task, different sampling point distributions can be used; example 3.11 shows three different approaches.

If we integrate over a $m \times n$ **grid** of spherical coordinates $(\vartheta_i, \varphi_j) = (\frac{\pi i}{m}, \frac{2\pi j}{n})$, the weight is given as stated in equation (3.36) by $w_{\vartheta_i,\varphi_j} = \sin(\vartheta_i)$. Note that the weighting terms can be normalized such that the sum equals 4π by simply multiplying each weight with an appropriate scalar. The key point is that the weights represent the distribution of the sampling points correctly, i.e. smaller weights for points close to the poles (many points for a small area) and bigger weights for points close to the equator (few points for a large area). Alternatively, the points itself can be evenly distributed over the **sphere** such that each points covers the same area, i.e. $w_{\vartheta_i,\varphi_i}$ is constant for all $i = 1, \ldots, n$. An approximately equal distribution (regarding the number of points per area) can be created as shown by Deserno (2004). Both approaches have in common that the sampling points form 'rings' parallel to the equator (sampling points with the same elevation value ϑ) as necessary for the fast Fourier transform (section 3.10.2). To examine if insects could be able to perform the Fourier transform, we furthermore added a **biologically inspired** distribution of sampling points based on the distribution of ommatidia (Seidl, 1982) in the eye of the worker honey bee[8] (example 3.11).

3.10.2 Fast Fourier Transform

We use the **separation of variables** approach from Kostelec and Rockmore (2008) to implement the **fast Fourier transform** (FFT) on S^2. Their approach states a FFT on the rotation group $SO(3)$ and uses the Nyquist–Shannon sampling theorem (section 3.10.1). Since we apply the FFT on the unit sphere S^2 and want to use oversampling to avoid artifacts, we do not use their approach directly but simplify it for our use-case: The basic idea is to separate the calculations for the variables ϑ and φ and perform a standard 1D Fourier transform (definition 3.1) for each 'ring' of sampling points parallel to the equator (sampling points with the same elevation value ϑ). From the Peter-Weyl theorem (theorem 3.10) we have that the Fourier coefficients of a function $f : S^2 \to \mathbb{C}$ in the basis of SH are given by

$$A_m^l = \int_{\varphi=0}^{2\pi} \int_{\vartheta=0}^{\pi} f(\vartheta, \varphi) \bar{Y}_m^l(\vartheta, \varphi) \sin\vartheta \, d\vartheta \, d\varphi. \tag{3.116}$$

Note that the sine term in equation (3.116) is a consequence of the coordinate transform (integrating in spherical coordinates). Now we assume that $m > 0$ — the other cases are shown equivalently — then we can write the SH as $Y_m^l = K_m^l e^{im\varphi} P_m^l(\cos\vartheta)$ using the alternative formulation of SH from section 3.5.4. Substituting this into the Fourier transform from equation (3.116), we obtain

[8] The distribution of ommatidia as stated by Seidl (1982) can be found on `http://www.insectvision.org/flying-insects/bee-model`. In a personal correspondence with the authors, we obtained the distribution data of ommatidia to implement the biologically inspired sampling point distribution.

by reordering the integrals

$$A_m^l = \int_{\varphi=0}^{2\pi} \int_{\vartheta=0}^{\pi} f(\vartheta,\varphi)\bar{Y}_m^l(\vartheta,\varphi)\sin\vartheta d\vartheta d\varphi \tag{3.117}$$

$$\overset{(3.47)}{=} \int_{\varphi=0}^{2\pi} \int_{\vartheta=0}^{\pi} f(\vartheta,\varphi)K_m^l P_m^l(\cos\vartheta)e^{-im\varphi}\sin\vartheta d\vartheta d\varphi \tag{3.118}$$

$$= \int_{\vartheta=0}^{\pi} K_m^l P_m^l(\cos\vartheta)\sin\vartheta \underbrace{\int_{\varphi=0}^{2\pi} f(\vartheta,\varphi)e^{-im\varphi}d\varphi}_{\mathcal{F}_{f(\cdot,\varphi)}(m)} d\vartheta, \tag{3.119}$$

where $\mathcal{F}_{f(\cdot,\varphi)}(m)$ is the standard 1D Fourier transform (definition 3.1). Each set of sampling points with the same elevation value ϑ can therefore be Fourier-transformed individually using existing FFT techniques (Das (2012), chapter 8). Due to the arrangement of the sampling points, this can easily be implemented in our C++ library for the sampling points on a *grid* and *sphere*. Moreover, this approach works with oversampling (section 3.10.1) and can simply be adapted to halve the computation time for hemispherical continuation (section 3.6.2) by using even or odd bands l only. The FFT for RSH is derived analogously or, as an alternative, the transformation matrix from section 3.6 can be used to obtain the Fourier coefficients in the basis of RSH. For arbitrary sampling point distributions — for example the *biologically inspired* sampling point distribution — the FFT as stated in this section cannot be applied and is replaced with the slower DFT.

Analogously to the fast Fourier transform, we can define an **inverse fast Fourier transform** (IFFT). From equation (3.35) we have that the inverse Fourier transform is given by:

$$f(\vartheta,\varphi) \overset{(3.35)}{=} \sum_{l=0}^{L-1}\sum_{m=-l}^{l} A_m^l Y_m^l(\vartheta,\varphi) \tag{3.120}$$

Again, we assume $m \geq 0$ and use the alternative formulation of SH from section 3.5.4 to obtain

$$\overset{(3.47)}{=} \sum_{l=0}^{L-1}\sum_{m=-l}^{l} \underbrace{A_m^l K_m^l P_m^l(\cos\vartheta)}_{B_m^l :=} e^{im\varphi} = \sum_{l=0}^{L-1}\sum_{m=-l}^{l} \underbrace{B_m^l e^{im\varphi}}_{\mathcal{F}_{B_m^\cdot}^{-1}(\varphi)}. \tag{3.121}$$

Therefore the inverse Fourier transform can also be calculated for each set of sampling points with a fixed value ϑ individually using a standard 1D inverse Fourier transform (definition 3.2).

Using the FFT addresses two problems: First, the computation time of the (inverse) FFT is significantly reduced in comparison to the (inverse) DFT. Second, we do not have to precalculate the spherical coordinates for each sampling point as necessary by the DFT. As can be seen in the equations (3.119) and (3.121), for the (inverse) FFT the variables ϑ and φ have been separated. The computation of the (inverse) FFT therefore simplifies to a sum of standard of 1D standard FFTs. Each element of the sum is weighted by a term depending only on l, m, and ϑ; these terms can furthermore be precalculated. In more detail, this is the term $K_m^l P_m^l(\cos\vartheta)\sin\vartheta$ from equation (3.119) for the FFT and the term $K_m^l P_m^l(\cos\vartheta)$ from equation (3.121) for the IFFT.

3.10.3 Non-Spherical Input

Fish-eye lenses are low-cost imaging devices which can be used to capture panoramic images. However, current fish-eye lenses[9] are limited to a maximal opening angle of around 220°. As soon

[9] To the best of our knowledge (July 2016) there are no low-cost fish-eye lenses with an opening angle greater than 220° available at the consumer market. More sophisticated imaging devices (e.g. multi camera systems) are available, but comparably expensive.

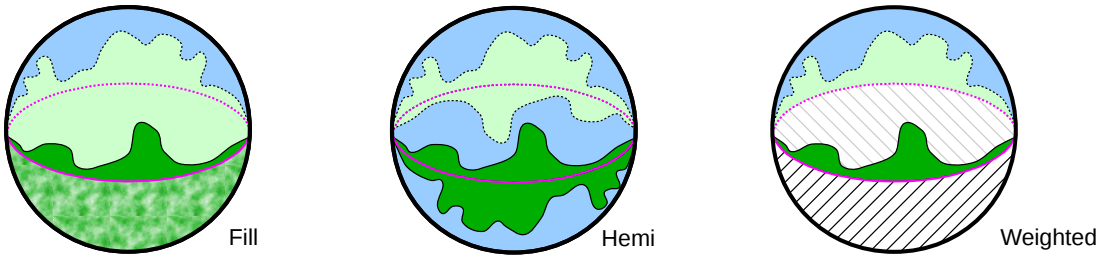


Figure 3.10: Sketch of our three approaches used to work with hemispherical input. Fill: The lower hemisphere is filled with noise (example 3.13). Hemi: Using hemispherical continuation we can mirror the upper hemisphere to the lower hemisphere (example 3.6). Weighted: The lower hemisphere is 'removed' by applying an appropriate weighting function.

as we try to compare two panoramic images with each other, undefined areas (e.g. the camera housing, parts of the robot, blind spots) may influence the result. Filling undefined areas with a constant value is no solution to the problem: Applications like the visual 3D compass (chapter 5), which try to find a transformation that reduces the difference between two panoramic images, tend for constant values to match the undefined areas to each other. Therefore a full-spherical approach should be used, where undefined areas do influence the comparison between two panoramic images as little as possible. Throughout this work we use a fish-eye lens with 180° or 220° opening angle as shown in example 3.4 such that the upper hemisphere of S^2 is always well defined. Figure 3.10 gives an overview of the three different approaches used in this work to deal with hemispherical input.

First, a naive approach is to fill in missing information (method **fill**); however the added information should affect the comparison between two functions f and g defined on S^2 as little as possible. Instead of filling missing information with constant data, we fill in different types of noise. A description of the used types of noise can be found in section 3.11.1.

Second, we can use hemispherical continuation (method **hemi**) from section 3.6.2 to mirror the image information from the upper hemisphere to the lower hemisphere. Using the invariant subsets I_{RM} or I_{RMN}, this approach halves the computation times for all calculations since only odd or even bands are used. The function f is Fourier-transformed as usual (i.e. the FFT from section 3.10.2), however for the input of a 180° fish-eye lens it is sufficient to Fourier transform the upper hemisphere for odd (or even) bands only. Using hemispherical continuation, the additional information of panoramic images with an opening angle of more than 180° viewing angle (pixel of the lower hemisphere) cannot be used.

Third, we suggest to use weighting functions (method **weighted**) allowing us to use arbitrary panoramic imaging devices. As for the approach *fill*, we fill in the missing image information of a panoramic image with noise. However, we additionally use weighting functions to reduce the influence of missing (invalid) information. Let f, g be panoramic images and $w_f, w_g : S^2 \to [0,1]$ their weighting functions, then we can adapt the integral squared error (ISE, section 3.9.1) as follows: In image space we define the ISE for weighted functions $f, g : S^2 \to \mathbb{R}$ (**WISE**) as

$$WISE(f,g,w_f,w_g) = \frac{\int_{S^2}[f(s)-g(s)]^2 w_f(s) w_g(s) ds}{\int_{S^2} w_f(s) w_g(s) ds} = \frac{\langle f-g, (f-g) w_f w_g \rangle}{\langle w_f, w_g \rangle}, \tag{3.122}$$

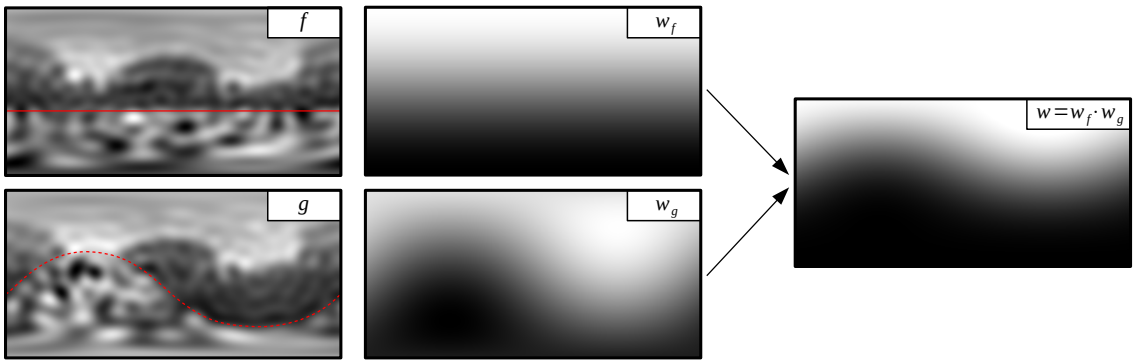
where $w_f, w_g : S^2 \to \mathbb{R}$ are the weighting functions for f, g. Now let $\vec{f}, \vec{g}, \vec{w}_f, \vec{w}_g$ be the Fourier coefficients of f, g, w_f, w_g, then we already know that the scalar products (equation (3.105)) and point-wise products (section 3.5.5) between two functions can be calculated directly in the basis of RSH as

$$WISE(f,g,w_f,w_g) = \frac{\left\langle \vec{f}-\vec{g}, (\vec{f}-\vec{g}) \cdot \vec{w}_f \cdot \vec{w}_g \right\rangle}{\langle \vec{w}_f, \vec{w}_g \rangle} \tag{3.123}$$

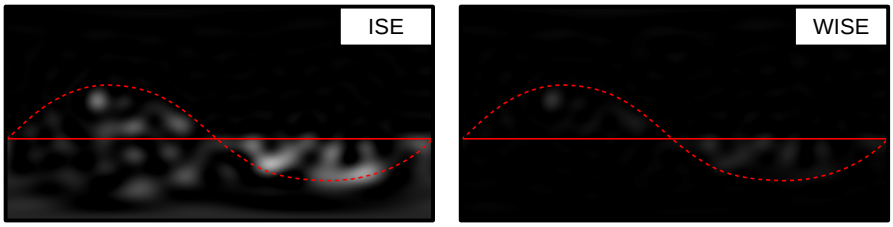
where '$\cdot$' is the point-wise product for RSH (theorem 3.19). It is straight-forward to modify the

visual 3D compass used in chapter 5 to use WISE instead of the ISE. The computation time necessary to calculate the point-wise products depends on the maximal number of bands (section 3.5.5). Note that arbitrary weighting functions can be used, e.g. to adapt for blind spots of the camera, however, with an increasing complexity of the weighting functions shape, the number of bands L necessary to represent it increases.

Example 3.12: ISE versus WISE



Let f, g describe the current view and snapshot with a viewing angle of 220° collected at the same location but with a rotational offset. Afterwards, we manually aligned both images and as a consequence — due to their different viewing angles — different parts of the images are filled with noise (red lines). The weighting functions w_f, w_g are as defined in equation (3.124), however the weighting function w_g was additionally rotated by the same amount as g (both are rotated by $-40°$ around the X-axis). The weighting function $w = w_f \cdot w_g$ is used to calculate the WISE.



The squared pixel-wise differences $[f(s) - g(s)]^2$ (i.e. ISE) and the weighted squared pixel-wise differences $[f(s) - g(s)]^2 w_f(s) w_g(s)$ (i.e. WISE) are shown. The differences are for both formulas between zero (black) and one (white). As can be seen, for the ISE errors are encountered for areas filled with noise. In contrast, these errors are strongly reduced by the weighting function $w = w_f w_g$ using WISE.

For time-critical applications the most important objective is to reduce the number of maximal bands used for the weighting functions. Here we suggest such a weighting function w_f which reduces the influence of the noise in the lower hemisphere, while still providing additional information from the wide-angle lens. Our weighting function of choice for the visual 3D compass (assuming that we use fish-eye lenses) is:

$$w : S^2 \to [0,1], \quad (\vartheta, \varphi) \mapsto \frac{\cos\vartheta + 1}{2} \tag{3.124}$$

This function can be expressed using RSH as $w(\vartheta,\varphi) = \sqrt{\pi}\ y_0^0(\vartheta,\varphi) + \sqrt{\pi/3}\ y_0^1(\vartheta,\varphi)$, i.e. its Fourier coefficient vector is $\vec{w} = (\sqrt{\pi}, 0, \sqrt{\pi/3}, 0)^T$. A visualization of w can be found in example 3.12. The weighting function w has two important properties: First, the transition from $\vartheta = 0$ to $\vartheta = \pi$ is smooth as only low-frequency RSH are used. This allows us to use arbitrary wide-angle

fish-eye lenses ($\vartheta > \frac{\pi}{2}$). Second, the Fourier coefficient vector $\vec{w}$ has only a maximum of $L = 2$ bands such that rotations of $\vec{w}$ as well as point-wise products can be computed efficiently.

3.11 Further Improvements

In this section we describe different techniques which can be used to tweak algorithms based on RSH, especially the visual 3D compass (section 5.2). These techniques can be used to increase the overall performance, e.g. by choosing an appropriate type of noise to fill in missing information on panoramic images (section 3.11.1), or to increase the invariance against illumination changes by using preprocessed panoramic images (section 3.11.2). Furthermore, arbitrary transformations can be approximated by linearizations and used to obtain invariances, e.g. for small translations of the camera in the visual 3D compass (section 3.11.3).

3.11.1 Noise

In section 3.10.3 we showed three different methods to deal with non-spherical input as obtained by a fish-eye camera. Except for the method *hemi* applied to hemispherical images, these methods use noise to fill in missing information. The type of noise should influence the comparison between two function f and g defined on S^2 as little as possible. In this work we compare two different types of noise (example 3.13):

- **White Noise**: Missing information is filled with noise whose amplitude spectrum (in the basis of RSH) is constant.
- **Natural Noise**: Missing information is filled with noise whose amplitude spectrum (in the basis of RSH) is equal to the average amplitude spectrum of the images collected in the *mixed* databases (*indoor*, *winter*, and *summer*, appendix D).

We compute the noise in the frequency domain (for L bands) as follows: First, an amplitude spectrum $\mathrm{AS}(l) : [0, L-1] \to \mathbb{R}_{\geq 0}$ (based on the desired type of noise) is defined. Then we draw random values from an uniform distribution on the interval $[-1, 1]$ to fill a Fourier coefficient vector $\vec{a}^L$. Afterwards, the amplitude spectrum $\|\vec{a}^l\|^2$ from equation (3.107) is calculated for each band l and scaled (assuming that $\|\vec{a}^l\| \neq 0$) by multiplying all Fourier coefficients by a factor such that it is equal to $\mathrm{AS}(l)$.

In this way, we can calculate a random Fourier coefficient vector $\vec{a}^L$ with the specified amplitude spectrum. By applying an inverse Fourier transform, we can also obtain the noise in the spatial domain to fill in missing information, e.g. in panoramic images (section 3.10.3). We precalculate several panoramic images containing noise as described above. This allows us to quickly fill in missing areas of panoramic images as for example captured using fish-eye cameras. The noise inserted into a panoramic image is always normalized such that the mean and standard derivation of the noise and the available information of the panoramic image are equal. Otherwise, the noise would not represent missing image information adequately, e.g. a bright panoramic image should not be filled with comparably dark noise.

To obtain the natural amplitude spectrum used to create natural noise (example 3.13) we calculated the mean amplitude spectrum (for $L = 100$ bands) of all panoramic images in the *mixed* databases (*indoor*, *winter* and *summer*). Afterwards, the amplitude spectrum was fitted using *lsqcurvefit* from MATLAB (2012) as

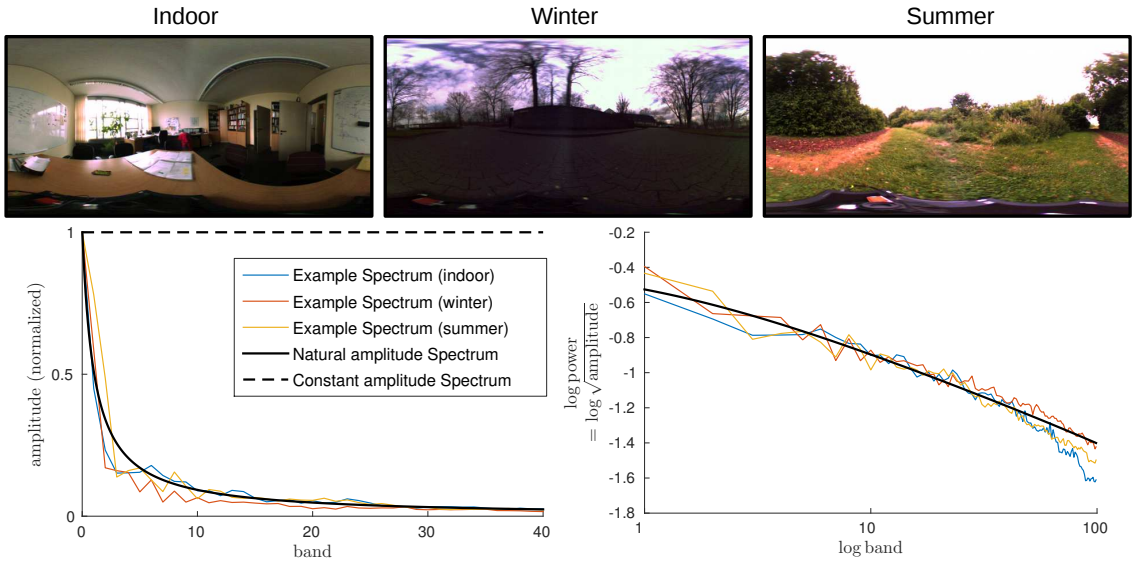
$$\mathrm{AS}_{\mathrm{Natural}}(l) = (0.0013l^2 + 0.9673l + 0.9983)^{-1}. \tag{3.125}$$

The result is shown in example 3.13.

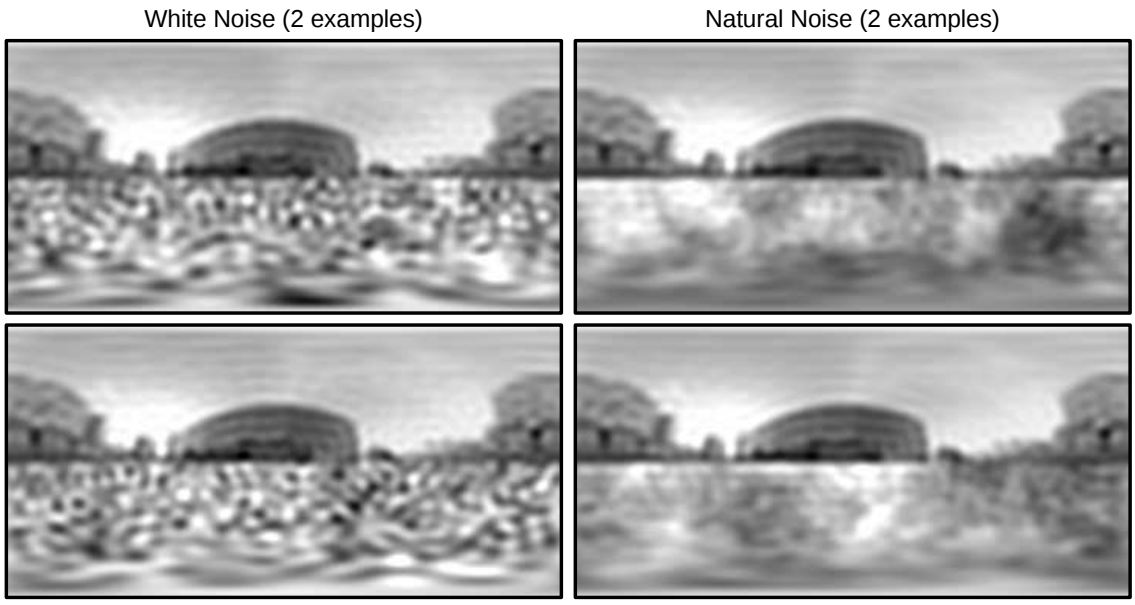
3.11.2 Image Preprocessing

Illumination changes strongly influence the visual appearance of a scene. For example, an indoor cleaning robot might visit the same place multiple times over a day, however the visual appearance will be altered by the lighting conditions: Sunlight shining through the window in the morning

Example 3.13: Noise



The left plot shows the amplitude spectra of three different examples taken from our spherical databases (appendix D) for the first 40 bands. The right plot shows the logarithmized power spectra — power is the square root of the amplitude — of the same data (for 100 bands) for comparison with Ruderman and Bialek (1994) and van der Schaaf and van Hateren (1996) who studied the power spectra of natural scenes in perspective images. Their results indicate that the logarithmized powerspectra as a function of the logarithmized band is nearly linear. As can be seen, the logarithmized power spectra in the basis of RSH is also nearly linear which copes well with the power spectra collected in those studies. Furthermore, we fitted a function to the average amplitude spectrum of all images in the *mixed_** databases (*natural amplitude spectrum*).



We use the natural amplitude spectrum to create **natural noise** as described in section 3.11.1. For comparison, we use a *constant amplitude spectrum* which produces **white noise**.

has disappeared at late hours, lights can be turned on or off, and shadows wander through a room. Comparable situation are encountered outdoors, e.g. a cloud passing in front of the blue sky can influence the visual appearance of the scene significantly. As discussed in Möller et al. (2014) and Dederscheck et al. (2010a), illumination changes which influence the complete visual scene can be split into two groups: First, absolute changes or **shifts**, e.g. if the overall illumination is increased or decreased. Second, multiplicative changes or **scaling**, e.g. the appearance of highly reflective surfaces changes stronger on illumination changes in comparison to poorly reflective surfaces. Formally, for global illumination changes which affect the complete visual scene, the new brightness value of each pixel p can be approximated by

$$I(p) = \alpha p + \beta \quad \text{with} \quad \alpha, \beta \in \mathbb{R}. \tag{3.126}$$

Note that this simple approximation does not take into account the local information of each pixel, as for example illumination changes caused by shadows or reflective properties. Moreover, this approximation does not consider over- or underexposed pixels.

We already discussed the use of the skyline as landmark for visual navigation which is invariant to illumination changes (chapter 2), however the use is limited to outdoor environments. In this section we discuss more general ideas which can be used for both indoor and outdoor environments to approach problems induced by illumination changes. Commonnly used methods to achieve illumination invariance on images are **edge-filtering** (Stürzl and Zeil, 2007, Möller et al., 2014), **histogram equalization** (Milford and Wyeth, 2012), and shadow removal (Corke et al., 2013). In this work we use edge-filtering and histogram equalization to increase illumination invariance (example 3.14).

A pixel in an image is called an edge if the brightness values of pixels in its vicinity change abruptly. If illumination changes appear, the change of the gradient is commonly retained and therefore the edge still visible. The most simple method to realize an edge-filter is to apply a high-pass filter in the frequency domain. Applied to RSH, this can be done by setting the Fourier coefficient vectors $\vec{a}^l = 0$ for all bands $l < L_{\text{high-pass}}$. While this method is computationally cheap, the visual 3D compass used with the resulting panoramic images showed poor performance in all tests and is only mentioned for the sake of completeness. Commonly, edge-filtering is achieved by convolving the image with appropriate filter-kernels in the spatial domain. For an overview of edge-filtering methods, we refer to Parker (1997), chapter 2. Here we use the *Sobel* operator (Gonzalez and Woods (1992), section 7.1) which computes absolute edge values, making it robust to illumination changes with $\alpha < 0$.

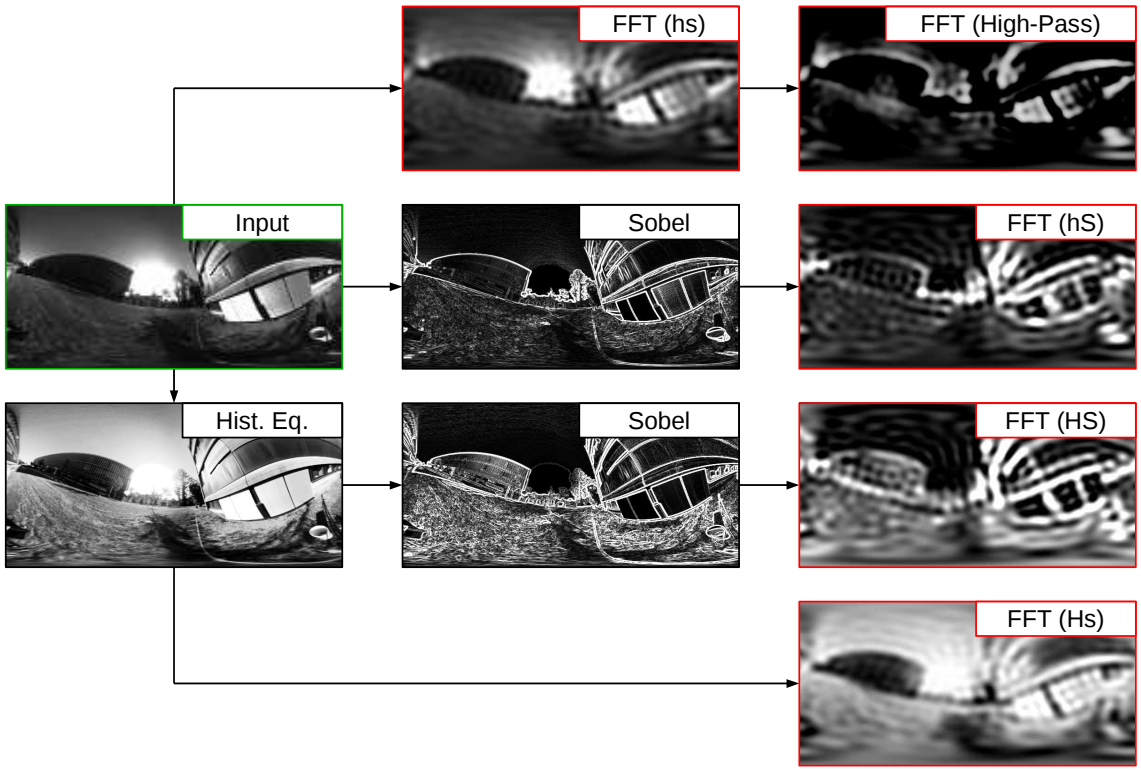
The aim of histogram equalization is to transform the image such that the histogram of the brightness values is uniform. As a result, the contrast in the image is increased, reducing the impact of illumination changes. A histogram equalization is performed by applying a pixel-wise transform for brightness values as described in Gonzalez and Woods (1992), section 4.2. Histogram equalization can be applied globally on the complete image, on local regions, or a mixture of both (Zhu et al., 1999). In our implementation we use both global and local histogram equalization.

For both global and local histogram equalization we obtain an edge-filtered panoramic image in frequency space by applying the Fourier transform afterwards. The suggested methods are computationally cheap and are therefore suitable for time critical applications as the visual 3D compass (chapter 5).

3.11.3 Tangent Distance

The **tangent distance** is a tool originally used for classification problems, especially pattern recognition, by Simard et al. (1998). The idea is that in a feature space the (Euclidean) distance between two features does not take a priori knowledge about the classification problem into account. However, for pattern recognition we know that a handwritten digit from different persons might appear rotated, translated, shifted, or scaled compared to each other. These differences can be observed and formulated as operations on the feature space itself. The goal is to find a

Example 3.14: Preprocessing



Preprocessing applied to a panoramic image (green frame), each resulting in a different output (Fourier-transformed with $L = 30$ bands, red frames). (hs): No preprocessing applied. (High-Pass): Applying a high-pass filter on the Fourier coefficient vector of a panoramic image by only keeping high frequencies (here: $7 < l < 30$). (hS): The Sobel operator is applied on a panoramic image to extract edges. (HS): Global histogram equalization and the Sobel operator, in this order, are applied to the image. (Hs): Global histogram equalization is applied to the input image to increase the contrast.

distance measure which is invariant to these operations. There are mainly two problems which need to be dealt with: First, the feature space is high-dimensional, e.g. for pattern recognition on 20×20 images it is already 400-dimensional. Second, the operations are commonly non-linear. The tangent distance addresses both problems by fitting a tangent plane to each feature which is tangential to the applied operation. As a consequence, the problem reduces to the computation of the shortest distance between those tangent planes.

Let us denote by $\vec{x} \in \mathbb{R}^n$ a feature in a n-dimensional feature space and by $f(\vec{x}, \vec{p}) : \mathbb{R}^n \to \mathbb{R}^n$ an operation on the feature space with parameters $\vec{p} = (p_1, \ldots, p_m)^T$. For example, this could be an angle for a rotation or a tuple of coordinates for translations. Furthermore, we require that the operation $f(\vec{x}, \vec{p})$ is differentiable at $\vec{p} = 0$ and that $f(\vec{x}, 0) = \vec{x}$. Then the columns of the matrix

$$\mathbf{L}_{\vec{x}} = \frac{df(\vec{x}, \vec{p})}{d\vec{p}}\bigg|_{\vec{p}=0} = \left(\frac{df(\vec{x}, \vec{p})}{dp_1}, \ldots, \frac{df(\vec{x}, \vec{p})}{dp_m}\right)\bigg|_{\vec{p}=0} \tag{3.127}$$

are tangential to $f(\,\cdot\,, \vec{p})$ at $\vec{x}$, and we can write the tangent plane as

$$\underline{\vec{x}}(\vec{p}) = \vec{x} + \mathbf{L}_{\vec{x}}\vec{p} \tag{3.128}$$

Now the tangent distance between two features $\vec{x}$ and $\vec{y}$ is defined as the shortest distance between their tangent planes (figure 3.11) as follows.

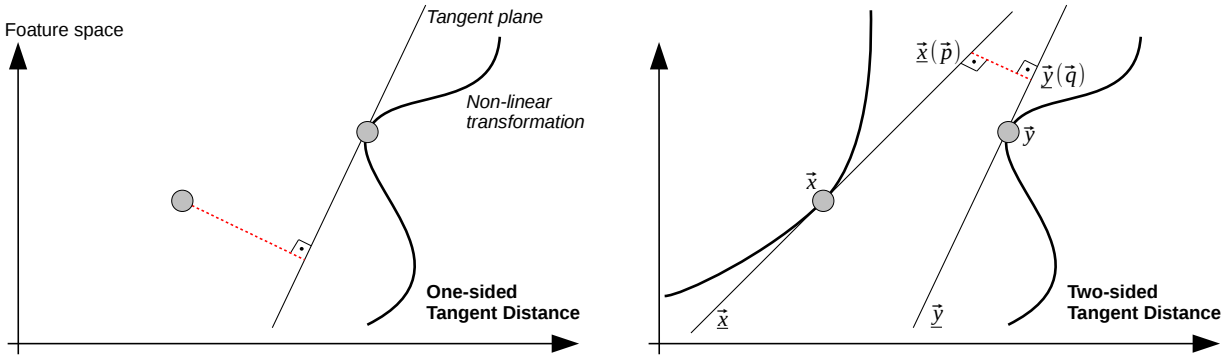


Figure 3.11: Instead of using the Euclidean distance between two features (grey circles) directly, we calculate the shortest Euclidean distance between all possible transformed features which differ only by a given transformation. Since transformations are commonly non-linear, we approximate them by calculating a tangent plane in feature space for one (left) or both features (right). Note that the sketched feature space has at least three dimensions, otherwise the tangent planes would intersect.

Definition 3.35 (Tangent Distance)**.** *Let $\vec{x}, \vec{y} \in \mathbb{R}$ be features with tangent planes $\underline{\vec{x}}, \underline{\vec{y}}$, respectively. Then the* two-sided *tangent distance is defined as*

$$TD_2(\vec{x}, \vec{y}) = \min_{\vec{x} \in \underline{\vec{x}},\, \vec{y} \in \underline{\vec{y}}} \|\vec{x} - \vec{y}\| \tag{3.129}$$

and the one-sided *tangent distance as*

$$TD_1(\vec{x}, \vec{y}) = \min_{\vec{y} \in \underline{\vec{y}}} \|\vec{x} - \vec{y}\|. \tag{3.130}$$

Note that the roles of $\vec{x}$ and $\vec{y}$ can be interchanged for the one-sided tangent distance.

For tangent planes

$$\underline{\vec{x}}(\vec{p}) = \vec{x} + \mathbf{L}_{\vec{x}}\vec{p} \tag{3.131}$$
$$\underline{\vec{y}}(\vec{q}) = \vec{y} + \mathbf{L}_{\vec{y}}\vec{q} \tag{3.132}$$

Simard et al. (1998) show that the parameters $\vec{p}$ and $\vec{q}$, which minimize the distance TD_2, are given by

$$\begin{aligned}(\mathbf{L}_{\vec{x}\vec{y}}\mathbf{L}_{\vec{y}\vec{y}}^{-1}\mathbf{L}_{\vec{y}\vec{x}} - \mathbf{L}_{\vec{x}\vec{x}})^{-1}(\mathbf{L}_{\vec{x}\vec{y}}\mathbf{L}_{\vec{y}\vec{y}}^{-1}\mathbf{L}_{\vec{y}}^T - \mathbf{L}_{\vec{x}}^T)(\vec{y} - \vec{x}) = \vec{p} \\ (\mathbf{L}_{\vec{y}\vec{x}}\mathbf{L}_{\vec{x}\vec{x}}^{-1}\mathbf{L}_{\vec{x}\vec{y}} - \mathbf{L}_{\vec{y}\vec{y}})^{-1}(\mathbf{L}_{\vec{y}\vec{x}}\mathbf{L}_{\vec{x}\vec{x}}^{-1}\mathbf{L}_{\vec{x}}^T - \mathbf{L}_{\vec{y}}^T)(\vec{x} - \vec{y}) = \vec{q},\end{aligned} \tag{3.133}$$

where $\mathbf{L}_{\vec{x}\vec{x}} = \mathbf{L}_{\vec{x}}^T\mathbf{L}_{\vec{x}}$, $\mathbf{L}_{\vec{x}\vec{y}} = \mathbf{L}_{\vec{x}}^T\mathbf{L}_{\vec{y}}$, and so on. Substituting the results back into equations (3.131) and (3.132), the two-dimensional tangent distance $TD_2(\vec{x}, \vec{y})$ is obtained. For the one-dimensional tangent distance TD_1 the solution from equation (3.133) simplifies to

$$\mathbf{L}_{\vec{y}\vec{y}}^{-1}\mathbf{L}_{\vec{y}}^T(\vec{x} - \vec{y}) = \vec{q} \tag{3.134}$$

which can, again, be substituted back into equation (3.132) to obtain the one-dimensional tangent distance. Since it is often not possible (or too complicated) to obtain $\mathbf{L}_{\vec{x}}$ analytically, its values can be approximated. In this work we will use a symmetric approximation of the tangent plane for rotations around an axis $\vec{v}$ as

$$\mathbf{L}_{\vec{a}} = \mathbf{R}_{\vec{v},-\alpha} \circ \vec{a}^L - \mathbf{R}_{\vec{v},\alpha} \circ \vec{a}^L = (\mathbf{R}_{\vec{v},-\alpha} - \mathbf{R}_{\vec{v},\alpha}) \circ \vec{a}^L, \tag{3.135}$$

where α is a small rotation angle. For rotation matrices around the X/Y/Z-axes, we derived formulas to calculate the matrix $\mathbf{R}_{\vec{v},-\alpha}$ from $\mathbf{R}_{\vec{v},\alpha}$ directly (section 3.7). Similarly, we approximate the tangent plane for translations $\vec{t}$ as

$$\mathbf{L}_{\vec{a}} = \mathbf{T}_{\vec{t}} \circ \vec{a}^L - \mathbf{T}_{-\vec{t}} \circ \vec{a}^L = \left(\mathbf{T}_{\vec{t}} - \mathbf{T}_{-\vec{t}}\right) \circ \vec{a}^L. \tag{3.136}$$

Again, we use equation (3.93) to calculate the translation as a concatenation of rotations and a translation along the Z-axis. This allows us to calculate the translation matrix $\mathbf{T}_{-\vec{t}}$ directly from $\mathbf{T}_{\vec{t}}$ (theorem 3.30) which nearly halves the computation time.

CHAPTER 4

Localization

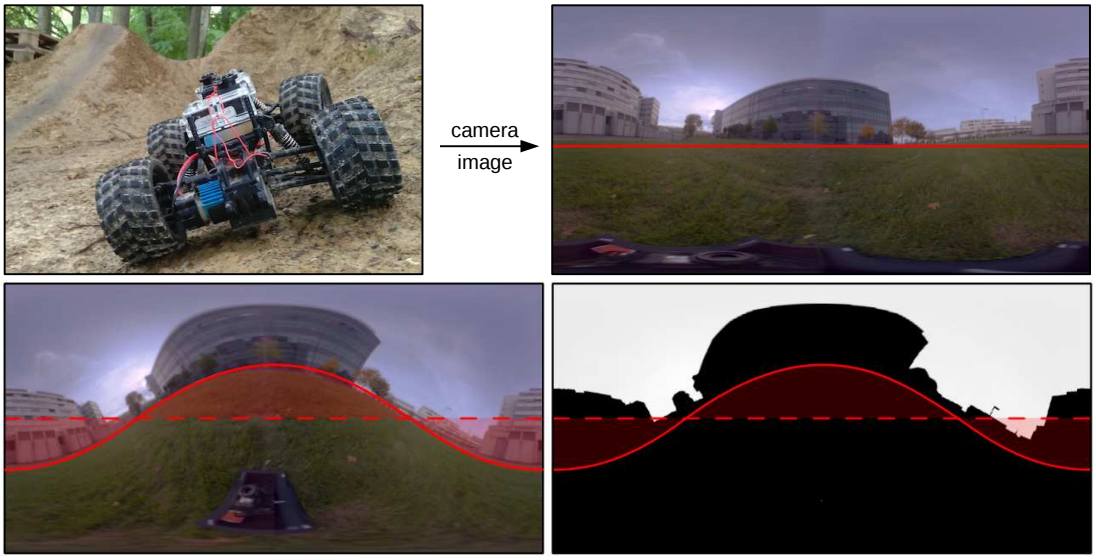
The localization of autonomously driving agents is a crucial step to accomplish more complex navigational tasks like route following. In this chapter we describe a method to localize an agent on a previously driven track in an outdoor environment. The key idea is to extract a binary panoramic skyline image as an illumination-invariant scene descriptor. This is accomplished by by a UV-only camera using a UV-filter and a fish-eye lens. The skyline image classifies each pixel either as ground object or sky, independent of the illumination conditions. By projecting the skyline image to the unit sphere, we can express the skyline in frequency domain and use its amplitude spectrum (or alternatively its bispectrum) as scene descriptor. By using a sequence-based method (here: seqSLAM), we could show that these scene descriptors are representative even in repetitive environments like forests. The scene descriptors are — due to the rotation-invariance of the amplitude spectrum — invariant to rotations and can be used to perform localization on bumpy terrain. Our method is finally compared to the original implementation of seqSLAM as well as the feature-based method FABMAP (both using RGB fish-eye images as input) in two tests: First, we performed localization on a track where the tilt of the camera between the training and the test run is systematically increased. Second, we performed localization on tracks in challenging environments, e.g. on a BMX-track. As we could show, our localization method outperformed the original seqSLAM and FABMAP on all tested tracks. This chapter is mainly based on Stone et al. (2016), a collaboration with Thomas Stone (University of Edinburgh), Barbara Webb (University of Edinburgh), and Michael Milford (Queensland University of Technology).

4.1 Introduction

Localization of autonomously driving wheeled robots or micro air vehicles is a challenging task and has various applications as for example in autonomous cars (Lowry et al., 2016), lawn mowing (Yang et al., 2015), and disaster relief (Matthies et al., 2002). A reliable localization of an agent provides important information for several navigational tasks, e.g. the localization of cars in areas without GPS signal or self driving service robots. Here, we refer to localization as the task to determine the agent's location on a previously driven track (a more detailed introduction to localization can be found in section 1.2.1): An agent is manually driven along a track and collects scene descriptors from its sensory input for each visited location (training run). Afterwards, the agent is driven along the track a second time, but this time the location of the agent should be estimated by comparing the scene descriptor of the current location with the scene descriptors from the training run (test run).

Several visual localization methods (review: Lowry et al. (2016)) have been suggested which use visual features as scene descriptors (section 1.2.2.2). Visual features often allow precise pose estimations of an agent moving in suitable environments, however a major drawback of visual features is that they rely on high-resolution images and are susceptible to illumination and appearance changes of the environment. Moreover, methods based on visual features are usually computationally expensive. Holistic methods (section 1.2.1) are — due to the comparably large amount of image information used — partially invariant to illumination and appearance changes

Example 4.1: Overview



A skywards oriented camera is mounted on a RCC (top left) to capture panoramic images (top right) using a fish-eye lens with an opening angle of 180° (red line). If the agent is tilted, the panoramic image is distorted (bottom left). Due to the limited opening angle of 180°, areas which are not visible in either one or the other panoramic image appear (solid/dashed red lines, red areas). These areas increase the difficulty to match two differently tilted panoramic images recorded at the same location, leading to false matches in the localization process. In our approach, we extract the skyline and fill the lower hemisphere with 'ground' (bottom right). Since regions slightly above the horizon are commonly dominated by ground objects as houses or trees, the error introduced by tilt is reduced.

and are computational cheap, but often limited in their degrees of freedom (e.g. to planar movement).

In this chapter, we propose a biologically inspired approach for localization from the study of insects, e.g. of the desert ant *Cataglyphis bicolor*: In chapter 2 we already discussed that desert ants apparently use the skyline — a binary panoramic image classifying objects on the ground and the sky — as illumination-invariant scene descriptor (Möller, 2002, Graham and Cheng, 2009a). As suggested by Mote and Wehner (1980), the eyes of the desert ant are sensitive to UV light which provides sufficient information to extract the skyline using local separation techniques. The process of skyline extraction using the UV channel is explained in detail in section 4.3.1. Desert ants are also well known for their ability to perform complex navigational tasks despite the tilt induced by legged motion or movement over uneven terrain (Ardin et al., 2015). Inspired by the desert ant, we suggest a localization method which on the one hand uses the skyline as an illumination-invariant scene descriptor and on the other hand is rotation-invariant.

Panoramic images, as captured by a skywards oriented camera equipped with a 180° fish-eye lens, are shown in example 4.1. As can be seen, an advantage of the skyline over a standard panoramic image is that the image information is invariant against tilt as long as the sky is completely visible. This is often the case since the sky is commonly covered by buildings, trees, or similar. We propose to apply Fourier analysis on the sphere (chapter 3) to calculate the amplitude spectrum of the skyline as rotation-invariant scene descriptor. By combining the amplitude spectrum of the skyline with the sequence-based method seqSLAM by Milford and Wyeth (2012), we obtain a reliable localization method for outdoor environments.

(a) (b)

Figure 4.1: The figure shows the two experimental setups used to record the training and test data. (a) One camera (either UV or RGB) is mounted on a helmet with a tilt angle α above the horizon. The track has been recorded using both the UV and the color camera for varying tilt angles $\alpha \in \{-30°, -20°, \ldots, 30°\}$. (b) Both cameras are mounted on the RCC, allowing us to record the same run with both cameras simultaneously.

Training and test data have been collected on three different tracks to compare our method (in the following called **seqSLAM (sky)**) with the unmodified method seqSLAM (Milford and Wyeth, 2012) (**seqSLAM (vanilla)**) and the feature-based method **FABMAP** (Cummins and Newman, 2009). The first track contains training and test data recorded by bike in Bielefeld city using a helmet camera with adjustable tilt angles. These data can be used to systematically compare the quality of all three tested localization methods for an increasing tilt angle. The other two tracks were recorded using a remotely controlled car (RCC) in highly repetitive areas with bumpy terrain. We could show that our method showed superior performance on both tracks. Finally, we examine the influence of the maximal number of bands, the performance on RGB images (especially for cross-database tests), and the bispectrum as an alternative to the amplitude spectrum.

The idea of using the amplitude spectrum of the skyline as scene descriptor was developed independently by Thomas Stone and Dario Differt. Michael Milford and Thomas Stone combined seqSLAM with our scene descriptors to increase the reliability of our localization method. The training and test data were collected by Dario Differt and evaluated by Thomas Stone. The visuals were created by Dario Differt. The study was supervised by Michael Milford and Barbara Webb. Independently of the collaboration, a comparison between the amplitude spectrum and bispectrum has been carried out (section 4.4.6) by Dario Differt.

4.2 Experimental Setups

For the experiments conducted in this chapter we use two identical cameras (*GoPro Hero 3+ Black*). Both cameras are mounted with $1/3.2''$ panoramic fish-eye lenses (*RageCams*), each with a focal length of 1.19 mm and 185° field of view. The lenses are equipped with different glass filters: The first lens uses an IR cut-off filter to record LDR color images, while the second lens uses a glass filter sensitive to UV-only with a peak response at around 350 nm (*Skyline Sensors*). Unfortunately, for the latter no explicit transmission spectrum can be shown since it is not disclosed to the public[1]. In all experiments, both cameras are using the same settings: The video resolution is set to '1080p super view' with a frame rate of 24 FPS and automatic adjustment of the exposure time. Only images with a single exposure time were collected, HDR imaging techniques (section 2.2.4) were not applied.

We used two different experimental setups to collect our datasets: For the first experimen-

[1] As experiments show, the cut-off at 350 nm does not block all light with a higher wavelength; for example the infrared emitter of a remote control is slightly visible.

tal setup, we equipped a helmet with a skywards oriented camera (figure 4.1 (a)). The tilt of the camera can be adjusted to systematically collect training data with varying tilt angles $\alpha \in \{-30°, -20°, \ldots, 30°\}$. To collect both UV and color images, the same track was consecutively driven two times with each camera for each angle α. The second experimental setup consists of the RCC (*Truck Fighter 3* by *Drive&Fly-models*) without chassis but equipped with both the UV and color sensitive cameras simultaneously (figure 4.1 (b)).

4.3 Method

Example 4.2: Matching of Tilted Panoramic Images

Left: Two panoramic UV images (fish-eye images) recorded at the same location in the *City* dataset, but with a 60° tilt offset from each other. The houses (red and white squares) appear differently between both images due to the strong distortion induced by the tilt. This results in poor localization estimates for methods which are not rotation-invariant, e.g. seqSLAM (vanilla). Right: Only the skylines of both images are shown. Using their rotation-invariant amplitude spectra, both panoramic images are represented by similar scene descriptors and provide a match during the localization process.

In this section we introduce a novel visual localization method. This method is based on the illumination-invariant skyline as scene descriptor. Instead of using the skyline directly, we calculate its rotation-invariant amplitude spectrum. Example 4.2 shows the benefits of using amplitude spectra as scene descriptors to match two views which strongly differ in their visual appearance due to the strong camera tilt.

4.3.1 Skyline Extraction

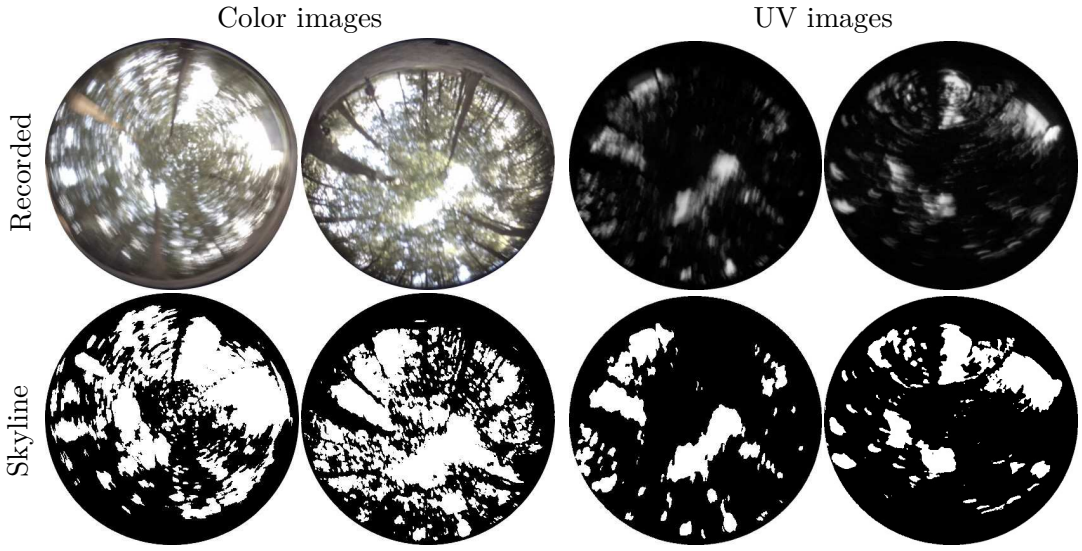
In chapter 2 we showed that for images recorded with a UV-only camera, objects on the ground can be separated from the sky using simple thresholding methods. For the experiments performed in this chapter we use a fish-eye camera sensitive to UV-only to capture panoramic images (figure 4.1). On each captured image, we apply the local separation technique *Otsu* (section 2.2.8.2) to obtain a binary panoramic image (example 4.3)[2]. We refer to these binary panoramic images — in which each pixel is classified either as object on the ground or as sky — as **skylines**. Pixels which are not visible in the camera image (e.g. pixels under the horizon) are set to ground.

4.3.2 Scene Descriptors

After extracting the skyline from the camera input (section 4.3.1), we use the coordinate system shown in example 3.4 to describe the skyline as a real-valued function $f : S^2 \to \mathbb{R}$ on the unit sphere S^2. To obtain the Fourier coefficient vector $\vec{a}^L$ of f for the first L bands, we use 4000 equally distributed sample points on the sphere to perform the Fourier transform (section 3.10.1). Finally, we calculate the amplitude spectrum $\vec{\mathcal{AS}}_f$ (section 3.9.2), which we use as scene descriptor

[2] In ongoing experiments, we improved the skyline extraction by using the local separation technique NA_λ (section 2.2.8.3) and by dividing the panoramic image into multiple segments (section 2.3.7). Unfortunately, these techniques were not yet available at the time this experiment was carried out.

Example 4.3: Skyline Extraction



The top row shows raw panoramic images (fish-eye images) captured with the color sensitive camera (left) and the UV sensitive camera (right). The images are recorded at roughly the same location during the training and test run. As can be seen, both the UV and color images suffer strongly from motion blur, making it difficult to extract visual features. The bottom row shows the same images after applying the local separation method *Otsu* to extract the skyline (bottom row). Contrary to the noisy and tattered appearance of the skylines extracted from the color images, the skylines from the UV images show only a few but clearly distinguishable sky regions.

for our localization method. The maximal number of bands was set to $L = 120$ such that each scene descriptor has a total of 120 entries (480 byte using 32 bit-floats) per image.

To compare two different scene descriptors, we need to define a distance measure for amplitude spectra. We use the distance measure AS from definition 3.32 and denote in the following by $AS(i, j)$ the distance between the i-th and j-th images of the training and test run, respectively.

4.3.3 Sequence SLAM

To motivate the idea of using image sequences for visual localization, assume that we initially try to use a *single* scene descriptor to localize the agent during the test run. This scene descriptor is mostly sufficient to decide if the current location looks similar or different compared to all locations on the training run. However, scene descriptors collected at locations close to each other or in similar environments are often similar. This might lead to problems if the appearance of the scene changes between the records of the training and test runs (e.g. by illumination changes or moving objects) such that the exact location on the track cannot be determined. In the worst case it can happen that a scene descriptor of a wrong location is matched with the current camera image (false positive). To address this problem, we use the sequence-based method seqSLAM (Milford and Wyeth, 2012) which does not compare two single scene descriptors $\vec{\mathcal{A}}_i^T$ and $\vec{\mathcal{A}}_j$ with each other, but the scene descriptors of two sequences: Let $\{\vec{\mathcal{A}}_i^T\}_{i_1 \leq i \leq i_2}$ and $\{\vec{\mathcal{A}}_j\}_{j_1 \leq j \leq j_2}$ be sequences of scene descriptors with the same length in the training and test run, then we can calculate the distance between both sequences by summing up the single distances between the subsequent pairs of scene descriptors. The original implementation of seqSLAM (**seqSLAM (vanilla)**) uses histogram equalized RGB camera images as scene descriptors. Our implementation (in the following called **seqSLAM (sky)**) uses the amplitude spectrum of the skyline as scene descriptors

instead. The seqSLAM algorithm itself uses the default parameters taken from Milford and Wyeth (2012).

4.3.4 FABMAP

A popular feature-based approach for visual localization is to use the bag-of-words method (Wallach, 2006). In this method, local features are extracted from a set of RGB images and clustered based on a similarity measure. Each cluster is called a word, the set of all clusters the bag-of-words. The bag-of-words can then be used to create a scene descriptor for any new image: First, local features are extracted from the image (section 1.2.2.2). Second, each extracted local feature is compared with each word in the bag-of-words and for the closest match a counter (corresponding to this word) is increased. Finally, after each local feature in the image has been assigned to a word, the image is represented by a histogram with the same number of entries as the number of words in the bag-of-words. Note that the training data used to create the bag-of-words does commonly not contain data from the training runs since over-fitting might occur. Instead, the data used to learn the bag-of-words should consist of additional runs collected in comparable environments.

For comparison with seqSLAM (vanilla) and seqSLAM (sky) we used the implementation OpenFABMAP[3] (Cummins and Newman, 2009). OpenFABMAP depends on a wide set of parameters, including the choice of a feature detector and descriptor: We used the standard parameters provided in the sample script. Furthermore, we used SURF (Bay et al., 2008) for both feature detection and description on panoramic images with a resolution of 480×270 pixel.

4.3.5 Datasets

We used the experimental setups from section 4.2 to record training and test runs in three different environments (example 4.4). The datasets were recorded at sunny days during September 2015. Each dataset was recorded two times, once at midday and once in the early evening, to increase the visual dynamics in the scene (e.g. illumination changes). In both the test and training runs the tracks were driven in the same direction. The first dataset *City* was recorded by bike in the city of Bielefeld using the camera mounted on the helmet on a track of approximately 520 m during the afternoon. The environment was highly dynamic with varying light conditions and multiple cars and people interfering with the recording process; either as appearance in the recordings or by altering the course driven with the bike to evade collisions. For both cameras (UV and color) we collected eight records with varying tilt angles $\alpha \in (-30, -20, \ldots, 30)$ — the track has been recorded twice for $\alpha = 0$ — allowing us to test the different localization methods on training and test runs with tilt of up to 60°. The second and third datasets were collected using the RCC in the vicinity of Bielefeld university. The second dataset *BMX Track* has a length of approximately 600 m and can coarsely be divided into three parts: The track starts on a BMX track in a forest, providing bumpy terrain and slippery ground, before entering a path enclosed by forest on one side and gardens on the other side. Finally, the track passes through a suburban area. The third dataset *Disposal Site* was recorded in Borgholzhausen on a highly repetitive track of approximately 510 m on a road through the forest. On this track only a detour to a disposal site of less than 100 m and a junction provide artificial structures.

To reduce the data load, we only used 5 frames per second (instead of the recorded 25). As ground truth we manually identified for each recorded track keyframes which allow a reliable match between multiple runs on the same track. The keyframes do not necessarily have to be captured at the same location during each run — especially since it was not possible to drive a track exactly in the same way as before — but are chosen at moments where a salient landmark was passed. Matches of frames between keyframes were linearly interpolated. For the evaluation of the tested methods we consider a match between training and test run as correct if it is within

[3] The OpenFABMAP implementation, version 2.02, was downloaded from `https://code.google.com/archive/p/openfabmap/`

Example 4.4: Recorded Tracks

Perspective images (only for visualization) taken at the locations where the training and test data were collected. The visual appearance of the locations differs strongly and contains landmarks typical for urban as well as rural scenes.

Aerial imagery of the tracks on which the training and test data were collected. The route driven by bike (left: City) or the RCC (middle, right: BMX Track, Disposal Site) as well as the moving direction is indicated by red arrows. Map data from *Google, Geobasis-DE/BKG (@2009).*

±3 seconds of the ground truth.

4.4 Results

In this section we evaluate the performance of the three localization methods seqSLAM (vanilla), seqSLAM (sky), and FABMAP. In the first experiment we evaluate the performance of these methods by systematically increasing the tilt angle between the training and test run. The second experiment is performed using a RCC on challenging tracks in highly repetitive environments.

4.4.1 Precision versus Recall Plots

In the following we use **precision versus recall** plots to compare the performance of the localization methods. Let us denote by X a set of matches — which can be correct or false — found by a method. Then the precision value is the number of correctly matched frames in X. The recall value is the size of X divided by the total number of frames available in the test run. The set X is not chosen arbitrarily, but contains the frame matches with the highest possibility to be correct; both seqSLAM (vanilla/sky) and FABMAP have internal (probability-based) methods to

training run	**(α_1)**	0°	10°	10°	20°	20°	30°	30°
test run	**(α_2)**	0°	0°	−10°	−10°	−20°	−20°	−30°
total	**(α)**	0°	10°	20°	30°	40°	50°	60°

Table 4.1: The dataset *City* (variable tilt) was recorded with varying tilt angles α_1 and α_2 applied to the camera during the training and test run, respectively. By combining the runs as shown in the table, we can compare the performance of seqSLAM (vanilla) and seqSLAM (sky) on runs with increasing tilt angles of up to 60°.

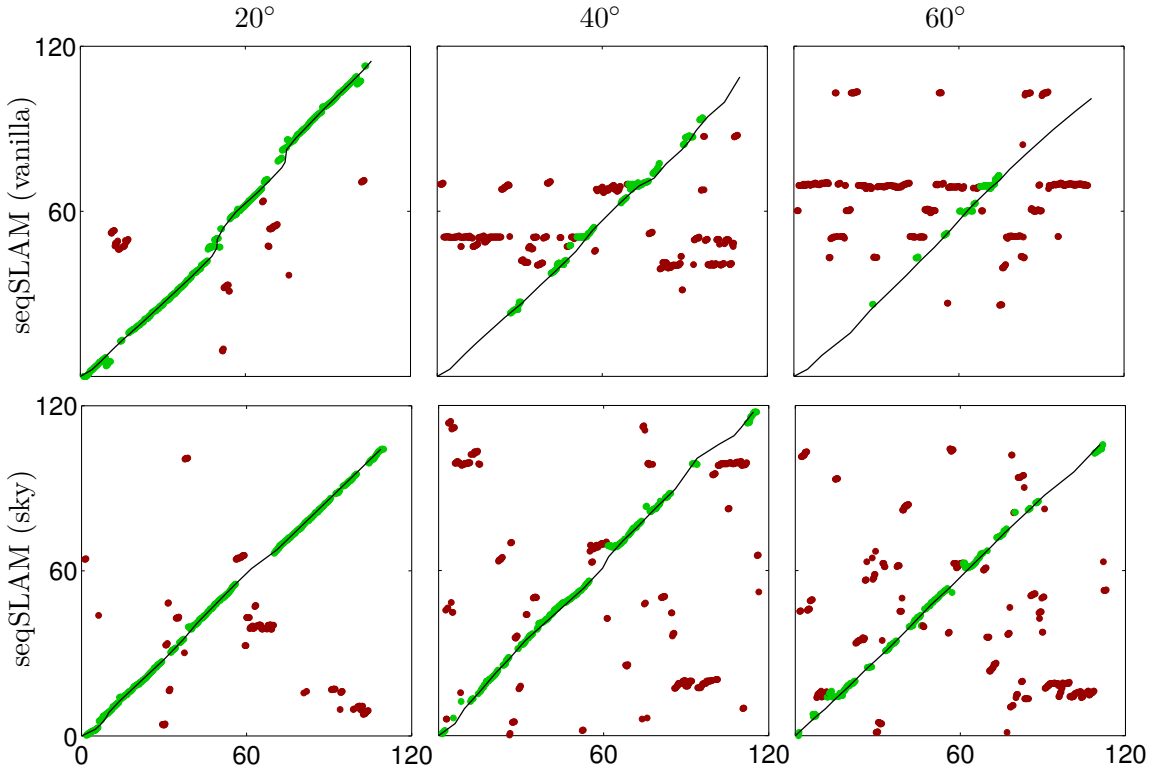


Figure 4.2: Each plot shows the matches obtained by seqSLAM (vanilla) and seqSLAM (sky) for tilt angles of 20°, 40° and 60° (compare table 4.1) on the *City* track. The X/Y-axes show the time passed in seconds since the record started for the training and test run, respectively. Each dot represents a match of the test run with the training run, its color depicts if the match is correct (green) or false (red) considering the ground truth (black line).

estimate this possibility for each frame match. As a consequence, with an increasing recall value the set X contains more matches which are likely false. For example, a precision value of 60% at a recall value of 40% means that for the best 40% of all matches (as estimated by seqSLAM (vanilla) and FABMAP) 60% were correctly matched. A perfect match between the training and test run would result in a precision versus recall plot with 100% precision for all recall values.

4.4.2 City Dataset with Tilt Variation

Using the *City* dataset, we are able to systematically evaluate the performance of seqSLAM (vanilla) and seqSLAM (sky) for increasing tilt between the data recorded during the training and test run. Table 4.1 shows which training and test runs have been combined to achieve tilt of up to 60° between camera images of the training and test runs. Figure 4.2 shows the matches between the training and test runs for varying tilt angles α. As can be seen, for low tilt angles $\alpha = 20°$, the matches obtained by seqSLAM (vanilla) and seqSLAM (sky) are roughly comparable. Even though the number of mismatches increases for both methods with an increasing tilt angle, seqSLAM (sky) performs noticeably better than seqSLAM (vanilla): For $\alpha = 60°$, seqSLAM

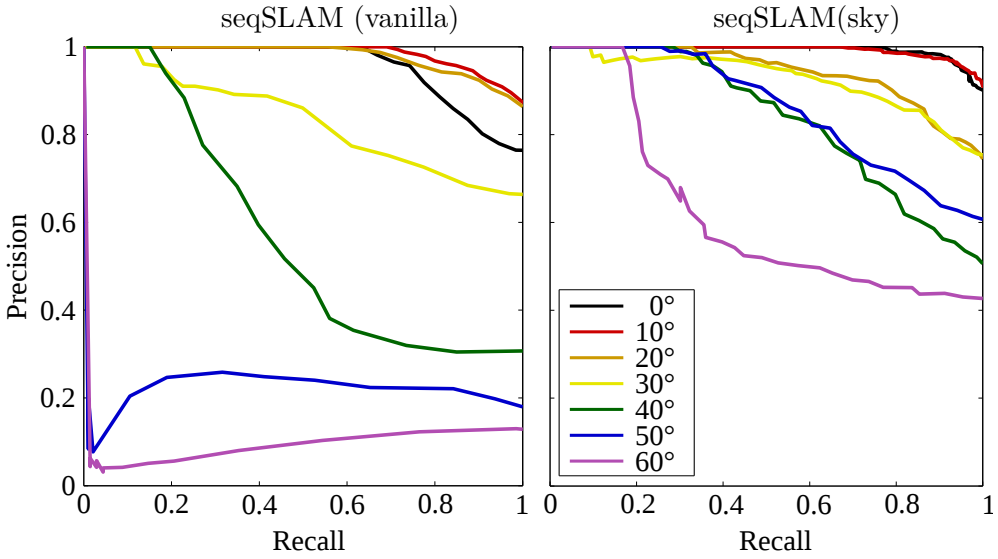


Figure 4.3: The precision versus recall plots are shown for the methods seqSLAM(vanilla) and seqSLAM(sky) and tilt angles α between 0° and 60°. The tilt angles α are obtained by combining two runs with varying tilt angles α_1 and α_2 (table 4.1).

(sky) is able to find correct frame correspondences between the training and test run for most parts of the track, while seqSLAM (vanilla) is only able to find correspondence for small intervals mainly between 60 s to 100 s. The reason for the increased error is different for both methods: While seqSLAM (vanilla) suffers from distortion and rotational misalignment between images, the cropped skylines — which occur more frequently with increasing tilt — induce errors for seqSLAM (sky).

Figure 4.3 shows a precision versus recall plot for various tilt angles and confirms the visual impression of figure 4.2: As can be seen, over the complete track (i.e. at 100% recall) the performance of seqSLAM (vanilla) is comparable with seqSLAM (sky) for tilt up to 30° (66% vs 76% correct matches). However, the performance of seqSLAM (vanilla) noticeably decreases for tilt angles of 40° (31% correct matches), while seqSLAM (sky) still finds 53% correct matches. Moreover, it can be seen that for tilt angles of 50° or higher and a precision of 100%, seqSLAM (sky) is able to achieve a recall of around 20%, while seqSLAM (vanilla) completely fails (recall 0%).

4.4.3 BMX Track Dataset

The *BMX Track* (example 4.4) consists of three consecutive parts which strongly differ in their visual appearance: During the first part the RCC drives through a forest and the frequently appearing bumps — including spacious sinks and steep hillocks — cause significant tilt. The second and third parts provide many landmarks like houses and cars, however the low sun is leading to intense lens flare. While lens flare only appears rarely on the first part, the lens flare is particularly strong in the forest due to the longer exposure times (figure 4.5).

Examining the results of FABMAP shown in figure 4.4, it can be seen that on the *BMX track* the matched frames form three 'blocks', each belonging to one part of the track. This indicates that the feature-based methods FABMAP is able to distinguish between the three different parts of the track, however it is not able to pin down the exact location of the RCC. From visual inspection, the methods seqSLAM (vanilla) and seqSLAM (sky) are able to localize the RCC on most parts of the track and show comparable performance (figure 4.4, top middle, top right). Regarding the precision versus recall plot in figure 4.5 (left), qualitative differences become more visible: The number of correct matches found by FABMAP on the complete track (recall 100%) is below 50%, while seqSLAM (vanilla) achieves 70%. The best result is obtained by seqSLAM (sky) with 84%. For both methods seqSLAM (vanilla) and seqSLAM (sky), a precision of 100%

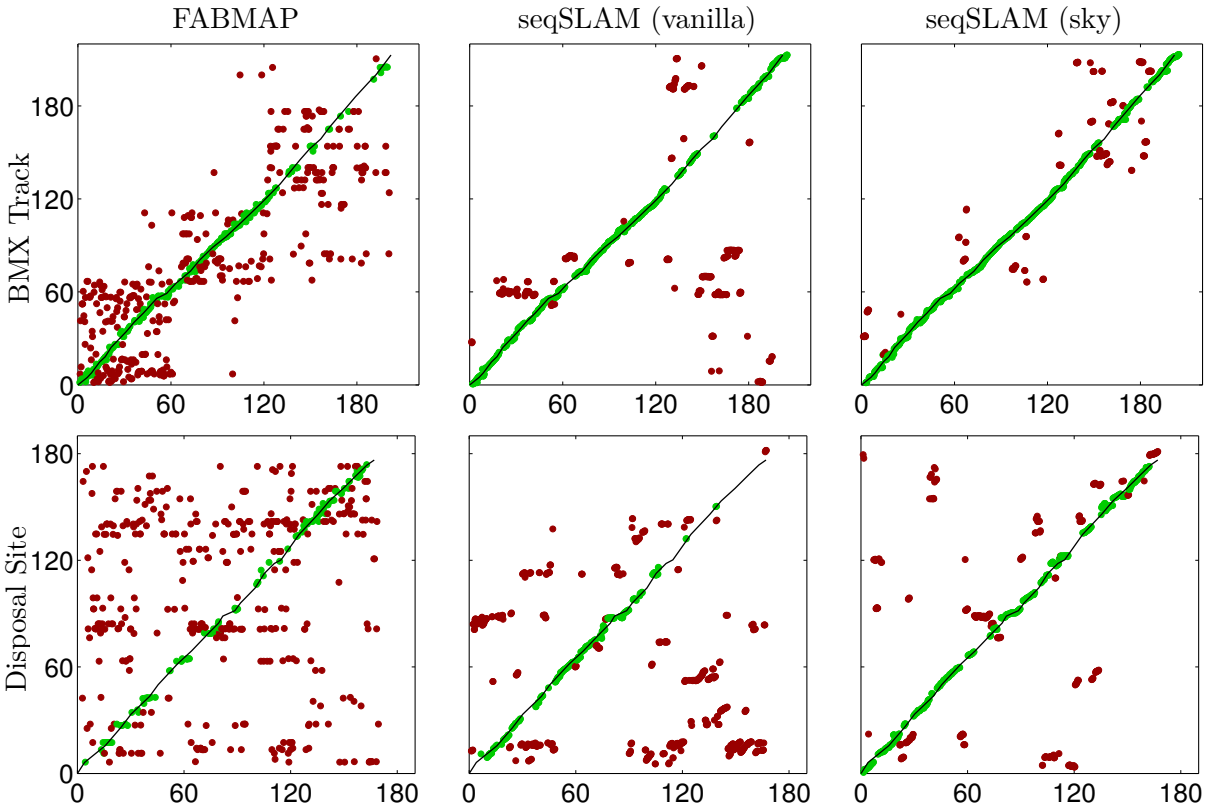


Figure 4.4: The figure shows the frame matches found by the three localization methods FABMAP, seqSLAM (vanilla), and seqSLAM (sky) on the tracks *BMX Track* and *Disposal Site*. For details, see figure 4.2.

can be reached for around 40% of the matches.

4.4.4 Disposal Site Dataset

The track *Disposal Site* is highly repetitive and artificial objects as houses appear only on the first half of the track (example 4.4). The images recorded in the second half are dominated by trees appearing equally on both sides of the street. The RCC was driven aggressively, including zigzagging and high speeds, increasing blur in the images. Apart from that, the RCC is (nearly) not tilted on the track and lens flare does not appear.

Due to the high similarity of most visual features (especially trees), the difficulty to successfully find correct matches increases. The frames matched by FABMAP appear randomly chosen and do not provide any reliable information (figure 4.4, bottom left). While seqSLAM (vanilla) is able to find correct frame correspondences for the first half of the track, nearly no correct matches can be found on the second half (figure 4.4, bottom middle). The best results were achieved by seqSLAM (sky) which correctly matches frames along the complete track (figure 4.4, bottom right). Overall, all tested localization methods performed better on the *BMX Track* than on the *Disposal Site* (figure 4.5): FABMAP and seqSLAM (vanilla) only matched 20% and 34%, respectively, of all frames correctly; most correct matches were found by seqSLAM (sky) with 63%. For all three localization methods a precision of 100% can only be obtained at a recall of around 5%.

4.4.5 Tilt-Invariance versus Sequence Length

Figure 4.5 shows the influence of the sequence length used for seqSLAM (sky) for varying values $r \in \{1, 10, 20\}$. By increasing the sequence length to $r = 20$, the total number of correctly matched frames could be increased from 84% to 89% and from 63% to 70% for the tracks *BMX*

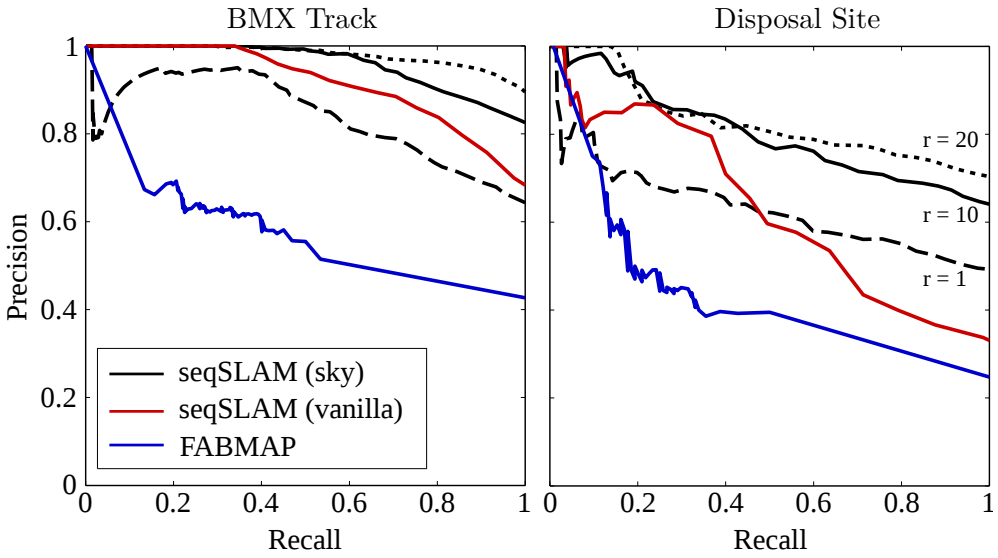


Figure 4.5: Precision versus recall plots for the three localization methods FABMAP, seqSLAM (vanilla), and seqSLAM (sky) on the tracks *BMX Track* and *Disposal Site*. For seqSLAM (sky), the precision versus recall plots are shown for sequence lengths of $r = 1$ (dashed), $r = 10$ (solid, default value), and $r = 20$ (dotted). For details, see figure 4.3.

Track and *Disposal Site*. Moreover, the recall rate is improved for both tracks meaning that less false positives are found. By reducing the sequence length to $r = 1$ the number of correctly matched frames on the complete tracks decreases to 64% and 49% for *BMX Track* and *Disposal Site*. Moreover, the recall rate is worsened strongly.

Note that a sequence length of $r = 1$ is equivalent to using the amplitude spectrum of the skyline as a single scene descriptor. Although the amplitude spectrum contains only a small fraction of the skyline information, the number of correctly matched frames is surprisingly high. This indicates that even a single amplitude spectrum of the skyline is a distinctive scene descriptor usable for visual localization.

4.4.6 Comparison of the Amplitude Spectrum and Bispectrum

In the previous sections, we performed localization experiments on different outdoor tracks. However, the scene descriptors used for localization were chosen application-specific, i.e. we used the amplitude spectrum with a maximal number of $L = 120$ since it showed good performance on the collected databases. The computation time to calculate the amplitude spectrum of a panoramic image is around 23 ms on an Intel(R) Core(TM) i7 CPU 870 @2.93 GHz (single core). In this section we discuss more systematically alternative parameter settings as well as the bispectrum (section 3.9) as an alternative for the amplitude spectrum. In comparison to the phase-invariant amplitude spectrum, the bispectrum is phase-variant and therefore contains more information than the amplitude spectrum. However, the calculation of the bispectrum is demanding regarding both computational power and memory usage (precalculation of the real coupling matrices, section 3.6). Therefore the question arises if the bispectrum provides additional information for localization tasks which justify the increased computational costs.

Besides the comparison between the amplitude spectrum and the bispectrum, we also examine the influence of the maximal number of bands L used. Recalling chapter 3, the maximal number of bands L used to Fourier transform a function in the basis of RSH and the maximal number of bands L_{CG} used to precalculate the real coupling matrices (necessary to calculate the bispectrum, sections 3.5.5 and 3.9.3) can be set arbitrary. We are therefore interested in the impact of the maximal number of bands L and L_{CG} on the information contained in both the amplitude spectrum and bispectrum.

In the experiments performed throughout this chapter we used skyline-segmented panoramic

images for localization. Alternatively, common LDR panoramic images can be used. In contrast to skyline-segmented images, LDR images are not illumination-invariant but contain more information (8 bit gray values instead of 1 bit binary values). Additionally to use raw LDR images, we also apply various image preprocessing techniques (Sobel filter, histogram equalization, and the concatenation of both, section 3.11.2) on cross-databases. In contrast to normal databases, for each location in a cross-database multiple images were recorded under varying illumination conditions.

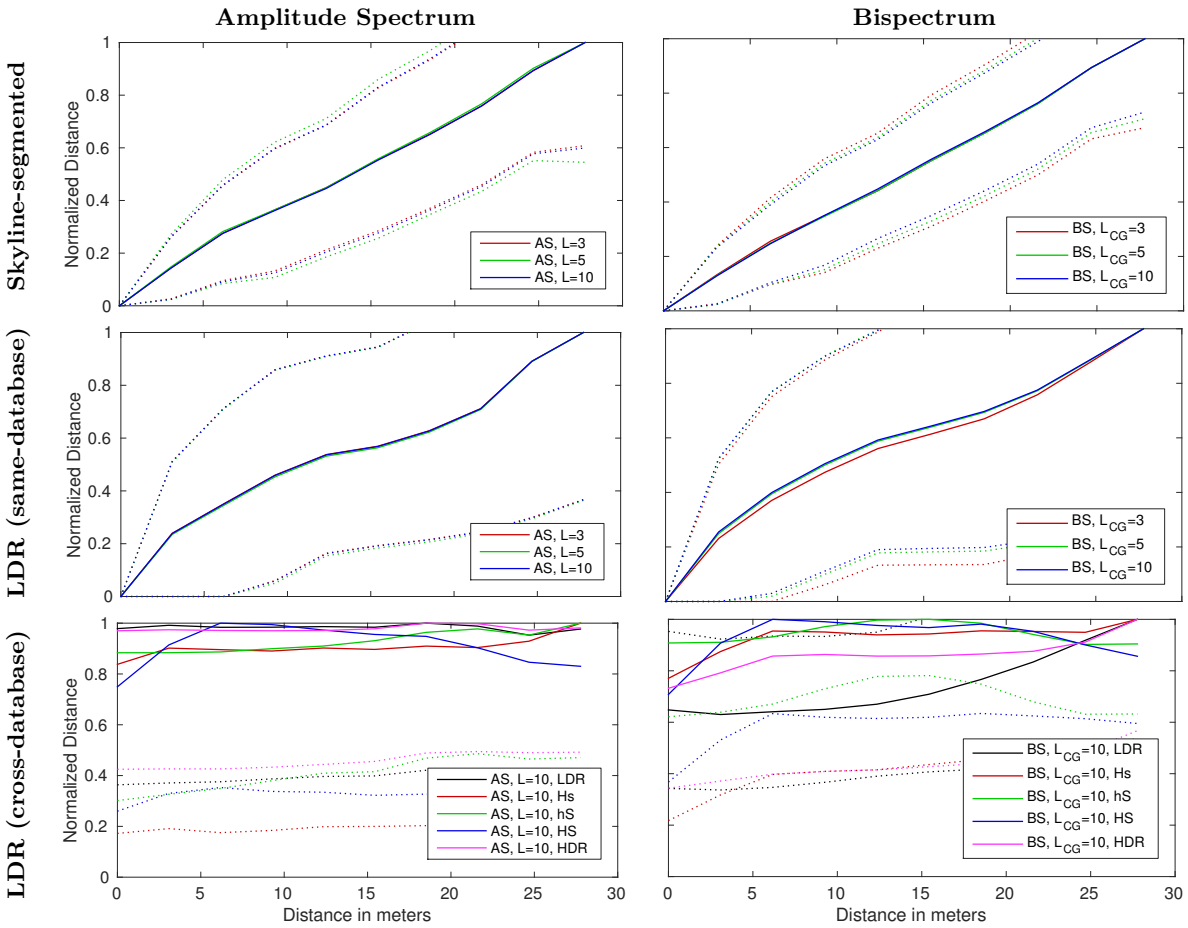


Figure 4.6: The figure shows the distance measures AS (amplitude spectrum, left column) and BS (bispectrum, right column) of two panoramic images as a function of their metric distance (translation in meters). The plots are histogram-based (approximately 3 m bins) and each plot is normalized to the interval $[0, 1]$. The mean value (solid lines) and the standard derivation (dotted lines) are shown. The skyline-segmented (top row) and LDR (middle and bottom row) panoramic images are taken from the databases *uni_early*, *uni_late*, and *uni_winter* (appendix D). For each plot, different values for the maximal number of bands L, L_{CG} and different preprocessing techniques (hs, Hs, hS, HS, section 3.11.2; HDR, section 2.2.4) are used. The same-database experiments were carried out on each database individually and the results were pooled afterwards. For the cross-database experiments, the tests were performed on each combination of two different databases and the results were pooled afterwards.

To compare the impact of each setting (amplitude spectrum versus bispectrum, influence of number of bands, and skyline-segmented versus LDR images) we calculated the distances measures $AS(f, g)$ (definition 3.32) and $BS(f, g)$ (definition 3.33) for pairs of panoramic images represented by functions f, g. The panoramic images are taken from the databases *uni_early*, *uni_late*, and *uni_winter* which were captured on a street parallel to the Bielefeld university main building; a detailed description of the databases together with example images can be found in appendix D. Across the databases, the records were collected at the same locations and same orientations

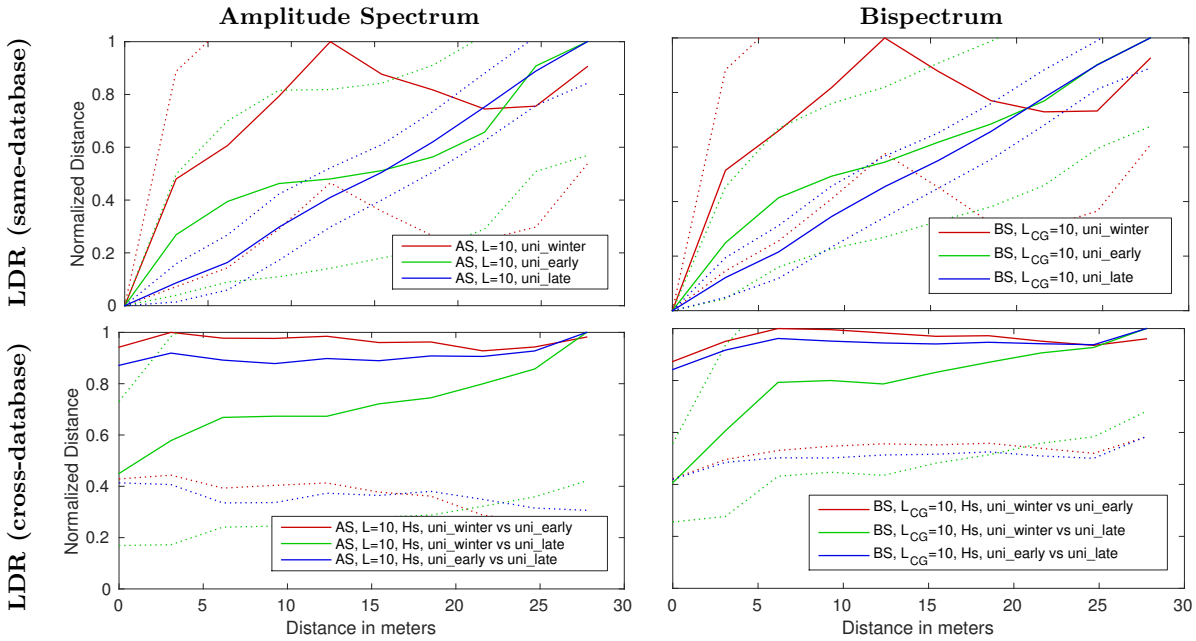


Figure 4.7: The figure shows the distance measures AS (amplitude spectrum, left column) and BS (bispectrum, right column) of two panoramic images as a function of their metric distance (translation in meters). The plots are histogram-based (approximately 3 m bins) and each plot is normalized to the interval $[0, 1]$. The mean value (solid lines) and the standard derivation (dotted lines) are shown. The results of the same-database (top row) and cross-database (bottom row) tests are shown for each single database and database combination, respectively.

on a rectangular 20×4 grid covering an area of 27.4 m $\times$ 4.7 m. Due to the differing dates and daytimes of record, the appearance of the environment (e.g. the university main building) differs significantly between panoramic images of the three databases. Finally, we manually created a skyline-segmented image for each panoramic image from the database *uni_early*. Note that we did not repeat this procedure for the other two databases, instead we assume — following the results of chapter 2 — that the skyline is congruent for all three databases and could be extracted using a UV camera. Since the location of each panoramic image in our databases is known, we can plot the distance measures AS and BS as a function of the metric distance between the locations of the panoramic images. We calculated the amplitude spectrum and bispectrum for three different types of input images, each with an opening angle of 220°: First, we used the skyline-segmented images as input. Second, we performed *same-database tests* on LDR images, i.e. we used the databases *uni_early*, *uni_late*, and *uni_winter* independently as input. For same-database tests, the exposure times for all images of a database are constant and can be found in the appendix, table D.1. Third, we performed *cross-database tests* on LDR images, i.e. we used a pair of different databases as input (e.g. *uni_early* for current views and *uni_late* for snapshots). For the skyline-segmented and LDR images, we filled the missing areas in the lower hemisphere as before with 'ground' or white noise, respectively. The results are summarized in figure 4.6 and 4.7.

At first sight, for the skyline-segmented images and the same-database tests, the mean distance measures are similar for both the AS and BS; only minor differences can be spotted for the standard deviation which is slightly smaller for the BS. For skyline-segmented images, both distance measures AS and BS increase nearly linear with the metric distance in meters. For the same-database tests, this relation is non-linear and increases faster for the first 5 m and slower after around 10 m. Moreover, the standard derivation is strongly increased for both the AS and BS measure. For the cross-database tests it can be seen that the measures AS and BS perform slightly different, however both fail to establish a reliable relation to the metric distance between two images. This finding underlines the importance of the skyline as illumination-invariant scene

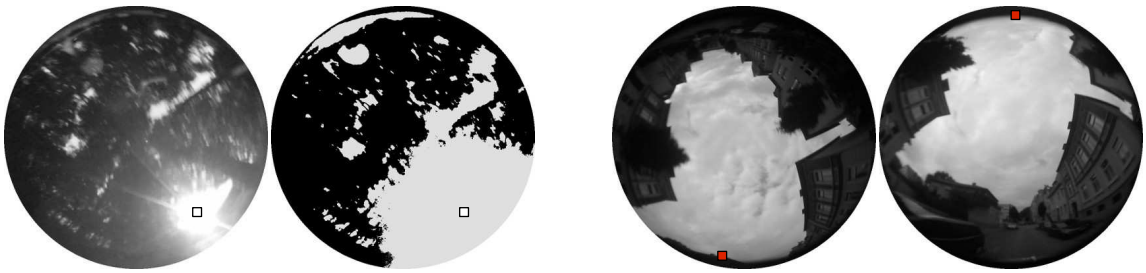
descriptor for outdoor applications.

It can be seen that increasing the number of bands only marginally decreases the standard deviation. The comparably small influence might be due to the mainly large structures (e.g. the university main building) in the panoramic images. For finer structures, as for example received by an agent driving under trees, the influence of the maximal number of bands L and L_{CG} could increase.

Figure 4.7 shows that the relation between the distance measures AS and BS and the metric distance strongly depends on the databases. As can be seen, for same-database tests the databases *uni_late* and *uni_early* provide a (nearly) linear relation between the distance measures AS and BS and the metric distance. In contrast, for the database *uni_winter* we have that for distances of less than 12 m the AS and BS increase, but for more than 12 m decrease. Using Sobel-filtered images (hS), the best relation for the cross-database tests is obtained for the pairing *uni_winter* and *uni_late*. Both histogram equalization with (HS) and without (Hs) subsequently applying the Sobel filter only provided a poor relation between the distance measures AS and BS and the metric distance for all database combinations.

4.5 Discussion

Example 4.5: Failure Cases



Examples of panoramic images (raw fish-eye images as captured by the camera) where two typical kinds of errors occur: Left: Direct incident sunlight onto the camera sensor causes lens flare and oversaturation in the UV channel, leading to misclassification errors for the skyline extraction (white squares). These images are from our study Stone et al. (2016); in ongoing experiments we use a more strict UV filter glass (cut-off at 390 nm) and split the panoramic image into multiple segments (section 2.3.7) to reduce these effects. Right: The skyline extraction is erroneous if the sky portion is not completely visible in the panoramic images, often due to strong tilt of the camera (red squares). By increasing the angle of view of the camera, this effect can be reduced.

In this chapter a new method has been suggested for visual localization using only the skyline as scene descriptor. To extract the skyline, methods derived in chapter 2 are used on images acquired by a UV-sensitive fish-eye camera. Our method does not rely on expensive hardware; only a standard (low resolution) camera, a fish-eye lens, and a UV-only filter glass are required. As of the skyline as rotation- and illumination-invariant scene descriptor it allows the localization of aggressively maneuvering robots. We have seen in section 4.4 that the feature-based method FABMAP could not cope with the challenging conditions (e.g. repetitive environments and motion blur) in our experiments. For less jittery driving, e.g. for autonomous cars, feature-based methods do not have to deal with motion blur and — if appropriate hardware is available — high resolution images can be used. Moreover, in many applications the agent is not exposed to rotations but drives on homogeneous tracks as for example streets. Therefore it is likely that in structured environments (e.g. urban areas) feature-based approaches perform as good or better than our method. However, feature-based methods are computationally expensive and can therefore often

not be used in real-time applications on low-cost hardware. The method seqSLAM (vanilla) is computationally cheap and achieves illumination-invariance by applying local histogram equalization, however due to the missing rotation-invariance, unexpected tilt of the robot remains a problem.

As long as a complete view of the skyline is available, we could show that our proposed method allows reliable localization. Our approach is — due to the rotation-invariance — not limited to planar movements, but could also be used on micro air vehicles. Even though, our method has some obvious disadvantages: First, in environments where no salient skyline feature can be observed, e.g. flat environments as deserts or plains, the skyline might not provide sufficient information to find reliable frame correspondences. Second, in the evening the extraction of the skyline becomes more difficult due to the low UV-light intensities, however in this situation infrared light could be used as an alternative (Meguro et al., 2008). Third, for strongly tilted images, the skyline can be cropped (example 4.5), however this problem can be avoided can be avoided by using fish-eye lenses with a larger angle of view. Another problem — which also affects the other tested localization methods — is lens flare. The results from chapter 2 suggest that for lens flare the quality of the skyline extraction could be increased by using multiple segments.

Our comparison between different spectra from section 4.4.6 shows that the bispectrum performs only slightly better than the amplitude spectrum. For applications with limited computational resources, the amplitude spectrum should therefore be preferred. Moreover, the question arises if the distance measures AS and BS — which calculate the squared differences of the amplitude spectra and bispectra — are optimal choices. Alternatives to our "naive" distance measures could be measures which are additionally weighted (Hammer and Villmann, 2002). A more extensive evaluation of the influence of each entry in the amplitude spectrum or bispectrum could reveal if additional weighting terms could be used to improve the quality of localization. Furthermore, calculating the bispectrum only for neighboring bands (instead of combinations between all bands, section 3.9.3) could reduce the computation time of the bispectrum strongly while still providing additional information.

4.6 Conclusion

In this chapter we proposed a biologically inspired approach for localization from the study of insects. Using a UV-sensitive fish-eye camera, we extracted the skyline from panoramic images and used its amplitude spectrum as illumination- and rotation-invariant scene descriptor. As we could show, our localization method outperformed the original seqSLAM and FABMAP under challenging conditions on three different tracks, where the robot was exposed to strong illumination changes, tilt, and repetitive environments like forests.

CHAPTER 5

Holistic Visual 3D Compass

The majority of approaches for rotationally aligning panoramic images in outdoor environments use feature-based methods which extract and match visual features to determine the rotational offsets analytically. However, these methods can be computational expensive and might suffer from distortion effects, motion blur, calibration errors, or highly repetitive and featureless environments. An alternative approach is to use a holistic visual compass; a method which simulates a wide range of possible camera orientations and searches for the best match. While this approach has been successfully implemented for rotations around a single axis, the increasing computation time for 3D rotations limits the usability for real-time applications. It has been suggested to use spherical harmonics to represent panoramic images, which allows to implement arbitrary rotations using sparse matrix-vector multiplications. In this chapter, we present strategies which are crucial to implement a real-time visual 3D compass using spherical harmonics. We provide a software implementation of the visual 3D compass and analyze the effect of increasing tilt and translation between pairs of panoramic images as well as the effect of illumination changes. The visual 3D compass can be used on low-cost hardware in real-time.

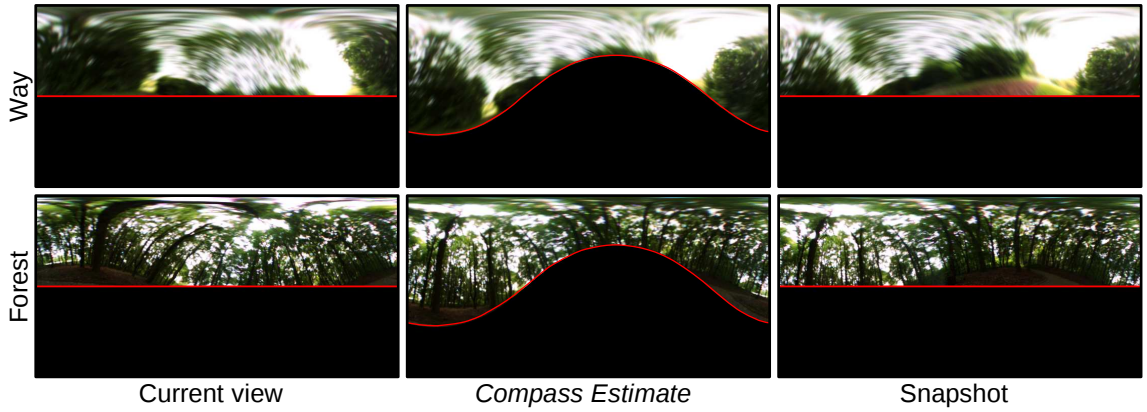
5.1 Introduction

A common problem in machine vision — especially for autonomously driving agents like wheeled robots or micro air vehicles (MAV) — is rotational misalignment between subsequently captured panoramic images. Since many robots are equipped with a panoramic camera, the visual information provided by it can be used to determine the rotational offset instead of installing additional hardware like accelerometers or gyroscopes. One widely used approach is to detect and describe visual features in the scene and match them over subsequent images. This allows to determine the rotation of the camera, e.g. by extracting point features (Davison et al., 2007, Makadia et al., 2007, Engel et al., 2012) or vanishing points (Bazin et al., 2008, Lee and Yoon, 2015). While feature-based methods commonly estimate the complete 5 DoF of the camera pose between two images, they might fail under challenging circumstances, e.g. on blurred images, images captured in structureless environments (example 5.1), or for illumination changes (example 5.2). Moreover, feature-based methods are sensitive to camera calibration errors Scaramuzza and Siegwart (2008). An alternative to feature-based methods are holistic methods which use the complete image instead of single features. Well-known examples include optical flow methods (e.g. Horn-Schunk (Bruhn et al., 2005) or Lucas-Kanade (Tamgade and Bora, 2009)) which determine the camera pose from the visual changes in the image (review: (Sun et al., 2014)), or predictive methods which internally simulate possible camera poses and search for the best match between two images (e.g. warping (Möller et al., 2010)). A more detailed overview on methods used to determine the relative pose between two images can be found in section 1.2.

In this chapter we use the predictive method called visual compass by Zeil et al. (2003) to determine the rotation between two panoramic images. A visual compass simulates a wide range of possible rotations and performs an exhaustive search to find the rotation which minimizes the sum of squared pixel-wise differences between those images. Performing rotations around

Example 5.1: Panoramic Image Alignment: Blur

Two examples of panoramic images with an opening angle of 180° recorded at roughly the same location — here referred to as current view and snapshot — which suffer from rotational misalignment. The rotational misalignment can be estimated and corrected by performing an exhaustive search to find the rotation which minimizes the sum of squared pixel-wise differences between the current view and the snapshot. This approach is feasible for different applications as visual navigation, camera stabilization, and 3D shape alignment. Contrary to feature-based methods, the visual 3D compass does not depend on diverse/distinguishable features.



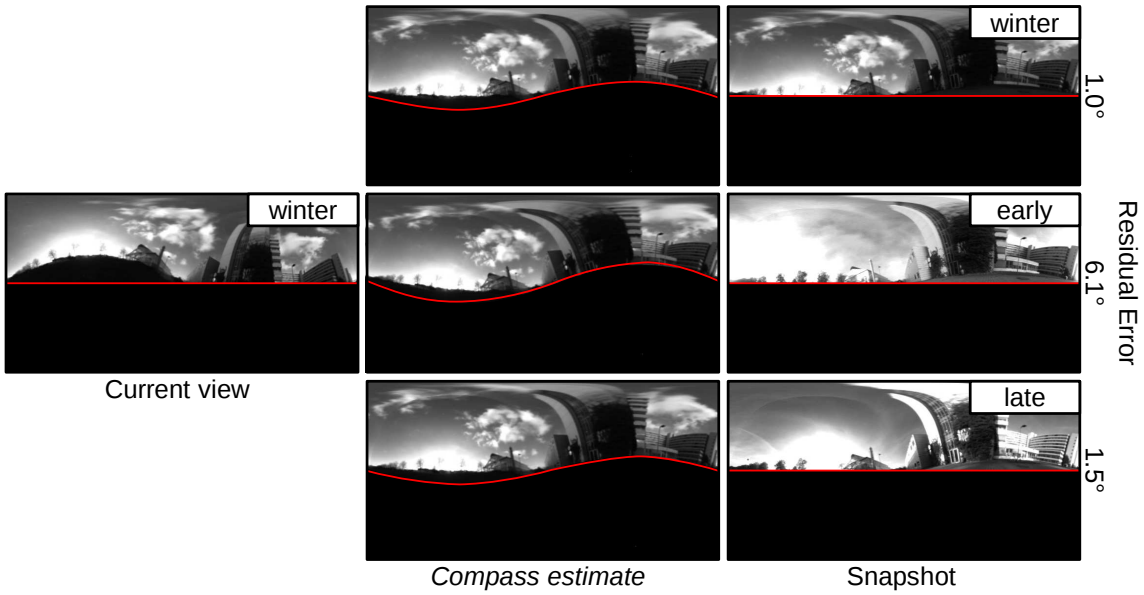
The images shown are taken from a live demo and correctly estimate the rotational offsets under challenging conditions: The panoramic images are strongly blurred due to camera motion and do not contain lines as necessary for determining vanishing points. Common failure cases for the visual 3D compass can be found in example 5.3 and 5.4.

arbitrary axes on panoramic images in the spatial domain suffers from various drawbacks (section 3.2). Therefore it has been suggested to use Fourier analysis on the sphere — i.e. real spherical harmonics (RSH) — in order to search for the best match in frequency domain (Makadia et al., 2004). However, to reduce the computation time most approaches are limited, e.g. to single axis rotations (Friedrich et al., 2007, Stürzl and Mallot, 2006), zonal spherical harmonics[1] (Makadia and Daniilidis, 2003), or a low number of bands (Burel and Henoco, 1995). While these approaches work well for Z-axis rotations, our preliminary experiments show that their performances decrease for increasing X/Y-axis rotations, especially if only hemispherical data (as captured by common fish-eye lenses) are available. Here we present a real-time implementation of the visual 3D compass using RSH which is optimized for hemispherical panoramic images: We suggest to fill in the missing visual information with noise and use weighting functions to decrease its influence on the matching process[2]. We also present methods to reduce the computation time, e.g. by using an efficient rotation parameterization and a coarse-to-fine approach. These improvements allow us to achieve real-time performance on low-cost hardware. While our approach cannot compete with feature-based methods regarding accuracy and reliability in structured environments, it requires significantly less computation time and works in featureless environments as well as with strong motion blur.

[1] Zonal spherical harmonics are a subset of the spherical harmonics y_m^l with $m = 0$ and form the central column in figure 3.3. An important property of zonal spherical harmonics is that they are invariant under rotations around the Z-axis, which can be used to separate the determination of Z-axis rotations from X/Y-axis rotations.

[2] An alternative approach to mask out specific areas by "slicing" the view is given by Dederscheck et al. (2010b). However, their approach is limited to planar movement and requires additional mappings to the image plane.

Example 5.2: Panoramic Image Alignment: Varying Illumination



The visual 3D compass is partially able to cope with illumination changes by preprocessing the images (e.g. edge filtering, section 3.11.2). This example shows the estimates for a cross-database test: The current view (from the database *uni_winter*) is compared to snapshots from all *uni* databases (*uni_winter, uni_early, uni_late*). As can be seen, the residual rotational error depends on the database from which the snapshot is taken.

5.2 Visual 3D Compass

Our visual 3D compass uses spherical harmonics (SH) to represent panoramic images (chapter 3); more precisely we use real spherical harmonics (RSH) to avoid computations within the complex numbers. In this section we present details for the implementation of the visual 3D compass. Our approach is specialized for speed rather than accuracy; for high precision calculations in the basis of SH and RSH, other libraries (e.g. (Moore, 2008, Wieczorek, 2015)) are available.

5.2.1 Exhaustive Search

As described in example 3.4, we represent two panoramic images by $f, g : S^2 \to \mathbb{R}$ and denote their Fourier coefficient vectors by $\vec{f}, \vec{g}$. The visual 3D compass aims to find a rotation which minimizes the difference between two rotationally misaligned panoramic images. This can be achieved by performing an exhaustive search over all possible rotations and search for the rotation which minimizes the integral squared error (ISE, section 3.9.1) between f and g:

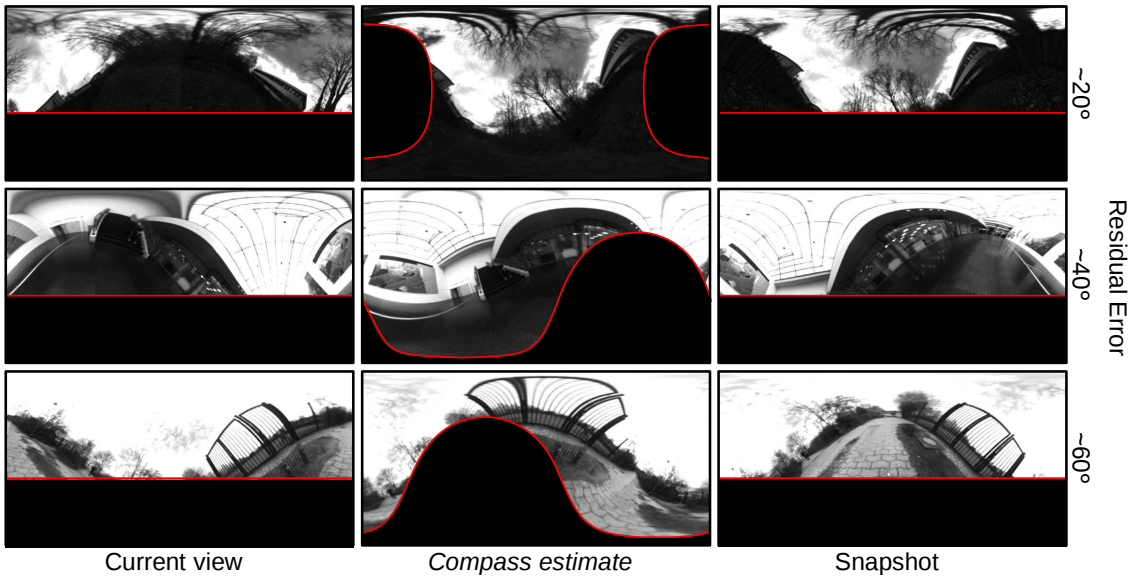
$$\min_{\mathbf{R}\in SO(3)} \mathrm{ISE}(\mathbf{R}\circ f, g) \overset{(3.99)}{=} \min_{\mathbf{R}\in SO(3)} \|\mathbf{R}\circ\vec{f}-\vec{g}\|^2 \tag{5.1}$$

Note that in practical applications the set of rotations $\mathbf{R}$ has to be finite and that the computation time of the visual 3D compass is linearly increasing with the number of tested rotations. Now we expand equation (5.1)

$$\|\mathbf{R}\circ\vec{f}-\vec{g}\|^2 = \left\langle \mathbf{R}\circ\vec{f}-\vec{g}, \mathbf{R}\circ\vec{f}-\vec{g}\right\rangle \tag{5.2}$$

$$= \left\langle \mathbf{R}\circ\vec{f}, \mathbf{R}\circ\vec{f}\right\rangle + \left\langle \vec{g},\vec{g}\right\rangle - 2\left\langle \mathbf{R}\circ\vec{f},\vec{g}\right\rangle \tag{5.3}$$

Example 5.3: Panoramic Image Alignment: Residual Error



Examples of mismatches with increasing residual errors of around 20°, 40°, and 60°. As can be seen, the visual difference between the compass estimates and the snapshots is — at least for the most areas — small, however the panoramic images still do not completely overlap, e.g. the painting (middle row) and fence (bottom row) are clearly misplaced. For humans, these features give an immediate impression of the existing misalignment, however they do not provide much visual information usable for the visual 3D compass.

and apply lemma 3.11 to obtain

$$\overset{\text{le.3.11(i)}}{=} \left\langle \mathbf{R}\mathbf{R}^T \circ \vec{f}, \vec{f} \right\rangle + \left\langle \vec{g}, \vec{g} \right\rangle - 2 \left\langle \mathbf{R} \circ \vec{f}, \vec{g} \right\rangle \tag{5.4}$$

$$= \left\langle \vec{f}, \vec{f} \right\rangle + \left\langle \vec{g}, \vec{g} \right\rangle - 2 \left\langle \mathbf{R} \circ \vec{f}, \vec{g} \right\rangle . \tag{5.5}$$

Since $\vec{f}$ and $\vec{g}$ are constant, this allows us to reformulate the optimization criterion as follows:

$$\min_{\mathbf{R} \in SO(3)} \|\mathbf{R} \circ \vec{f} - \vec{g}\|^2 \;\Leftrightarrow\; \max_{\mathbf{R} \in SO(3)} \left\langle \mathbf{R} \circ \vec{f}, \vec{g} \right\rangle \tag{5.6}$$

In section 5.2.5, this alternative representation will be used to show that the exhaustive search using the ISE is invariant to global illumination changes as described in section 3.11.2.

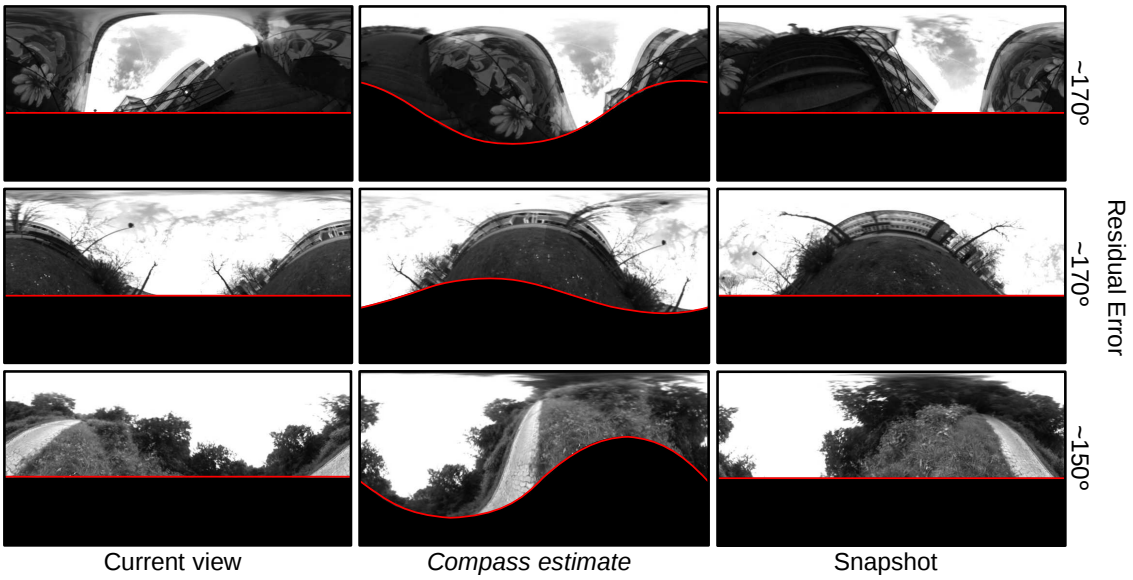
5.2.2 Rotation Parameterization

A crucial step for an efficient implementation of the visual 3D compass is to choose an appropriate rotation parametrization. Recall that a rotation $\mathbf{R}$ in the basis or RSH is represented by the Wigner-D matrix $\mathbf{d}^L(\mathbf{R})$, where L is the maximal number of bands used (section 3.43). Due to its intuitiveness, we choose the XYZ Tait-Bryan parametrization

$$\mathbf{R} = \mathbf{R}_{X,\gamma}\mathbf{R}_{Y,\beta}\mathbf{R}_{Z,\alpha} = \mathbf{R}_{Y,-90°}\mathbf{R}_{Z,\gamma}\mathbf{R}_{Y,90°}\mathbf{R}_{X,90°}\mathbf{R}_{Z,\beta}\mathbf{R}_{X,-90°}\mathbf{R}_{Z,\alpha} \tag{5.7}$$

as often used in aeronautics. Here, we substituted the X/Y-axis rotations by Z-axis rotations and fixed X/Y-axis rotations of $\pm 90°$. As a consequence, only the rotation matrices $\mathbf{d}^L(\mathbf{R}_{Y,90°})$ and $\mathbf{d}^L(\mathbf{R}_{X,90°})$ have to be calculated using recursive formulas (theorem 3.21). Note that the matrices $\mathbf{d}^L(\mathbf{R}_{Y,90°})$ and $\mathbf{d}^L(\mathbf{R}_{X,90°})$ are sparse and only have $\frac{1}{3}(L^3 + 2L)$ and $2L^2 - L$ non-zero

Example 5.4: Panoramic Image Alignment: Total Failure Cases



In some cases, the visual 3D compass estimates rotations with residual rotational errors close to 180°. As can be seen, all compass estimates shown look similar to the snapshots with respect to the overall shape of dark and bright regions. However, they match completely different objects onto each other. This problem often occurs due to symmetries in the scene: For example, buildings or trees on both sides of the robot increase the possibility to wrongly add a 180° Z-axis rotation. Using a limited search space significantly reduces the number of mismatches (section 5.5).

entries, respectively (corollary 3.27). Moreover, the remaining rotations around the Z-axis can be implemented in a way which benefits from SIMD instructions (section 5.2.3).

In the following, we assume that two panoramic images represented by functions f, g use the same number of bands L for the Fourier transform; therefore the corresponding index L is omitted. Using the parametrization from equation (5.7), we can calculate the ISE (section 3.9.1) for a rotation $\mathbf{R}$ as

$$\begin{aligned} ISE(f,g) &\overset{(3.99)}{=} \|\mathbf{R} \circ \vec{f} - \vec{g}\|^2 \\ &\overset{(5.7)}{=} \|(\mathbf{R}Y, -90°\mathbf{R}_{Z,\gamma}\mathbf{R}_{Y,90°}\mathbf{R}_{X,90°}\mathbf{R}_{Z,\beta}\mathbf{R}_{X,-90°}\mathbf{R}_{Z,\alpha}) \circ \vec{f} - \vec{g}\|^2 \\ &= \|(\underbrace{\mathbf{R}_{Z,\gamma}\mathbf{R}_{Y,90°}}_{III}\underbrace{\mathbf{R}_{X,90°}\mathbf{R}_{Z,\beta}}_{II}\underbrace{\mathbf{R}_{X,-90°}\mathbf{R}_{Z,\alpha}}_{I}) \circ \vec{f} - \mathbf{R}_{Y,90°} \circ \vec{g}\|^2. \end{aligned} \tag{5.8}$$

This allows us to compute rotations systematically by calculating I, II, and III for a given set of rotation angles (γ, β, α) (in the following called *search space*). By evaluating the ISE in a nested loop (exhaustive search), we perform a high amount of cheap Z-axis rotations (III), while costly rotations around the X/Y-axes are performed rarely (I, II). Note that we rearranged (5.8) to remove the fixed rotation $\mathbf{R}_{Y,-90°}$ from III; instead we directly apply the inverse $\mathbf{R}_{Y,90°}$ to $\vec{g}$.

5.2.3 Fast Z/Y-Axis Rotations

From corollary 3.27 we know that the Wigner-D matrices for rotations around the Z-axis are sparse matrices with only $4l + 1$ non-zero entries for each band l. Therefore a rotation of the Fourier coefficient vector can be performed efficiently via a sparse matrix-vector multiplication. However, modern CPUs benefit from SIMD parallelization techniques which cannot efficiently be

applied to sparse matrix-vector multiplications. Alternatively, the rotation can be implemented as a sum of two vector-vector point-wise products using its explicit expression from theorem 3.23:

$$r_m^l = \cos(|m|\alpha) f_m^l + \sin(-m\alpha) f_{-m}^l \tag{5.9}$$

Let $\vec{c}^{\,l} = (\cos(|-l|\alpha), \ldots, \cos(|l|\alpha))^T$, $\vec{s}^{\,l} = (\sin(-l\alpha), \ldots, \sin(l\alpha))^T$, and $\underline{\vec{f}^{\,l}}$ be the reversed (regarding the order) of $\vec{f}^{\,l}$, then the rotated Fourier coefficient vector is

$$\vec{r}^{\,l} = \vec{c}^{\,l} \cdot \vec{f}^{\,l} + \vec{s}^{\,l} \cdot \underline{\vec{f}^{\,l}}. \tag{5.10}$$

This calculation can easily be parallelized using SIMD instructions, decreasing the computation time for Z-axis rotations.

From theorem 3.24 and figure 3.8 we have that rotations around the Y-axis are represented for each band by a direct sum of two smaller dense square matrices of dimensions l and $l+1$. By implementing them as dense matrices, SIMD parallelization techniques can be applied increasing the overall performance compared to an implementation using sparse matrices.

5.2.4 Coarse-to-Fine Search

An exhaustive search over a complete search space of arbitrary rotation angles (γ, β, α) is a time-consuming task. As described in Zeil et al. (2003), the ISE between two panoramic images commonly increases with their rotational offset. We take advantage of this by using a search space with a wide range and a coarse resolution at first; and then repeatedly narrow the search space down using a finer resolution (coarse-to-fine, figure 5.1). With each search we decrease the size of the search space while we increase the resolution. While the implementation of this step is trivial, the choice of coarse-to-fine parameters has a crucial impact on the performance of the compass: A too coarse resolution of the search space might miss the global minimum, while a too fine resolution increases the computation time.

5.2.5 Global Illumination Invariance

As described in section 3.11.2, the effect of global illumination changes on the appearance of a scene can be described by a linear function. As stated in equation (3.126), global illumination changes on a function f can be expressed as a linear function $I(f) = \alpha f + \beta$. As shown by Dederscheck et al. (2010a), we can determine the coefficients $\alpha, \beta \in \mathbb{R}$ in the spatial domain. Denote by m_f, m_g and v_f, v_g the empirical mean and variance of f and g in the spatial domain (e.g. on a panoramic image), then we have $\alpha = \sqrt{\frac{v_g}{v_f}}$ and $\beta = m_g - \alpha m_f$. Therefore it is sufficient to determine α, β and apply the illumination change to f.

Here we show how illumination changes can directly be represented in the basis of RSH and that — using the ISE (section 3.9.1) as distance measure — the exhaustive search of the visual 3D compass is not affected by global illumination changes. However, as can be seen in our cross-database tests (section 5.5.2), the method *weighted* — which uses the WISE (equation (3.122)) as distance measure — seems not to be invariant against global illumination changes.

To show that the visual 3D compass is invariant under global illumination changes, we first need that the associated Legendre polynomial for $l = m = 0$ is given by

$$1 \overset{(3.46)}{=} P_0^0(\cos\vartheta) \overset{(3.55)}{=} \frac{1}{K_0^0} y_0^0(\vartheta, \varphi), \tag{5.11}$$

allowing us to calculate the integral of a single RSH as

$$\int_s y_m^l(s)ds \overset{(5.11)}{=} \frac{1}{K_0^0} \int_s y_0^0(s) y_m^l(s) ds = \begin{cases} \frac{1}{K_0^0} & \text{if } l, m = 0 \\ 0 & \text{else} \end{cases}, \tag{5.12}$$

where we use that the RSH form an orthonormal basis. Now we can examine how illumination changes affect the calculation of the Fourier coefficients by Fourier transforming $I(f)$:

$$I(f_m^l) := \int_s y_m^l(s) I(f(s)) ds \overset{(3.126)}{=} \int_s y_m^l(s)(\alpha f(s) + \beta) ds \tag{5.13}$$

$$= \alpha \int_s y_m^l(s) f(s) ds + \beta \int_s y_m^l(s) ds \overset{(5.12)}{=} \begin{cases} \alpha f_m^l + \frac{\beta}{K_0^0} & \text{if } l = m = 0 \\ \alpha f_m^l & \text{else} \end{cases} \tag{5.14}$$

As can be seen, for scaling (multiplicative illumination changes) the Fourier coefficient vector $\vec{f}$ is scaled with α, while shifting (additive illumination changes) adds $\frac{\beta}{K_0^0}$ to the Fourier coefficient entry f_0^0. Taken together, we obtain the Fourier coefficient vector $I(\vec{f}) = \alpha\vec{f} + \frac{\beta}{K_0^0}\vec{e}_0$, where $\vec{e}_0$ is the Fourier coefficient vector $(1, 0, 0, \dots)^T$. The visual 3D compass searches for the rotation $\mathbf{R}$ which minimizes $\left\langle \mathbf{R} \circ \vec{f}, \vec{g} \right\rangle$ (equation (5.6)). If f is affected by illumination changes, this term becomes

$$\left\langle \mathbf{R} \circ I(\vec{f}), \vec{g} \right\rangle \overset{(5.14)}{=} \left\langle \mathbf{R} \circ \left(\alpha\vec{f} + \frac{\beta}{K_0^0}\vec{e}_0 \right), \vec{g} \right\rangle = \alpha \left\langle \mathbf{R} \circ \vec{f}, \vec{g} \right\rangle + \frac{\beta}{K_0^0} \left\langle \vec{e}_0, \vec{g} \right\rangle \tag{5.15}$$

$$= \alpha \left\langle \mathbf{R} \circ \vec{f}, \vec{g} \right\rangle + \frac{\beta g_0^0}{K_0^0}, \tag{5.16}$$

it can be seen that global illumination changes only scale the scalar product and add some fixed offset — both independent of the rotation itself. Therefore, we finally obtain the equivalence of both optimization problems

$$\max_{\mathbf{R}} \left\langle \mathbf{R} \circ \vec{f}, \vec{g} \right\rangle \quad \Leftrightarrow \quad \max_{\mathbf{R}} \left\langle \mathbf{R} \circ I(\vec{f}), \vec{g} \right\rangle \tag{5.17}$$

for all $\alpha > 0$.

5.2.6 Linearization of the Compass Search

The tangent distance introduced in section 3.11.3 can be used to linearize arbitrary transformations. In this section we describe how the tangent distance can be used to linearize rotations and translations (section 3.8) to increase the robustness of the visual 3D compass. Note that the two-sided tangent distance has to solve the underlying optimization problem from equation (3.133) for each search step. In contrast, by linearizing transformations of the snapshot only, the necessary calculations of the one-sided tangent distance (equation (3.134)) can be precalculated for the complete compass search. We use the one-sided tangent distance during the compass search to address two problems:

First, the tangent distance can be used to reduce the error induced by using coarse search steps. Especially the first search phase commonly uses large rotation angles. As shown in figure 5.1, the first three search phases of the search space used for the experiments performed in this chapter have search steps of 32°, 16°, and 8°. We use the tangent distance to linearize small rotations of the snapshot by 8°, 4°, and 2° for the first three search phases. The necessary tangent planes are approximated using equation (3.135). If we use the tangent distance during the compass search, we only use it during the first three search phases. For the fourth and ongoing search phases, the rotation angles are comparable small ($\leq 4°$) and we expect no noticeable gain by using the tangent distance.

Second, for the visual 3D compass we generally assume that panoramic images were recorded at the same location but with a rotational offset. The exhaustive search as described in section 5.2.1 does not take translations of the camera into account. As a consequence, translations can be misinterpret as rotations. With an increasing distance between the locations at which the images were recorded, this error increases and falsifies the rotation estimation. We use the

tangent distance to linearize small translations of the snapshot, the required tangent planes can be calculated using equation (3.136) with $\vec{t}_X = (0.02, 0, 0)^T$ (analogously for the Y/Z-axes). In contrast to rotations, we linearize translations during *all* search phases.

5.2.7 Search Spaces

Throughout the experiments conducted in this chapter we use two different search spaces: First, we try to find suitable parameters for the visual 3D compass to estimate arbitrary rotations around the Z-axis with strong tilt. This experiment should simulate the search for the rotational offset between two panoramic images without prior knowledge, e.g. a wheeled robot or micro air vehicle determining its orientation relative to a previously known panoramic image. Second, we try to estimate comparably small rotations using prior knowledge. For example, a wheeled robot performing route following subsequently corrects its orientation. Assuming a high frame rate — and therefore a high number of visual 3D compass estimates — the rotational offset between subsequent images is small.

Using the coarse-to-fine approach described in section 5.2.4, use a search space with a wide range and a coarse resolution at first; and then repeatedly narrow the search space down using a finer resolution. With each search we decrease the size of the search space while we increase the resolution. While the implementation of this step is trivial, the choice of coarse-to-fine parameters has a crucial impact on the performance of the compass: A too coarse resolution of the search space might miss the global minimum, while a too fine resolution increases the computation time.

Our two search spaces (*large* and *small*) differ by the rotation parameters chosen for the visual 3D compass An example code showing the search space parameters used for both search spaces can be found in appendix C.2. As can be seen, the two search spaces *large* and *small* only differ for the coarsest search phase (by including or commenting out one specific line in the source code).

5.2.8 Parameter Sets

The visual 3D compass can be customized using various parameter settings. Throughout our experiments (section 5.3) we systematically test various combinations of parameters to evaluate the performance of the visual 3D compass. An overview of the tested parameter sets can be found in table 5.1. Note that we keep some parameters fixed: The number of sampling points for the Fourier transform is set to 10^4, the sampling points are distributed equally on the sphere, the maximal number of bands is $L = 20$, and the maximal number of Clebsch-Gordan coefficients is $L_{CG} = 10$. These parameters strongly rely on the application area and might need to be fine tuned in particular.

In the following we use abbreviations to uniquely identify the parameter sets used. For example, the method *weighted* used on panoramic images with an opening angle of 220°, constant noise, tangent distance enabled, and no further preprocessing is abbreviated by *W220CThs*. A wildcard indicates that the parameter may vary, for example *W220C*hs* means that tangent distance might be enabled or disabled. In this case

5.3 Experiments

In the following we examine the performance of the visual 3D compass to align two rotationally misaligned panoramic images (current view and snapshot). As distance measure between the rotationally misaligned current view and snapshot we use the rotational difference defined in section 3.3.3. We use two different characteristics to describe the rotational difference between the current view and snapshot before and after correction. The first characteristic is the **failure rate**: After applying the visual 3D compass, the rotational difference is either increased (failure) or decreased. Failures mainly appear due to symmetries in the images, which sometimes lead to mismatches as shown in example 5.4. To avoid high failure rates for small rotations, we only call a rotation estimate a failure if it increases the rotational difference by 3° or more. The second characteristic is the **rotational difference after correction**: If the visual 3D compass decreased the rotational difference, this represents the residual rotational difference. In the following errors

Parameter	Values	Abb.	Description
Method	Fill	F	Lower hemisphere is filled with noise.
	Hemi	H	Lower hemisphere is filled with information from the upper hemisphere using symmetries.
	Weighted	W	Lower hemisphere is filled with noise. Additionally, weighting functions are used to reduce the influence of the noise.
	Complete	C	The full-spherical panoramic image is used.
Open. Angle	180°	180	Opening angle of common fish-eye cameras.
	220°	220	Recent fish-eye cameras can achieve opening angles of 220°.
	360°	360	A full-spherical camera setup is used.
Noise	Natural	N	Noise is based on an amplitude spectrum extracted from the *mixed_** databases.
	Constant	C	White noise.
Tang. Dist.	Off	t	Tangent distance is not used.
	On	T	Tang. dist. is used to linearize rotations and translations during the compass search.
Preproc.	hs	hs	No preprocessing is applied.
	hS	hS	The images are edge filtered (Sobel filter).
	Hs	Hs	The images are histogram equalized.
	HS	HS	The images is edge filtered and afterwards histogram equalized.
	HDR	HDR	HDR images are used.

Table 5.1: Overview of the different tested parameters; each set of parameters is discussed in detail in the following sections: Methods (section 3.10.3); Noise (section 3.11.1); Tangent distance (section 3.11.3); Preprocessing (section 3.11.2); Opening angle (section 1.3). The abbreviations are used to uniquely identify the methods together with their used parameter set. For example, the method *weighted* used on panoramic images with an opening angle of 220°, constant noise, tangent distance enabled, and no further preprocessing is abbreviated by *W220CThs*.

will be written as tuples $(x\%, y°)$, where $x\%$ is the failure rate and $y°$ the residual error. Note that it is difficult to define an order on the error tuple since the priority to reduce the failure rate or the residual error depends on the use-case. As our results show, both the failure rate and the residual error are commonly related (both are low or high).

In the following we describe three different experiments whose results are presented in section 5.5. Each of the three experiments is performed twice (once for the large and the small search space) and uses the same central assumptions: The panoramic images are taken from our full-spherical panoramic image database; for detailed information see appendix D. From the database images we extract panoramic images with an opening angle of 180° (hemispherical) or 220° (wide-angle). The database images have an opening angle of 310° (the remaining 50° show the camera rig) which allows us to create artificially tilted panoramic images with a maximal tilt of $\frac{310°-220°}{2} = 45°$. The rotational misalignment of each image pair (current view and snapshot) is randomly chosen depending on the tested search space: For the *large* search space, each image is rotated arbitrarily around the Z-axis and tilted (X/Y-axes rotations) by up to 45°. This allows a maximal tilt angle of up to 90° between two panoramic images. For the *small* search space, the Z-axis rotation is set to 0° and both images are tilted (X/Y-axes rotations) by up to 45°. In contrast to the *large* search space, we restrict the maximal tilt angle between two images to 60°. We chose these rotations for better comparability with the feature-based methods which were tested on panoramic images tilted by up to 60° without any Z-axis rotation applied. For all three experiments, we test the implementation of the visual 3D compass using various parameter settings as described in section 5.2.8.

During our first experiment we apply the visual 3D compass to pairs of rotationally misaligned panoramic images to estimate the rotational difference. The panoramic images are taken from the *mixed_** databases which contain images of 20 indoor and 55 outdoor locations (30 collected in

```
1  Shx shx;
2  // initialize coarse-to-fine approach;
3  // for better readability angles are passed in degrees
4  shx.init_rotations_sphere(30.0°, 120.0°); // only for 'large' exp.
5  shx.init_rotations_cone(  32.0°,  64.0°); // only for 'small' exp.
6  shx.init_rotations_cone(  16.0°,  32.0°);
7  shx.init_rotations_cone(   8.0°,  16.0°);
8  shx.init_rotations_cone(   4.0°,   8.0°);
9  shx.init_rotations_cone(   2.0°,   4.0°);
10 shx.init_rotations_cone(   1.0°,   2.0°);
11 shx.init_rotations_cone(   0.5°,   1.0°);
12 // set the maximal number of bands
13 shx.init_bands(20);
14 // set the number of sample points used by the Fourier transform
15 shx.init_surface(1e4);
16 // precalculate and initialize the visual 3d compass
17 shx.init();
18
19 // load two hemispherical panoramic images which need to be aligned
20 Shpm cv = shx.load_shpm("cv.png", HEMI_RM);
21 Shpm ss = shx.load_shpm("ss.png", HEMI_RM);
22 // estimate the X-Y-Z rotation necessary to align both images
23 Xyz xyz = shx.compass(cv, ss);
```

Figure 5.1: Implementation of the configuration *hemi* using our library libSHC to rotationally align two panoramic images. The calls to *init_rotations_*()* initialize the coarse-to-fine search, for better readability the angles are passed in degrees instead of radians. The first argument is the search-space resolution (e.g. 0.5° steps), the second the range (e.g. $1° = [-0.5°, 0.5°]$) for all three rotation axes. To ensure that rotations around the Z-axis are not limited during the first search-phase, the function *init_rotations_sphere()* sets the range for the Z-axis rotations to $[0°, 360°)$. The two experiments (*large* and *small*) tested in section 5.3 only differ by including or commenting out line 4. Note that for better readability, only a simple example of the visual 3D compass (e.g. no tangent distance) is shown.

winter, 25 in summer). The current view and snapshot are created from the same-database image such that neither environmental changes (e.g. lighting changes or moving objects) nor translational effects from camera movement occur. The results are presented in section 5.5.1.

In the second experiment we use cross-databases (*lab_** and *uni_**) — databases recorded multiple times under varying lighting conditions — to simulate situations in which the current view and the snapshot were captured under differing lighting conditions. For example, a wheeled robot driving in an indoor environment might perform a training run at midday (illumination via direct sunlight) and a test run at the evening (illumination via ceiling lights). As can be seen in example 5.5, the visual appearance of the scenes changes strongly such that a direct comparison of the LDR images becomes more difficult. Therefore we test different preprocessing techniques (section 5.2.8) to increase the robustness of the visual 3D compass. We test the visual 3D compass on all mutually exclusive combinations of the cross-databases; the results are presented in section 5.5.2.

In the third experiment, we test the influence of increasing translational offsets between the locations at which the current view and snapshot were recorded. For this purpose, we use databases with known positional ground truth, i.e. the databases *lab_**, *uni_**, *stairs*, *crossroads*, and *finnbahn*. In these tests, the images are aligned such that the visual 3D compass can only increase the rotational offset between the current view and snapshot. By determining the rotational misalignment as before, this allows us to examine the influence of translational offsets on the visual 3D compass. The results are presented in section 5.5.3.

5.4 Vanishing Points, Optical Flow, and Feature-Based Methods

In this section, three different approaches are discussed which can be used to determine the rotation between two images. Mainly three candidates seem to be suitable for comparison with the visual 3D compass, namely vanishing points, optical flow, and feature-based methods; a brief introduction can be found in section 1.2. In this section we briefly discuss the advantages and disadvantages of each.

It could be shown that vanishing points can reliably be used in urban environments as a computational cheap and highly exact method for rotation estimation between subsequently captured images (Bazin et al., 2008, Lee and Yoon, 2015). However, methods based on vanishing points assume that the environment exposes — mostly artificial — structures such as street markers or edges of houses to estimate vanishing points. Due to the high amount of structures in the ground region, the camera is often oriented in earthward direction such that visual information above the horizon is commonly lost. If the image contains no lines, the extraction of vanishing points fails. Since we do not only use images captured in suburban, but mainly in structureless environments (e.g. between scrubs, trees, and bushes), this approach is not suitable for comparison with the visual 3D compass.

In contrast to vanishing points, optical flow methods do not extract structures from the images, but estimate the camera pose from the pixel-wise changes between two images directly. Several methods have been suggested to estimate the optical flow, e.g. Horn-Schunk (Bruhn et al., 2005) or Lucas-Kanade (Tamgade and Bora, 2009). These methods are commonly based on the assumption that the changes between two subsequently recorded images are small, therefore a high frame rate (e.g. in extreme cases with up to 800 Hz as used by Adarve and Mahony (2016)) is crucial to achieve high accuracies. As a consequence, the estimation of large rotations at once is difficult. A common approach for large rotations is to resize the images repeatedly: Starting with a low resolution of the input image, a coarse optical flow estimate is calculated and refined iteratively by increasing the resolution. This approach has a major drawback: Parts of the panoramic images are cropped off or filled with invalid image information (example 4.1). Since it is not known which parts of the images are affected, these parts cannot be masked out and are always used during the calculation of the optical flow. With an increasing tilt angle this leads to strong errors during the coarse estimation steps of the optical flow.

An alternative approach is to extract visual features of the panoramic images using feature detectors and descriptors such as SIFT (Lowe, 1999) or ORB (Rublee et al., 2011) and match them to find point correspondences between both images. Afterwards, a pose estimation algorithm (e.g. the five-point algorithm by Stewenius et al. (2006)) is used to determine the 5 DoF pose change of the camera between both images. Since the five-point algorithm assumes that five correctly matched points are used as an input, a technique called random sample consensus (RANSAC, Fischler and Bolles (1981)) is used to find the best pose estimation among multiple sets of randomly chosen five-point correspondences. SIFT and ORB use — as most feature detectors and descriptors — perspective camera images and are invariant to scale and rotation of the images. Both are (up to some degree) robust against illumination changes and affine transformations (Wu et al., 2013, Karami et al., 2015). However, fish-eye lenses introduce strong radial distortions in the images which increase with the viewing angle. As a consequence, rotations of the camera strongly change the local appearance of each pixel, decreasing the performance of SIFT and ORB. Variations of SIFT have been formulated to adjust for the radial distortion, e.g. pSIFT (Hansen et al., 2009) and sRD-SIFT (Lourenço et al., 2012).

In the following we use the feature detectors and descriptors SIFT and ORB from the OpenCV library[3] and an openly available implementation of sRD-SIFT[4] for comparison with the visual 3D

[3] The library OpenCV, version 2.4.9, was downloaded from `http://opencv.org/`.

[4] The sRD-SIFT implementation, version 1.0, was downloaded from `http://arthronav.isr.uc.pt/~mlourenco/srdsift/`.

compass. The standard values are used for all three feature detectors and descriptors. Only an additional distortion factor ξ, which describes the radial distortion in the image, is needed by sRD-SIFT. Normally, this parameter is estimated using the EasyCamCalib (Barreto et al., 2009), however the toolbox does not work with fish-eye lenses with opening angles of 180° or higher. In a personal correspondence with the authors of sRD-SIFT, it was suggested to estimate an appropriate value for ξ. The performance of sRD-SIFT for varying parameters ξ can be seen in figure 5.4. For pose estimation, we use the implementation of the Stewenius five-point algorithm and RANSAC from the OpenGV library[5] (Kneip and Furgale, 2014). While the five-point algorithm does not require any parameters, we have to choose a maximal number of iterations and a threshold value for RANSAC. We use the default value of 10^3 maximal RANSAC iterations and a treshold value of 10^{-4}. Varying these parameters (decreasing the threshold value or increasing the number of iterations) can improve the pose estimation, but requires significantly more computation time.

We experimented with extracting fish-eye images from our panoramic image databases, however these resampled images suffered from strong blur and aliasing effects. Since these effects degraded the performance of all tested feature-based methods immensely, we tested the feature-based methods on a separate test set of raw fish-eye images. The raw fish-eye images were collected using a camera (*UI-3370CP-C-HQ* by *IDS Imaging*) equipped with a fish-eye lens (*BF16M220DC* by *Lensation*) with an opening angle of 220°. The camera was mounted on a tripod with adjustable orientation. We collected images at 10 different locations with a resolution of 1310×1310 pixel. At each location, we collected images with tilt angles of $-30°$ to $30°$ in $10°$ steps, allowing us to perform rotational realignment on image pairs with a rotational offset of up to 60° (example 5.6). Note that the rotation angles were set by hand such that a small error of the ground truth of around 1 degree is likely. The results are presented in section 5.5.4.

5.5 Results

In this section we present the results of the three experiments described in section 5.3. Moreover we present the performance of feature-based methods as well as a comparison with feature-based methods.

5.5.1 Single-Database Tests

In our first experiment, we tested the visual 3D compass using all parameter settings described in section 5.2.8 on the panoramic images taken from *mixed_** databases. The complete results are presented in detail in the appendix (table C.1), here we only discuss the most noteworthy observations and best working parameter sets. Generally, the following observations can be made:

1. The methods *fill* and *weighted* achieve noticeably better results on 220° images compared to 180° images. Both errors are approximately halved by using the wider opening angle.
2. The method *weighted* outperforms the other methods *fill* and *hemi* in all tested cases.
3. Misinformation induced by filling the lower hemisphere with noise is less influential for constant noise than for natural noise.
4. The tangent distance decreases the errors for the methods *fill* and *weighted*, but increases the error for the method *hemi* drastically. It is likely that the linearization of the translation and rotation is especially erroneous for regions around $\vartheta = \pi/2$ where the lower and upper hemisphere adjoin on each other.
5. The influence of preprocessing depends on the method used. The method *fill* performs best with Sobel-filtered images, while the method *weighted* achieves best result using either LDR images or histogram equalized images.

The results for all methods using the best parameter settings found are shown in figure 5.2. The overall best results were obtained by the following methods: For the *large* search space and an

[5] The OpenGV library was downloaded from `http://laurentkneip.github.io/opengv/` on 13-September-2016. A version number could not be found.

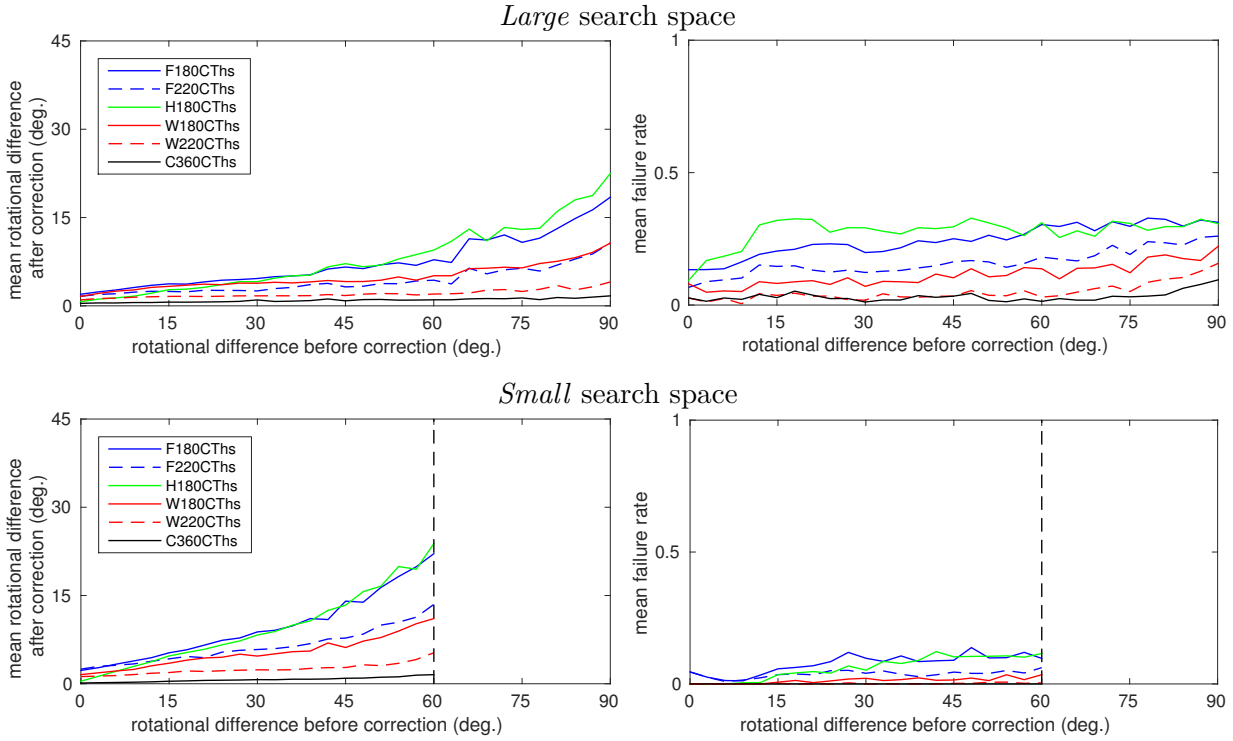


Figure 5.2: The mean rotational difference after correction and the mean failure rate are shown for the *large* and *small* search space on the *mixed_** databases. The best results were obtained with constant noise, tangent distance enabled, and histogram equalization. The best performing method for panoramic images with an opening angle of 220° using weighting functions (*W220CTHs*) achieves $(3.7\%, 3.1°)$ and $(0.1\%, 2.5°)$ for the large and small search space, respectively. For comparison, the method *C360CTHs* which requires full-spherical panoramic images achieves for the *large* and *small* search space $(0.9\%, 0.9°)$ and $(0.0\%, 0.7°)$, respectively.

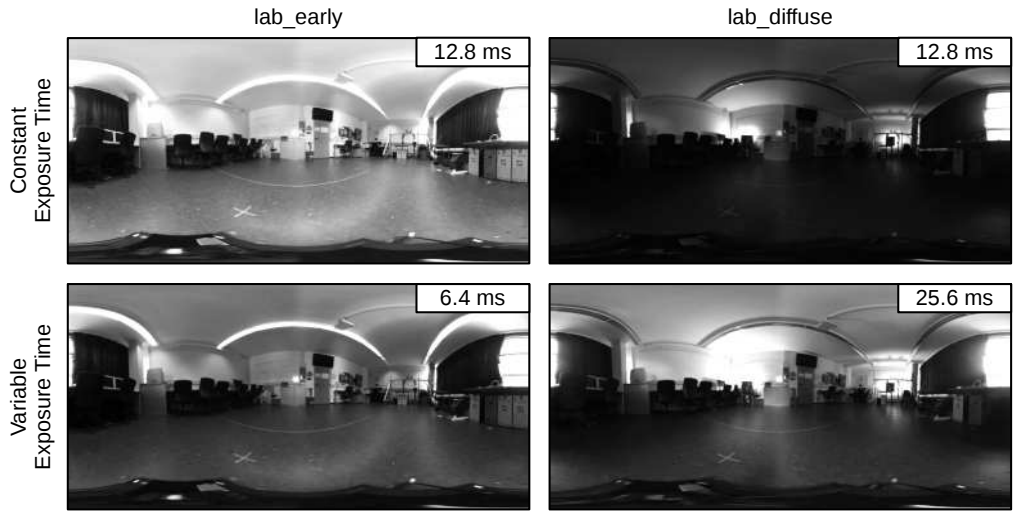
opening angle of 220° best results are obtained using the method *W220CTHs* with constant noise, tangent distance enabled, histogram equalization $(3.7\%, 3.1°)$. This is close to the performance achieved using the method *C360CTHs* $(0.9\%, 0.9°)$ which requires full-spherical panoramic images. Without using weighting functions, best performance is achieved using the method *F220CThS* on Sobel filtered images, however the performance is comparably worse with an error of $(15.5\%, 3.0°)$. For an opening angle of 180°, best results are again obtained using weighting functions on Sobel filtered images (method *W180CThs*) with an error of $(11.4\%, 4.9°)$. The methods *F180CThS* and *H180CThs*, which fill in noise and use hemispherical continuation, respectively, perform noticeably worse with errors of $(22.3\%, 4.9°)$ and $(25.5\%, 8.0°)$. For the *small* search space, both the mean failure rate and the mean rotational difference after correction are reduced: The best result is achieved by the method *W220CTHs* with constant noise, tangent distance enabled, and histogram equalization $(0.1\%, 2.5°)$. Filling in noise (method *F220CThS*) and using hemispherical continuation (method *H180CThs*) achieve at best $(1.6\%, 2.3°)$ and $(4.2\%, 6.2°)$.

5.5.2 Cross-Database Tests

To evaluate the performance of the visual 3D compass under varying lighting conditions, we perform cross-database tests by comparing current views and snapshots drawn from different databases. Since we used multiple exposure times to capture each panoramic image in our panoramic image databases, there are various options to choose exposure times for the current views and snapshots. We chose to use a constant exposure time for all images of a database, e.g. the exposure time of all *lab_** databases used for cross-database tests is 12.8 ms. The exposure times used for each database can be found in the appendix (table D.1). In most robot applications, the cameras exposure time can be controlled to keep the mean brightness level at

a constant level. Therefore, as an alternative approach we also compare images with the same mean brightness level. As depicted in example 5.5, the visual difference between two images with constant exposure times appears stronger. Moreover, table 5.2 shows that using controlled exposure times — the mean brightness value is kept at 65 assuming that the images have 8 bit pixel values — improves the performance for LDR images without preprocessing. The most prominent performance increase is obtained for the method weighted from $(67.8\%, 11.5°)$ to $(37.8\%, 9.9°)$, for all other methods the performance increases only slightly. However, after applying preprocessing techniques (here: Histogram equalization) — which aim to decrease the influence of lighting conditions — the influences of the two exposure modes is marginal. In the following we use the more challenging case of constant exposure times for the tests performed in this chapter.

Example 5.5: Variable Camera Exposure Times



To evaluate the performance of the visual 3D compass under varying lighting conditions, we perform cross-database tests by comparing current views and snapshots drawn from different databases (here: *lab_early*, left column, and *lab_diffuse*, right column). Throughout our tests we compared images which have been captured using a constant exposure time (top row). Alternatively, assuming that the exposure time can be controlled adequately the mean brightness levels of both camera images can be kept at a constant level. (here: 65, assuming 8 bit pixel values, bottom row). The impact on the performance of our tested methods can be seen in table 5.2.

The experiments were carried out on the indoor cross-database *lab_** and the outdoor cross-database *uni_** for each database combination (e.g. *lab_early* versus *lab_dark*). The results are presented in detail in the appendix: Table C.2 and C.3 show the mean performance over all mutually exclusive combinations for the databases *lab_** and *uni_**, respectively. Detailed results for single database combinations for the method *weighted* with an opening angle of 220°, constant noise, and tangent distance enabled are presented in table C.4 and C.5.

As can be seen, the overall performance of all methods is significantly decreased for cross-database tests in comparison to same-database tests. Especially without additional preprocessing, the strong visual differences between the LDR images increases the failure rate. For both the *small* and *large* search space, the best performance on the *lab_** cross-database is obtained by the method *W220CTHs* using weighting functions on images with an opening angle of 220°, constant noise, tangent distance enabled, and using histogram equalization with $(4.1\%, 5.0°)$ and $(2.0\%, 4.5°)$, respectively. For both search spaces, the error is generally increased strongly if one of the alternative preprocessing techniques *hS*, *HS*, or *HDR* is used. Only the method *W220CTHs* on

Exp. Time	Pre-Processing	Fill 180°	Fill 220°	Hemi	Weighted 180°	Weighted 220°	Complete
Constant	hs (none)	40.8% 9.9°	21.7% 5.6°	57.1% 15.0°	75.3% 13.5°	67.8% 11.5°	10.2% 3.3°
Variable	hs (none)	37.7% 9.5°	20.6% 5.4°	53.6% 13.5°	51.3% 12.4°	37.8% 9.9°	8.5% 3.2°
Constant	Hs (Hist. eq.)	32.5% 9.3°	15.2% 5.7°	33.2% 12.1°	14.9% 8.3°	4.1% 5.0°	0.8% 1.7°
Variable	Hs (Hist. eq.)	31.9% 9.2°	14.2% 5.5°	33.2% 12.0°	13.6% 8.1°	3.6% 4.8°	0.8% 1.6°

Table 5.2: The mean performance over all mutually exclusive combinations on the cross-database *lab_** for constant and variable (controlled) exposure times are shown. For comparison, we carried out the experiment once without preprocessing (*hs*) and once with histogram equalization (*Hs*). For the experiments presented in this table we use the *large* search space, constant noise, and the tangent distance is enabled.

the *small* search space has a competitive performance $(2.8\%, 4.5°)$. This is especially interesting since on the *uni_** cross-database only the preprocessing method *HS* shows acceptable results at all: For the *large* search space, the best result is again achieved by the method *W220CTHS* with an error of $(20.6\%, 7.4°)$. However, due to the high failure rate, the visual 3D compass seems not suitable to reliably align the current view and snapshot. For the *small* search space, the same method and settings reduce the failure rate noticeably to $(5.6\%, 6.8°)$.

The results indicate that the visual 3D compass — even though preprocessing techniques are applied — can only compensate for a limited amount of lighting changes. However, by limiting the visual 3D compass (*small* search space), the error is reduced. By examining the performance of the best methods *W220CTHs* and *W220CTHS* on single cross-database combinations (appendix, tables C.4 and C.5), it can be seen that the performance of the Visual 3D compass depends on the used databases: For example, using the *small* search space and the preprocessing technique *HS*, the error ranges between $(0.7\%, 3.4°)$ to $(6.0\%, 6.5°)$ on the *lab_** cross-database and between $(1.1\%, 4.0°)$ to $(8.5\%, 8.4°)$ on the *uni_** cross-database.

5.5.3 Influence of Camera Translation

The visual 3D compass internally simulates rotations of the current view to determine the rotational offset to the snapshot. However, effects of translations (changes of the camera location) between the current view and snapshot are not simulated and can be misinterpreted as effects of rotations. The effect of translations depend on the distance between the current view and snapshot as well as the distance between the camera and surrounding objects: On the one hand, if all objects are close — as for example in indoor environments — already small distances of the camera locations can lead to significant changes in the images. On the other hand, if all objects are far away — as for example in outdoor environments — the effects of translations are significantly smaller. As a consequence, it is not possible to examine the average effect of translations as a function of the translational distance over different databases[6]. To test the influence of translations on the visual 3D compass, we first randomly choose a current view and a snapshots at different locations from database with known ground truth, i.e. *crossroads*, *stairs*, *finnbahn*, *lab_**, or *uni_**. Note that the panoramic images are rotationally aligned. Afterwards, we apply the visual 3D compass to estimate the rotational offset between both images, which, in the best case, should remain zero. Since the visual 3D compass can only worsen the rotational alignment between both images, the failure rate is not of interest. Therefore we only examine the mean ro-

[6] The effect of translations depends not only on the distance traveled by the robot, but also on the distance to the surrounding landmarks. Since the distance to the landmarks is strongly inhomogeneous and depends on the databases, a comparison is not possible.

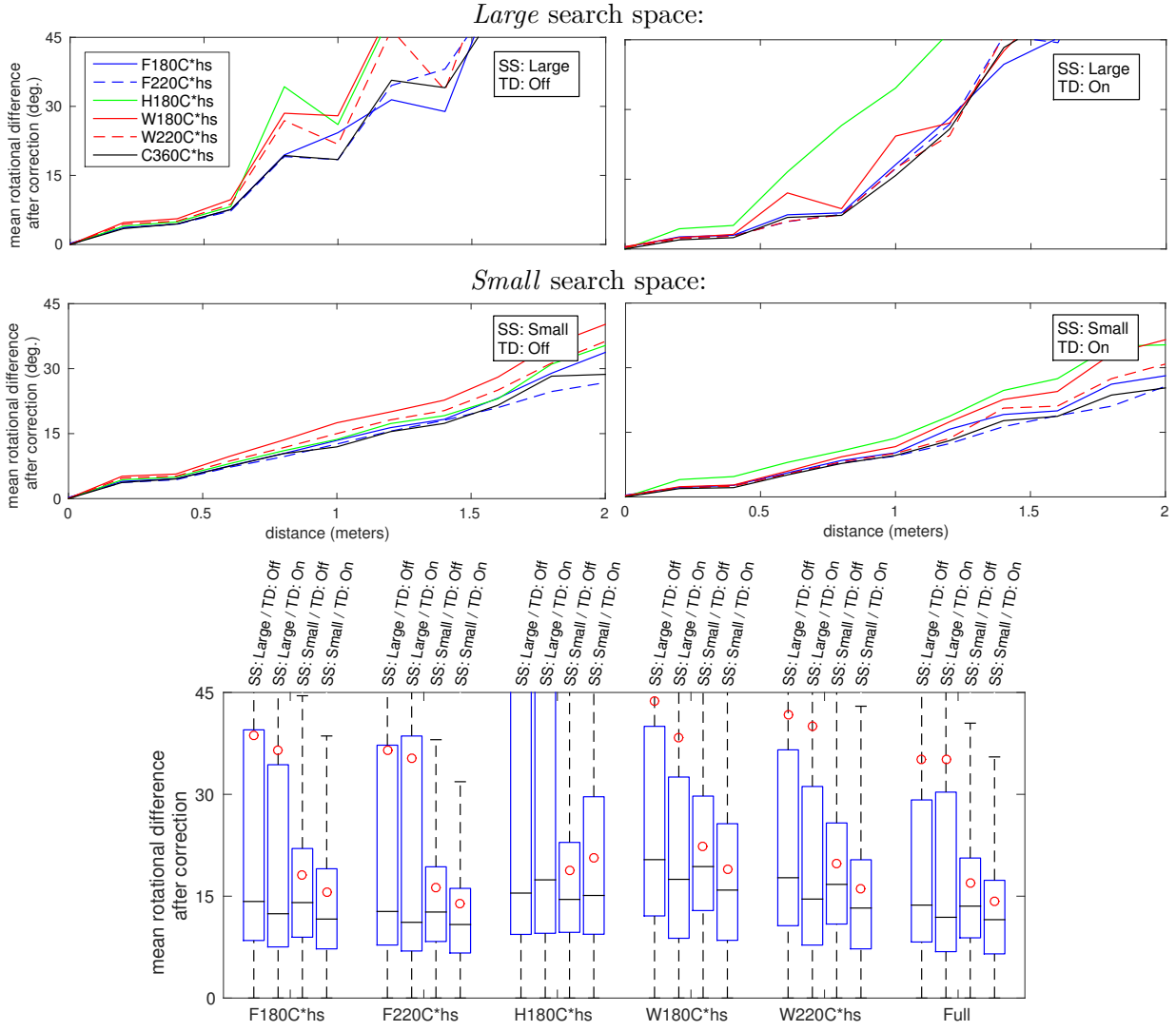


Figure 5.3: The figures show the influence of camera translation on the database ***lab_diffuse***. The mean rotational error after correction is plotted as a function of the translation distance (histogram with 0.5 m bins) using the *large* or *small* (rows) search space and with tangent distance enabled or disabled (columns). For better comparison, these data are also presented as a box plot which shows the mean (red dots), median (black bars), the 25th and 75th percentiles (blue boxes), and a coverage of $3\sigma \hat{=} 97.7\%$ (black dashed lines). For better readability, outliers are not shown.

tational difference after correction for all image pairs. The results are presented for all databases in the appendix (figures C.1-C.5).

Even though the exact results differ, the database *lab_diffuse* shows multiple effects which can be observed on most tested databases and which is therefore analyzed in the following in more detail (figure 5.3): As can be seen, the mean rotational difference after correction increases with the distance between the current view and snapshot. The visual 3D compass performs better on the *small* than the *large* search space. In general, method which fill in the lower hemisphere with noise perform best with an error close to the using full-spherical panoramic images. Using weighting functions or hemispherical continuation worsens the performance on the *large* search space, however on the *small* search space the difference is comparably small. As the box plot shows, the mean error is noticeably worse than the median error, indicating that many outlier exist. Except for hemispherical continuation, using the tangent distance reduces the mean and median errors. Considering all tested grid database shown in the appendix (figure C.5), method which use weighting functions or hemispherical continuation show the worst performance.

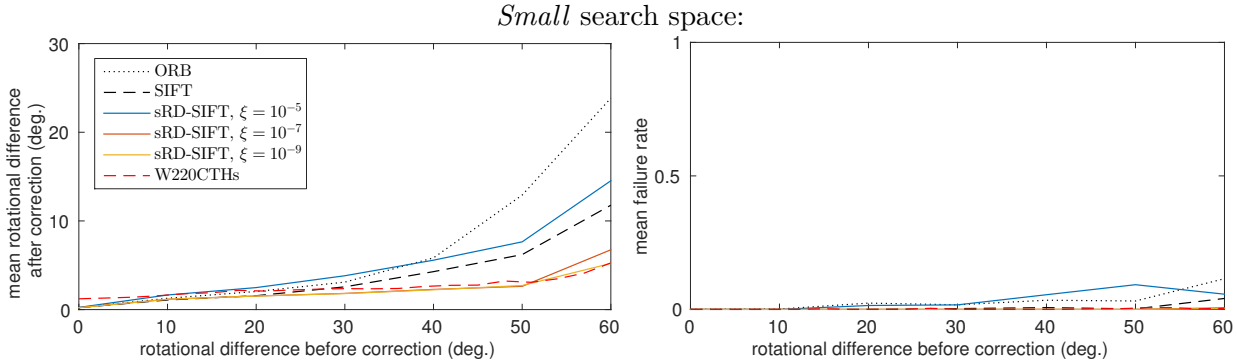


Figure 5.4: The figure shows the mean rotational difference after correction using SIFT, ORB, and sRD-SIFT (with varying parameters ξ). As input we use raw camera images collected at 10 different locations with tilt angles of $-30°$ to $30°$ in $10°$ steps, allowing us to perform rotational realignment on image pairs with a rotational offset of up to $60°$. For comparison, the best method *W220CTHs* (tested on the larger *mixed_** dataset, compare section 5.5.1) of the visual 3D compass is shown.

5.5.4 Feature-Based Methods on Raw Images

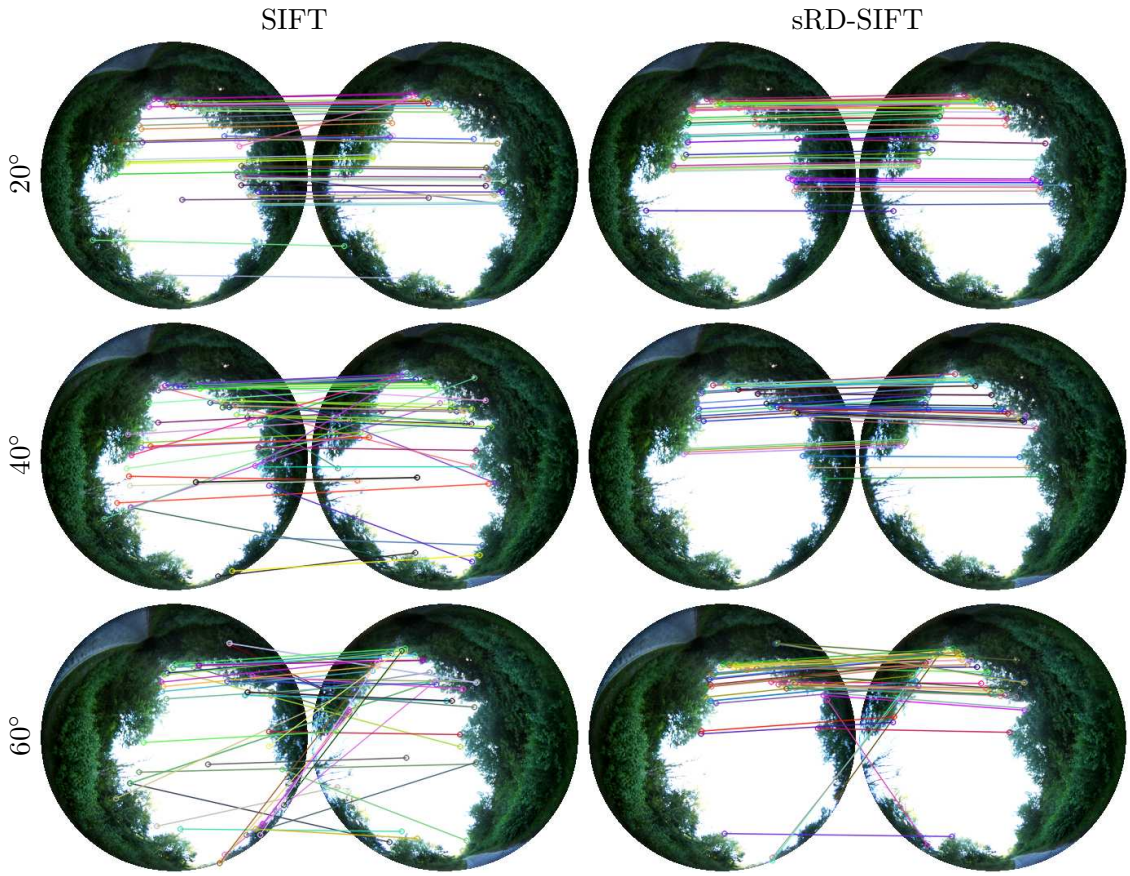
For comparison with our visual 3D compass, we tested the feature-based methods ORB, SIFT, and sRD-SIFT — combined with the Stewenius five-point algorithm and RANSAC — to rotationally align panoramic fish-eye images (section 5.4). Due to the limited generalization of the underlying division camera model for sRD-SIFT, a calibration of the parameter ξ was not possible. Instead we show the performance for varying values of ξ. As input we used raw camera images collected at 10 different locations with a resolution of 1310×1310 pixel captured using a fish-eye lens (example 5.6). At each location, we collected images at tilt angles of $-30°$ to $30°$ in $10°$ steps using a tripod with adjustable orientation. This allows us to perform rotational realignment on image pairs with a rotational offset of up to $60°$. Note that the orientation of the camera (tripod with adjustable orientation) was set by hand such that a small error of the ground truth of around 1 degree is likely.

As before, we use the characteristics, i.e. the failure rate and rotational difference after correction, as described in section 5.3. Figure 5.4 shows the performance of the feature-based methods. Since RANSAC is non-deterministic, we estimated the rotation for each image pair 100 times and calculated the average performance over all estimates. As can be seen, for all feature-based methods the mean failure rate is nearly zero. The performance is nearly equal for all tested methods for tilt angles of up to $20°$. However, for tilt angles of $30°$ or higher, the performance of ORB and SIFT degrades noticeably. For the highest tilt angle of $60°$, the performance is $(0.1\%, 23.8°)$ and $(0.0\%, 11.8°)$ for ORB and SIFT, respectively. In contrast, the performance for sRD-SIFT with $\xi = 10^{-9}$ is only $(0.0\%, 2.5°)$. The results show that the radial distortion correction of sRD-SIFT markedly reduces the distortion effects of the fish-eye lens. The best method *W220CTHs* (tested on the larger *mixed_** dataset, compare section 5.5.1) of the visual 3D compass achieves the same performance $(0.1\%, 2.5°)$.

5.6 Discussion

The experiments show that the visual 3D compass can be used to realign rotationally misaligned panoramic images. However, the performance of the visual 3D compass strongly relies on the chosen methods and parameters as well as the input images. For the same-database and cross-database experiments (sections 5.5.1 and 5.5.2), methods using weighting functions clearly outperform methods which fill in noise or use hemispherical continuation. Furthermore, for panoramic images with an opening angle of $220°$, methods using weighting functions perform nearly as good as the method *complete* which requires full-spherical images. This makes methods using weighting functions a suitable choice for rotational image alignment of panoramic images captured at the same location. However, methods using weighting functions suffer stronger from translational

Example 5.6: SIFT versus sRD-SIFT on Raw Images



Comparison between SIFT and sRD-SIFT on panoramic images. The panoramic images have a resolution of 1310x1310 pixel and were captured using a fish-eye lens (raw images, no preprocessing is applied). Each image pair shows the 50 best feature matches (around 2000 features are found in each images) by using SIFT and sRD-SIFT as feature detector/descriptor for tilt angles of 20°, 40°, and 60° (colored lines). As can be seen, the number of false matches increases with an increasing tilt angle for both methods. However, for large tilt angles — which enforces the matching of features which suffer from strong distortion effects — sRD-SIFT (here with distortion parameter $\xi = 10^{-9}$) noticeably improves the quality of matches compared to SIFT.

effects of the camera than methods which fill in noise (section 5.5.3). For homing, i.e. images captured at different locations, methods which fill in noise might therefore be preferred. Methods which use hemispherical continuation excel due to their low computation times (table 5.4) and can be used for fast rotational alignment of subsequently captured images with an opening angle of 180°.

The performance of the visual 3D compass can furthermore be increased by increasing the maximal number of bands or choosing a finer resolution of the search space. As can be seen in table 5.3, the number of bands has only a small impact on the performance of the visual 3D compass, however the resolution of the search steps has a crucial impact on the mean failure rate. Here, the mean failure rate of the tested method can nearly be halved using search steps of 8° instead of 32° for the most coarse search phase. This is due to the reduced risk of the finer search space resolution to find a local minimum instead of the global minimum, in which case the visual 3D compass would give a wrong estimate.

	$\mathbf{r = 32°}$	$\mathbf{r = 16°}$	$\mathbf{r = 8°}$
$\mathbf{L = 20}$	$(22.9\%, 4.6°)$	$(13.0\%, 3.6°)$	$(12.2\%, 3.8°)$
$\mathbf{L = 30}$	$(22.7\%, 4.1°)$	$(13.0\%, 3.1°)$	$(11.4\%, 3.1°)$

Table 5.3: By varying the maximal number of bands L and the resolution r of the search space (i.e. the coarsest search phase), the performance of the visual 3D compass can be improved. The table shows the performance of the method *F220Cths* with an opening angle of 220°, constant noise, tangent distance disabled, and without preprocessing on the database *mixed_** . For all tests, the size of the search space (maximal rotation angles) is equal to the *large* search space.

Method	**SS: Large**		**SS: Small**	
	TD: Off	**TD: On**	**TD: Off**	**TD: On**
Fill/Complete	2.3	16.3	1.7	14.7
Hemi	1.2	7.8	1.0	7.1
Weighted	24.5	38.7	10.7	23.9

Method	**Detector**	**Descriptor**	**Brute-Force Matcher**	**Stewenius & Ransac**	**Total**
ORB	81.3	49.7	10.4	81.3	222.7
SIFT	1239.5	1415.5	1021.8	252.6	3929.4
sRD-SIFT	7811.1	—	648.2	112.4	8571.7

Table 5.4: Computation times of the visual 3D compass and the feature-based methods in milliseconds on an Intel(R) Core(TM) i7 CPU 870 @2.93 GHz (using a single core only). For the visual 3D compass, the computation times are shown for different settings of the search space and tangent distance. The remaining settings are set to the default parameters described in section 5.3. For the feature-based methods, the computation times for each phase are shown. Since the used sRD-SIFT implementation is closed source, only the combined timings of the detection and description phases could be measured. Note that the brute-force matcher can be replaced with a faster approximative matcher for high-dimensional data (Muja and Lowe, 2014).

In section 5.5.4 we examined the performance of feature-based methods. The results indicate that the best feature-based methods sRD-SIFT and the visual 3D compass perform comparably during the same-database on panoramic images with an opening angle of 220°. For example, using the *small* search space (figure 5.2, (b)) the error rates for sRD-SIFT $(0.0\%, 2.1°)$ and the method *weighted* $(0.3\%, 1.8°)$ only differ slightly. However, we were not able to make a comparison between feature-based methods and the visual 3D compass on cross-databases since the effort to collect cross-databases with a tripod is too demanding. The effect of lighting and seasonal changes on feature-based methods is examined in more detail in Wu et al. (2013), Mikolajczyk et al. (2005), and Valgren and Lilienthal (2007), respectively. The main advantage of feature-based methods is that they are – at least to some amount — capable to correctly estimate rotations and translations between panoramic images. This is currently not possible for the visual 3D compass. However, the downside of feature-based methods is the comparably high computation time required (table 5.4).

First tests of the visual 3D compass on a Raspberry Pi 3 (ARM Cortex-A53 Quadcore @1.2 GHz; using a single core only) show that it can be run in real-time on low-cost hardware: Using the *small* search space, the methods *fill*, *hemi*, and *weighted* run with around 100 Hz, 200 Hz, and 10 Hz, respectively. The visual 3D compass could be used to rotationally align a robot with on a previously driven route such that only the drift of the robot needs to be corrected. Note that our approach can deal with a significant amount of blur as can be seen in example 5.1 due to the comparably low number of bands (frequencies) used for the RSH. The main error sources for the visual 3D compass are large rotational offsets and translations between the panoramic images. Both errors can be reduced using a high frame rate: Assuming a frame rate of 30 Hz, the rotational offset between subsequently captured images should remain small, even for a fast

moving remote control car (RCC). Assuming a speed of 20 km/h the RCC would travel less than 20 cm between two images which should have only small impact on the compass estimate even for narrow environments.

Future work will be focused on the comparison of the visual 3D compass with feature-based methods under varying lighting conditions. Other applications for a visual 3D compass besides navigation are possible, e.g. registration of 3D point clouds (Makadia et al., 2006) or morphological structures (Shen et al., 2009). These applications use full-spherical data and do not have to deal with filling in missing data, simplifying the process of Fourier Transform as well as the visual 3D compass search.

5.7 Conclusion

We presented different techniques necessary to implement a real-time visual 3D compass for rotationally aligning two panoramic images captured in featureless environments and examined its performance. We could show that the rotational difference between two rotationally misaligned panoramic images can be reduced effectively using low-cost hardware only, making it attractive for the navigation of MAVs and rapidly moving wheeled robots as well as general image registration.

CHAPTER 6

3D-Warping

Visual odometry is the task of determining the relative pose of a robot between two or more locations by analyzing their corresponding camera images. Commonly, one of these images is the current camera image (current view) and the other image was captured at some point of interest (snapshot), e.g. a loading station. To return to a previously visited location, we commonly do not need the complete relative pose, but only the direction — represented by a home vector — in which the snapshot was captured. Several methods have been proposed to determine the home vector: Feature-based methods extract, describe, and match visual features in image pairs. These methods are widely used in the field of robotics and are considered as highly precise. However, these methods often suffer from distortion effects, motion blur, or highly repetitive and featureless environments as for example forests. An alternative approach is to determine the home vector by simulating and evaluating a set of possible movement hypotheses of the robot. These methods are referred to as warping methods and have been successfully implemented and tested for indoor robots restricted to planar movement, e.g. domestic cleaning robots. However, the limitation to planar movement leads to errors as soon as the robot is tilted, making it unsuitable for applications with non-planar movement, particularly in outdoor environments. In this chapter we suggest a generalization of the warping method to lift the restriction to planar movement.

6.1 Introduction

A typical task in autonomous navigation is the return of a robot to a previously visited location. The task of determining the direction from the robots current location towards a known location is called **homing**. Using visual input only, this task is also referred to as **visual homing** (section 1.2.2). For visual homing, panoramic images as captured by a skywards-facing fish-eye camera are commonly used. We refer to panoramic images captured by the robot at its current location and goal location as **current view** and **snapshot**, respectively. The basic idea of warping is to internally simulate movements of the robot from the current location to another location (**movement hypothesis**) by distorting (**warping**) the current view. By searching for the movement hypothesis which minimizes some image distance measure between the warped current view and the snapshot, the home vector can be determined. In applications where planar movement can be assumed, e.g. for domestic cleaning robots, min-warping and feature-based methods achieve comparable results (Fleer and Möller, 2017). However, in contrast to feature-based methods which do not restrict the movement of the robot, 2D-warping and min-warping strongly suffer from non-planar movement.

Using the spherical harmonics introduced in chapter 3, we show in section 6.3 how the idea of 2D-warping can be generalized to simulate both rotations and translations of the robot in 3D space. This gives us the possibility to lift the restriction to planar movement.

6.2 Introduction to Warping

Warping is a holistic method to determine the home vector pointing from the current view to the snapshot. The basic idea is to simulate how the current view would look like if the robot moves in

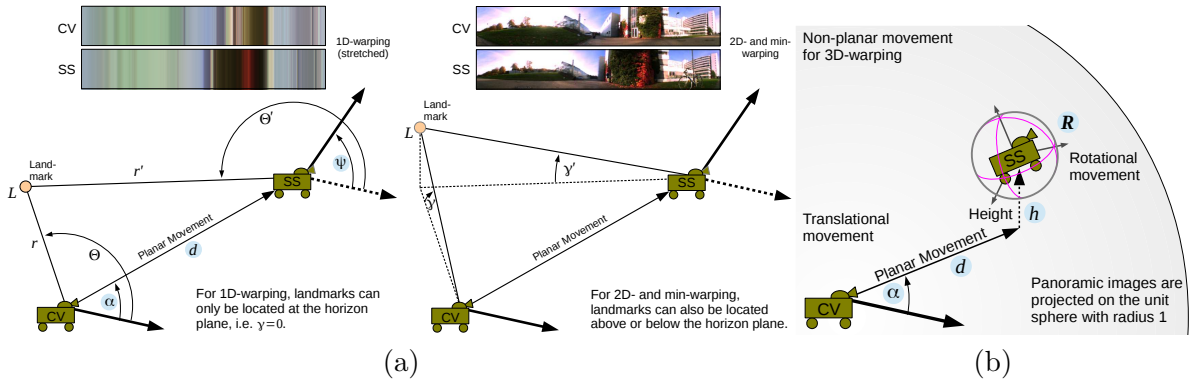


Figure 6.1: (a) The sketch shows a wheeled robot which captured panoramic images at its current location (current view, CV) and some goal location (snapshot, SS). The movement from the current view to the snapshot can be parameterized by a triplet (α, d, ψ), where α is the direction in which the robot moves, d the distance, and ψ the change of heading. The position of a landmark L as seen by the robot at its current location can be expressed as triplet (θ, γ, r) by its azimuth angle θ, elevation angle γ, and distance r relative to the robot's location. After the robot moved, the landmark is located at (θ', γ', r'). For 1D-warping, it is always assumed that $\gamma = 0$. The sketch is adapted from Möller et al. (2010). (b) For 3D-warping, we divide movements into their translational and rotational parts which are parameterized by (α, d, h) and $\mathbf{R}$, respectively. The current view of the robot is projected on the unit sphere with radius 1 (equal-distance assumption); we only choose translation parameters (α, d, h) which do not move the robot outside of the unit sphere.

a specified direction. This can be achieved by warping the current view accordingly to simulated movements.

The first implementation of warping uses one-dimensional panoramic images (1D-warping, Franz et al. (1998)) and is sketched in figure 6.1. These images only contain a panoramic image with a single row (horizon plane) or can, alternatively, be obtained by averaging column-wise a panoramic image. In the following it is assumed that each pixel i in the panoramic image corresponds to a landmark L_i, represented by a triplet $(\theta_i, \gamma_i, r_i)$, where θ_i is the azimuth angle, γ_i the elevation angle, and r_i the distance relative to the robot. The azimuth and elevation angles θ_i and γ_i between the robot and the landmark are given by the pixels coordinates in the panoramic image. Since 1D-warping assumes that all landmarks are in the robot's horizon plane, we set $\gamma_i = 0$; this limitation is lifted by 2D-, min-, and 3D-warping. The distance r_i is unknown, therefore the location of the landmark cannot uniquely be determined. To avoid this ambiguity, 1D-warping assumes that the distance to each landmark is $r_i = 1$ (**equal-distance assumption**). As a consequence, the location of each landmark L_i is known and we can simulate the effects of arbitrary movement of the robot on the image.

The movement of the robot — which is restricted to planar movement — is parameterized relative to its pose by the triplet (α, d, ψ), where α is the direction in which the robot moves, d the distance, and ψ the heading at the goal location. Using trigonometry, a closed form solution can be derived to calculate the triplet $(\theta_i', \gamma_i', r_i')$ at which the landmark L_i will be located after the movement. An exhaustive search over a set of movement hypotheses (α, d, ψ) for the movement which minimizes a pixel-wise distance measure between the warped current view and the snapshot can then be used to determine the home vector.

As shown by Franz et al. (1998), 1D-warping can be used to determine home vectors in the proximity of the snapshot location. However, 1D-warping has to deal with multiple problems: First, only a small fraction of the available information in panoramic images is used. Second, the equal-distance assumption is a vast simplification of the environment's geometry. Third, landmarks visible in the snapshot are not necessarily visible in the current view (occlusions, moving objects).

The first problem was addressed by Möller (2009); instead of using a single row in the

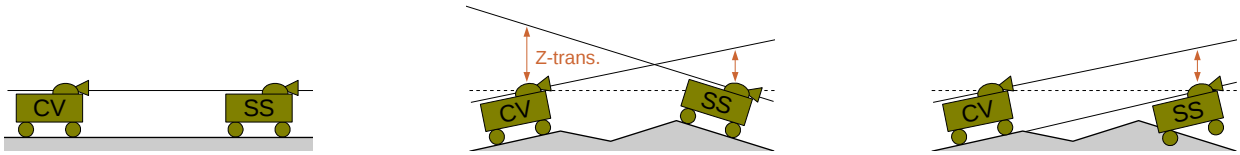


Figure 6.2: Assuming planar movement (left panel), the optical centers of the robot at the current view and snapshot are located in a common horizon plane (black line). The methods 2D- and min-warping simulate planar movement of the robot such that — at least theoretically — the current view could be warped into the snapshot. However, if the robot is tilted (center panel), additional translations along the Z-axis appear (red arrows) which cannot be compensated by 2D- and min-warping. Moreover, rotationally aligning the current view with the snapshot (e.g. by using the visual 3D compass) does not necessarily remove the translational offset (right panel); this would require knowledge about the *absolute* tilt of the robot (e.g. by using an IMU).

panoramic image only, the complete panoramic image is used. While the underlying theory is basically similar to 1D-warping, the movement of the robot now also affects the elevation angle between the robot and each landmark L_i. This can be taken care of by vertically scaling image columns accordingly to the robots movement. As homing experiments show, 2D-warping performs noticeably better than 1D-warping.

Moreover, it can be shown that the equal-distance assumption can — at least partially — be lifted (Möller et al., 2010): Instead of assuming that the distance to each landmark column is equal, it is only assumed that all pixels *within* the i-th image column have an equal distance r_i. In contrast to 1D- and 2D-warping, which warp the image for various distances d traveled by the robot, min-warping scales each image column j with $j \neq i$ and searches for the best match with image column i[1]. Min-warping shows a superior performance compared to both 1D-and 2D-warping. It has been shown that min-warping can be used to reliably navigate a cleaning robot under challenging conditions, including illumination changes (Möller et al., 2014). In this work we use the implementation by Möller (2016b). The parameters used for both 2D- and min-warping are similar and presented in section 6.5.

A common problem of 1D-, 2D-, and min-warping is the restriction to planar movement. As a consequence, these methods cannot cope with non-planar movement and tilt applied to the robot. As sketched in figure 6.2, tilt applied to the robot commonly also requires warping methods to compensate for Z-axis translations. In this chapter we examine the performance of 2D- and min-warping and our suggested 3D-warping method (section 6.3) under the influence of tilted input images.

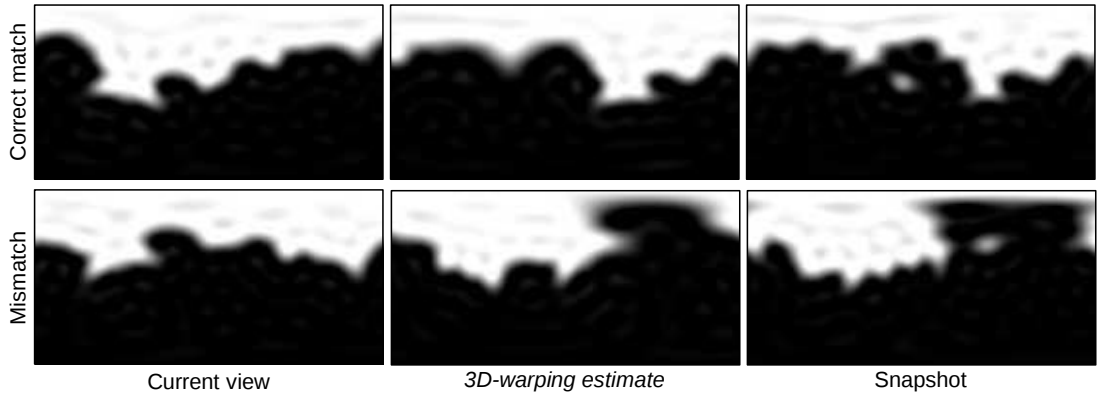
6.3 3D-Warping

In this section we suggest 3D-warping as a generalization of 2D-warping for arbitrary, i.e. non-planar, movement. Our approach uses the basis of real spherical harmonics (RSH) introduced in chapter 3 to represent panoramic images in the frequency domain. As for 2D-warping, we simulate movement of the robot by warping the current view accordingly. In contrast to the parameterization of 2D-warping, we divide movements into their translational and rotational part. The theory required for calculating rotations and translations directly in the basis of RSH is explained in detail in section 3.7 and 3.8, respectively. To compare the warped current views with the snapshot, we use the integral squared error (ISE) from section 3.9.1.

The translational part of each movement hypothesis is now represented by a triple (α, d, h), where the additional parameter h is the robot's change in height. This triple can be represented as vector $\vec{t} = (x, y, z)^T$, which we use to calculate the transformation matrix $\mathbf{T}_{\vec{t}}$ from equation (3.93) to warp current views directly in the basis of RSH. We make use of the equal-distance assumption and project the current view on the unit sphere, i.e. we assume that $r_i = 1$ for all landmarks

[1] By scaling the column j for various scaling factors ω, we indirectly search for the best matching distance ratio $\omega = \frac{r_j}{r'_j}$. Min-warping does not — in contrast to 1D- and 2D-warping — explicitly *warp* the current view anymore.

Example 6.1: 3D-Warping



Similar to the visual 3D compass, 3D-warping simulates various movement hypotheses of the robot. By warping the current view (left column) accordingly for each movement hypothesis, we can search for the warped current view (center column) which is most similar to the snapshot (right column). A current view and snapshot, which were captured around 8 m apart (database *meadow*), are shown with different random orientations of current view and snapshot (top row and bottom row). While for the images shown in the top row a correct match was found, the images in the bottom row are matched incorrectly. Therefore, only for the top row a correct home vector can be estimated.

L_i. Therefore, all movement hypotheses (α, d, h) are relative to the unit sphere and have to fulfill $\|\vec{t}\| < 1$. Recalling section 3.8.3, translations can be interpreted differently — e.g. for translations of point clouds (interpretation *density*) or panoramic images (interpretation *visual*) — using an appropriate weighting function. For 3D-warping, we use the interpretation *visual*. The rotational part is represented by a rotation matrix $\mathbf{R}$. A visualization of the translational and rotational movement of the robot is sketched in figure 6.1, (b).

The complete 3D-warping algorithm consists of four stages: First, a visual 3D compass (chapter 5) is used to obtain a coarse rotational alignment between the current view and snapshot. This step is optional and can be applied to all warping methods. Second, for each movement hypothesis the current view is warped by applying the translation (α, d, h). Third, the warped current view is rotated by applying the rotation $\mathbf{R}$. Fourth, we search for the movement hypothesis which minimizes the ISE between the warped and rotated current view and snapshot. Note that for a systematic search the third phase is a visual 3D compass. Example 6.1 shows a correct and an incorrect match between a warped current view and a snapshot.

6.4 Experiments

In this chapter we evaluate the quality of the homing methods 2D-, min-, and 3D-warping for tilted panoramic images. All three warping methods can be either used directly or, optionally, the visual 3D compass is applied beforehand to rotationally align the snapshot with the current view (figure 6.3, (a)). Since we are especially interested in the performance of the warping methods on skyline-segmented images, the methods are tested on the databases *uni_early* and *meadow* for which skyline-segmented images are available (appendix D). The parameter sets used for all methods are described in detail in section 6.5.

To examine the influence of tilt, the current views and snapshots are artificially tilted by α between 0° and 45° in 5° steps. To include random heading directions of the robot, the tilt applied to a pair of current view and snapshot is computed as follows: Both the current view and snapshot are rotated randomly around the Z-axis. Then the current view and snapshot are tilted by $\alpha/2$ in

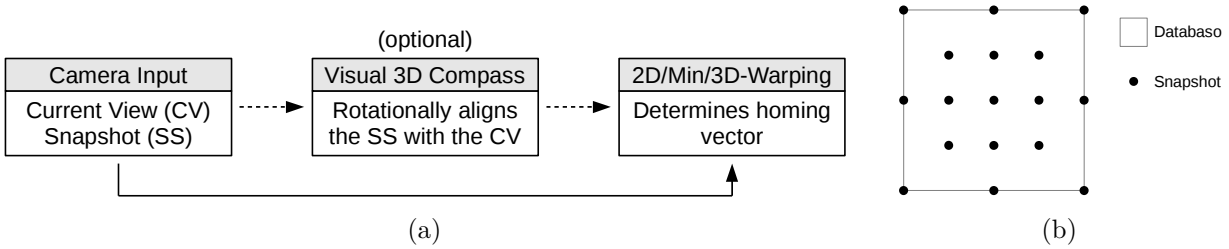


Figure 6.3: (a) To determine the home vector using 2D-, min-, or 3D-warping, we either use the warping method directly or optionally apply the visual 3D compass previously to align the snapshot with the current view. Note that due to translation between the current view and snapshot, the rotational alignment of the visual 3D compass can be erroneous. (b) To evaluate the homing performance of the various warping methods, we calculate the home vector from each image in a database to a set of 17 snapshots. The database images closest to the grid locations are used as snapshots; if the database is rectangular, the locations are stretched accordingly. The images closest to these locations are chosen.

opposite directions, where the tilt directions are given by β and $\beta + \pi/2$ for some random angle β (tilt is described in detail in section 3.3.2). Finally, the current view and snapshot are again rotated randomly around the Z-axis.

For both databases *uni_early* and *meadow* we performed the following experiment: First, we chose a set of 17 snapshot locations distributed as shown in figure 6.3, (b). Second, we calculated the home vectors from each location in the database (current views) to each of the 17 snapshot positions. As described above, the current view and snapshot are rotated as described above. The experiment is repeated five times to obtain a higher variety of random robot orientations between the current views and snapshots. Moreover, we tested the warping methods by applying the preprocessing techniques *hs*, *HS*, and *sky* to each image in the databases (see section 6.5 for more details).

6.5 Parameter Sets

The 2D-warping and min-warping parameters used are listed in table 6.1. Except for the camera opening angle and precision, these parameters were used in the study by Fleer and Möller (2017) and achieved good performance for indoor navigation. An overview over the most important parameters[2] used for all warping methods is shown in table 6.1. We optionally use a visual 3D compass to rotationally align the snapshot with the current view; the used parameters are presented in table 6.2.

For the experiments we use panoramic images with an opening angle of 220°. Since we use the basis of RSH to represent panoramic images, our 3D-warping method requires full-spherical panoramic images. Motivated by our results from chapter 5, we do not use weighting functions for the visual 3D compass since they increase the susceptibility to errors from translational effects. As for our localization method (chapter 4) and visual 3D compass (chapter 5), we fill-in the panoramic images (*hs*, *HS*) with noise and the skyline-segmented images (*sky*) with "ground". For 2D- and min-warping, no adjustments are necessary except when the visual 3D compass is previously applied: The current implementations of 2D- and min-warping cannot cope with tilted panoramic images (due to tilt, image regions would be cropped or needed to be filled-in with data). In this case we resample the tilted panoramic image from the full-spherical panoramic image to avoid undefined image regions.

For the experiments in section 6.4 we use three different preprocessing techniques; that is *hs* (no preprocessing), *HS* (histogram equalized with subsequent Sobel-filtering), and *sky* (skyline-segmented). Fore more details on the preprocessing techniques *hs* and *HS* see section 3.11.2. For a detailed description of skyline-segmented images and skyline segmentation techniques see

[2] An implementation of 3D-warping and the visual 3D compass used in this section — where all parameters as for example the exact coarse-to-fine parameters are shown — can be found in the appendix (section C.3).

2D-/min-warping		3D-warping	
Parameter	**Value**	**Parameter**	**Value**
Unfolded image size	288×176	Unfolded image size	180×90
Camera opening angle	220°	Camera opening angle	220°
n_α, n_ψ	96	Bands / Sampl. Points	16 / 10^4
Scale planes	9	Translation direction α	15° steps
Max. scale factor	2.0	Relative translation d	$0.05, 0.1, \ldots, 0.3$
Max. threshold	2.5	Relative translation h	$-0.3, -0.15, \ldots, 0.3$
Searcher	Full	Rotation (azimuth)	8° steps
Precision	Floats	Rotation (max. tilt)	±14°
Distance Measure	NSAD	Coarse-to-fine	On
ρ range	Off (0-100)	Weighting Functions	Off
Fine search	Off	Noise	Constant
Interpolation	Off	Tangent Distance	Off

Table 6.1: Parameter sets used for 2D- and min-warping (left) and 3D-warping (right). For 3D-warping, only an overview of the parameters is shown. The complete parameter sets — including the exact search-to-fine parameters — can be found in the appendix (section C.3).

Visual 3D compass	
Parameter	**Value**
Unfolded image size	180×90
Camera opening angle	220°
Bands / Sampl. Points	16 / 10^4
Rotation (all axes)	4° steps
Rotation (max. tilt)	±35°
Coarse-to-fine	On
Weight. Functions	Off
Noise	Constant
Tangent Distance	On

Table 6.2: Parameter sets used for the visual 3D compass; note that only an overview of the parameters is shown. The complete parameters — including the exact search-to-fine parameters — can be found in the appendix (section C.3).

chapter 2. The preprocessing techniques *hs* and *HS* are only applied to 3D-warping and the visual 3D compass. Since 2D- and min-warping already internally use preprocessing techniques (edge-filtering), our preprocessing techniques are not applied to 2D- and min-warping. Edge-filtering applied to skyline-segmented images is detrimental for 2D- and min-warping, therefore edge-filtering is deactivated for skyline-segmented images.

6.6 Results

As described in section 6.4, we calculated home vectors using 2D-, min-, and 3D-warping for the databases *uni_early* and *meadow*. The mean home vector errors over all tilt angles and image pairs (current views and snapshots) are presented in table 6.3. As can be seen, all methods perform better by previously applying the visual 3D compass, which rotationally aligns the current view with the snapshot. There is no best warping method for all tested databases and preprocessing techniques, however 3D-warping performs best on skyline-segmented images. The overall good performance of all warping methods on skyline-segmented images underlines the importance of the skyline as illumination- and rotation-invariant landmark (chapter 4).

More detailed results for homing experiments are shown in the figures 6.4 and 6.5. The figures show the homing performances of all warping methods for varying tilt angles α and distances between the current views and snapshots. Note that we either fixed a tilt angle or a distance and computed the mean home vector error over all distances or tilt angles, respectively. Moreover,

Method	DB: uni_early			DB: meadow		
	hs	HS	Sky	hs	HS	Sky
2D-warping	64°	64°	47°	57°	57°	54°
min-warping	60°	60°	52°	53°	53°	56°
3D-warping	57°	44°	23°	43°	56°	42°
VC + 2D-warping	40°	37°	11°	57°	56°	48°
VC + min-warping	40°	37°	16°	50°	51°	50°
VC + 3D-warping	51°	36°	10°	40°	55°	37°

Table 6.3: The mean home vector errors for all tested warping methods (with and without previously applying the visual 3D compass, VC) are shown for the databases *meadow* and *uni_early*. The home vector errors were averaged over all tilt angles $\alpha = 0°, 5°, \ldots, 45°$. For both databases, the experiments were performed with three different preprocessing techniques: No preprocessing is applied (*hs*), the images are histogram equalized and afterwards Sobel-filtered (*HS*), and skyline-segmented images (*sky*). The best performing methods are highlighted (red text color).

we tested different preprocessing techniques; namely non-preprocessed images (hs), histogram equalized and Sobel-filtered images (HS), and skyline-segmented images (sky). As the results from table 6.3 suggest, the overall best home vector accuracy is achieved using skyline-segmented images. Only on the database *meadow*, non-preprocessed images outperform skyline-segmented images as long as the tilt angle is small ($\alpha \leq 10$). For tilt angles $\alpha \geq 10°$, this disparity vanishes. By applying the preprocessing technique HS, the accuracy of 3D-warping and the visual 3D compass increases on the database *uni_early* and decreases on the database *meadow*. However, the improvement on *uni_early* using the preprocessing technique HS is rather small in comparison to skyline-segmented images. We are interested in both large tilt angles α and illumination-invariance without decreasing the home vector accuracy, therefore we focus in the following on skyline-segmented images.

As we have seen in chapter 5, translations between the current view and snapshot decrease the accuracy of the visual 3D compass estimate. This negatively impacts the homing performance for non-tilted images if the visual 3D compass is previously used: As can be seen in figure 6.5, for the database *meadow*, the mean home vector error for 2D- and min-warping increases for $\alpha = 0$ from around 30° to 45°, only 3D-warping is not affected and has in both cases an error of around 35°. For the database *uni_early* (figure 6.4), the overall performance is noticeably better and the influence of the visual 3D compass on non-tilted images is comparably lower.

Moreover, it can be seen that the influence of an increasing distance between the current view and snapshot on the mean home vector error differs for the databases *uni_early* and *meadow*. For the database *meadow*, the mean home vector error rapidly increases for distance greater than 8 m. In contrast, the increasing distance has nearly no impact on the database *uni_early*. This is likely a result of the different environments: While the database *uni_early* was captured on a street with a large distance to most surrounding objects, the database *meadow* is mostly surrounded by (partially overhanging) trees (see appendix D). Since 3D-warping performs noticeably better on the database *meadow* — especially for distances up to 10 m — than 2D- and min-warping, it seems that 3D-warping copes better with close objects.

For a visual impression, figure 6.6 shows the home vector fields for a single snapshot on the databases *uni_early* and *meadow* using skyline-segmented images. As can be seen, min-warping without previously applying the visual 3D compass performs best on non-tilted images. Especially for current views captured at large distances from the snapshot accurate home vectors can often be computed. Only in regions close to the border of the database *meadow* the home vectors become erroneous. In contrast, 3D-warping estimates accurate home vectors only for current views with a distance of 10 m or less to the snapshot. However, for tilted current views and snapshots (here: tilt $\alpha = 45°$), the performance of 3D-warping is only reduced slightly while min-warping completely fails. Using min-warping in combination with the visual 3D compass increases the robustness

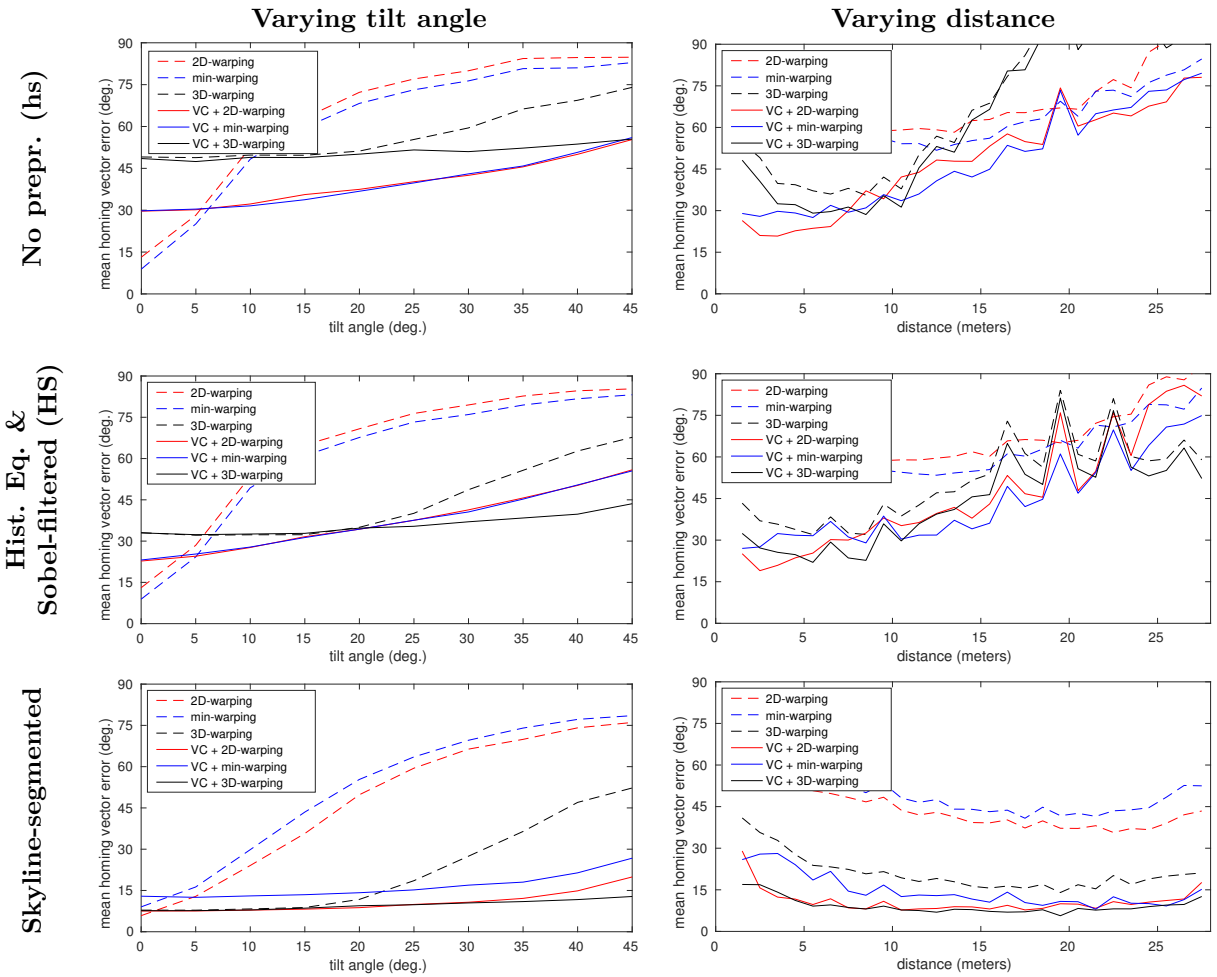


Figure 6.4: The plots show the mean home vector error in degrees either as a function of the applied tilt in degrees (left column; $\alpha = 0°, 15°, \ldots, 45°$) or distance between the current view and snapshot (right column; histogram, 1 m bins). Note that for fixed tilt angles (left column) the data is averaged over all distances; for fixed distances (right column) the data is averaged over all tilt angles. Each warping method is shown twice, once with (solid lines) and without (dashed lines) applying the visual 3D compass previously. Each row refers to the type of preprocessed image used. The results shown were computed on the ***uni_early*** database.

against tilt, however it still performs worse than 3D-warping.

6.7 Discussion

The results show that 2D- and min-warping reliably estimate home vectors from non-preprocessed panoramic images in outdoor environments as long as the robot is limited to planar movement and not tilted. For both databases *uni_early* and *meadow*, the average home vector error was around 15° and 10° for 2D- and min-warping, respectively. Especially the meadow database is challenging for 2D- and min-warping since it spans a large area, contains both close and distant objects, and has overhanging trees. In comparison, 3D-warping performs noticeably worse with an error of around 50° and 35° for the databases *uni_early* and *meadow*, respectively. For robots limited to planar movement, these results indicate that 3D-warping is not a preferable choice for visual homing using non-preprocessed panoramic images in outdoor environments.

As soon as tilt is applied to the robot, 3D-warping performs best — followed by 2D- and min-warping — on both non-preprocessed images and skyline segmented-images. Overall, best results are obtained using skyline-segmented images. Interestingly, 2D-warping performs better than min-warping on skyline-segmented images, which is contrary to the performance of 2D- and

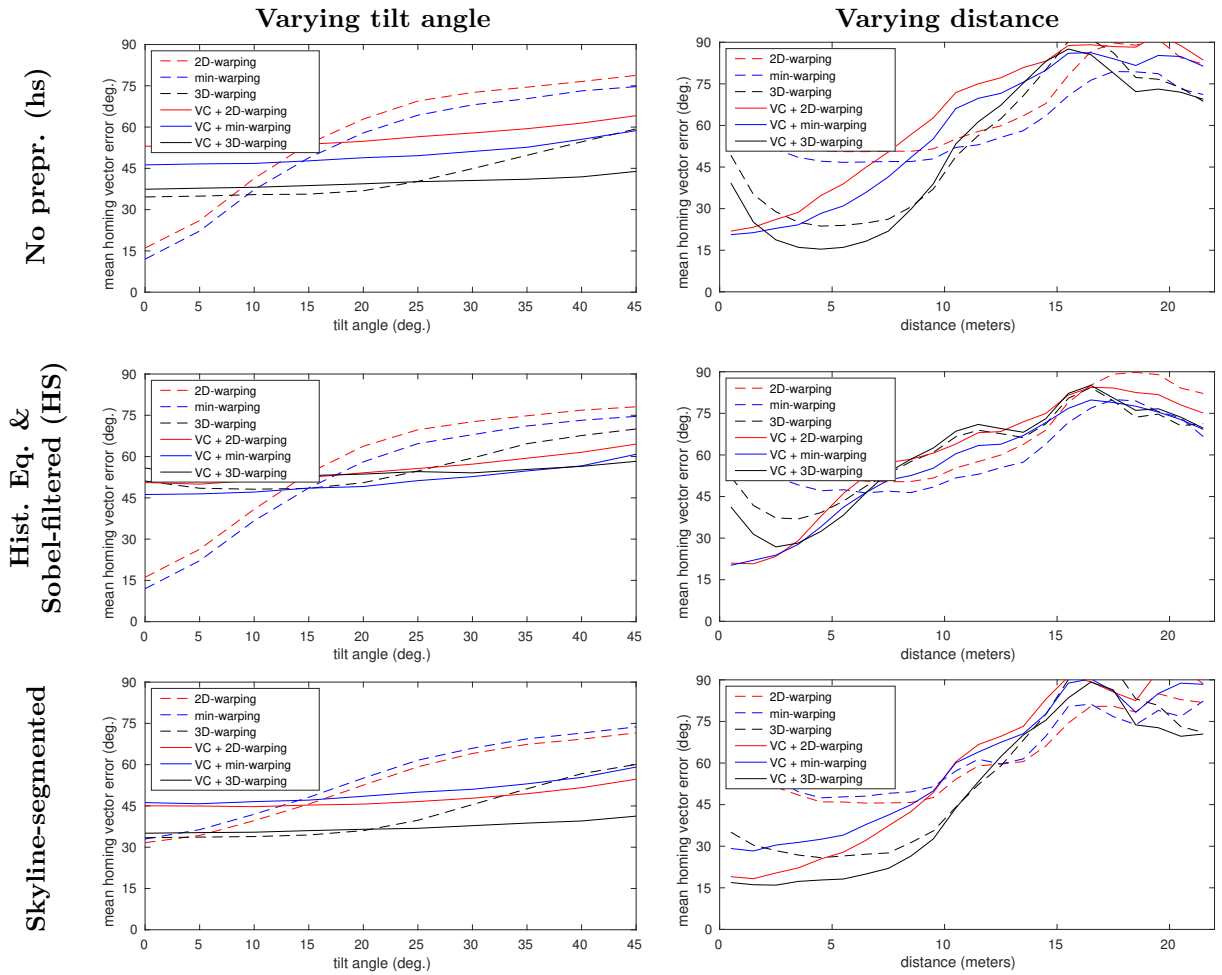


Figure 6.5: The plots show the mean home vector error in degrees as a function of the applied tilt in degrees (left column; $\alpha = 0°, 15°, \ldots, 45°$) or distance between the current view and snapshot (right column; histogram, 1 m bins). Note that for fixed tilt angles (left column) the data is averaged over all distances; for fixed distances (right column) the data is averaged over all tilt angles. Each warping method is shown twice, once with (solid lines) and without (dashed lines) applying the visual 3D compass previously. Each row refers to the type of preprocessed image used. The results shown were computed on the ***meadow*** database.

min-warping on non-preprocessed images. A reason for this could be that for 2D-and 3D-warping a spatial relation between neighboring columns exists, this spatial relation is lifted by min-warping. Without previously applying the visual 3D compass, the accuracy of 2D- and min-warping strongly decreases with an increasing tilt of the robot. In contrast, 3D-warping is robust against tilt of up to 15°. For tilt angles of more than 15°, all warping methods require previously rotational alignment of the current view and snapshot using the visual 3D compass. The results show that by applying the visual 3D compass beforehand, all warping methods achieve a nearly constant home vector error for tilt angles up to 45°. Due to the erroneous rotation estimation of the visual 3D compass if the current view and snapshot have a translational offset, the visual 3D compass can actually increase the rotational offset between the current view and snapshot. Especially for tilt angles smaller than 15° the visual 3D compass can therefore negatively impact the home vector estimate.

Skyline-segmented images are illumination-invariant and showed better performance than images which are histogram equalized and Sobel-filtered. Moreover, skyline-segmented images are rotation-invariant — as long as all ground objects are in the field of view — making them a suitable choice for visual homing in outdoor environments. A concern with skyline-segmented images is,

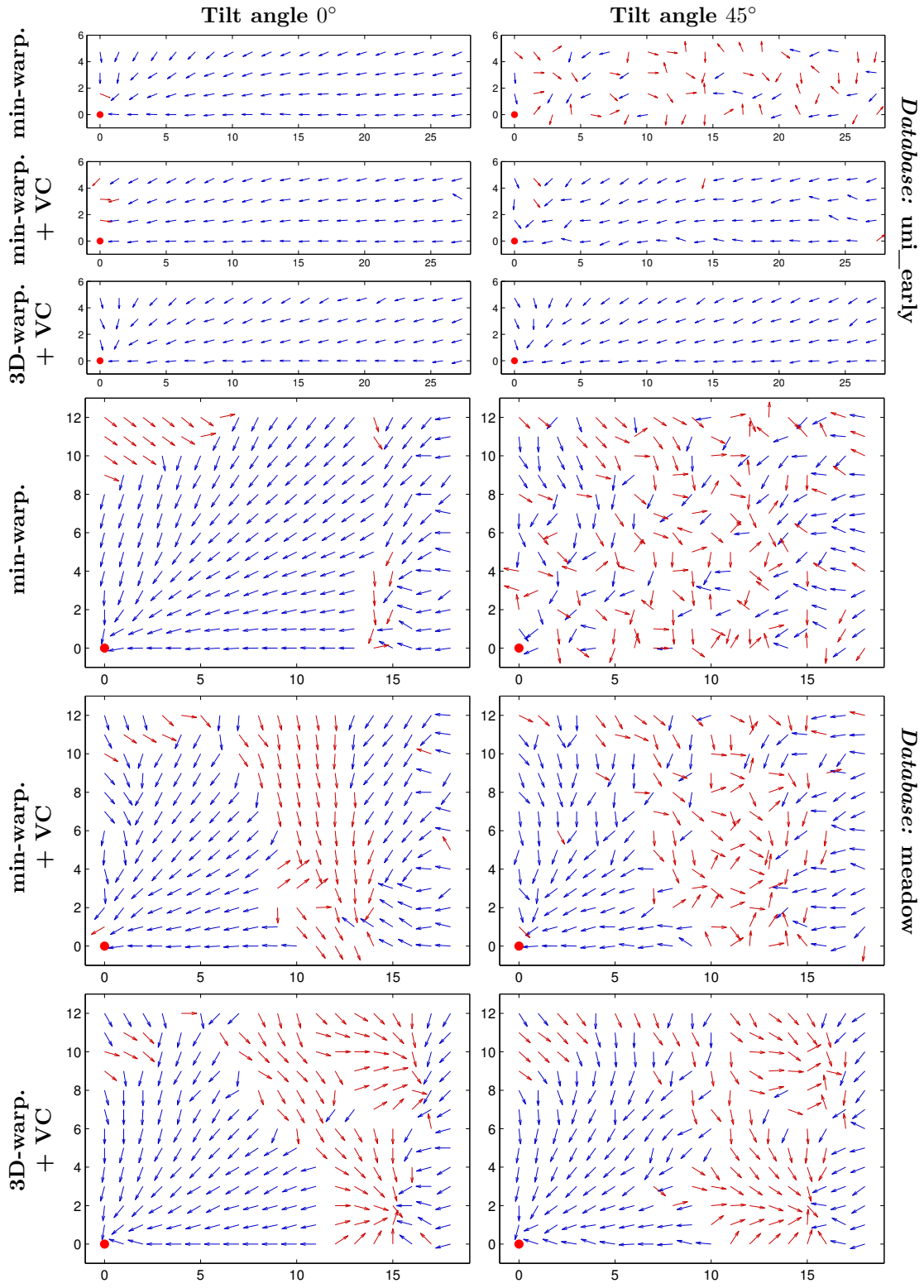


Figure 6.6: Home vector fields for the methods min-warping (with and without visual 3D compass) and 3D-warping with simulated tilt of 0° and 45° on the databases *uni_early* and *meadow* using **skyline-segmented** images. The home vectors (arrows) are determined for each current view and should, in the best case, point towards the exemplary snapshot position (red dot). Home vectors which differ by more than 45° from the optimal home vector are colored red.

that they provide a comparably small and inconclusive amount of information compared to RGB images. As for our visual localization technique from chapter 4, this concern was not confirmed. Quite the converse could be observed for 2D- and 3D-warping on the database *uni_early*, where the home vector error was strongly reduced in comparison to non-preprocessed images.

The average computation time of the visual 3D compass is around 37 ms on an Intel(R) Core(TM) i7 CPU 870 @2.93 GHz. The computation times of 2D- and min-warping are around 102 ms and 162 ms, respectively; 3D-warping requires around 891 ms. A more extensive examination of the parameter set used for 3D-warping would be required to optimize the computation times of 3D-warping. For example, we used a rather unspecific set of movement hypotheses (for both translations and rotations) which could be reduced. Note that for the experiments in this chapter 2D-, min-, and 3D-warping are using implementations based on *floats*; for 2D- and min-warping the precision can be changed to *chars* which approximately divides the computation time by three. This approach works well for indoor environments, however detailed tests for outdoor environments — in which an increased range of brightness values is encountered — are currently missing. In future works it could be examined if 3D-warping and the visual 3D compass could also be implemented using *integers* or *chars* to reduce the computation times.

Together with our suggested localization method from chapter 4, 3D-warping would allow us to perform more complex visual navigation tasks as route following (repeatedly homing towards a sequence of snapshots) on skyline-segmented images. The experiments in this chapter were performed using panoramic image databases. To examine the performance of 3D-warping in real environments, robot experiments would be required.

6.8 Conclusion

In this chapter we suggested 3D-warping as a generalization of 2D-warping for non-planar movement and could show that 3D-warping can be used to compute home vectors on outdoor databases. As the results show, for planar movement the competing methods 2D- and min-warping commonly perform better than 3D-warping. However, for non-planar movement 3D-warping often performs better than 2D- and min-warping. Especially using skyline-segmented images — which are illumination- and rotation-invariant — 3D-warping outperforms both 2D- and min-warping. This makes 3D-warping a suitable method for visual navigation in complex outdoor environments which suffer from illumination changes or bumpy terrain; an obvious application could be the navigation of lawn-mower robots.

CHAPTER 7

Overall Summary, Discussion, and Future Work

In this chapter we briefly summarize the main results of this work. A summary of each chapter is given in section 7.1, followed by a discussion of the main results in section 7.2. Finally, we suggest future working directions in section 7.3.

7.1 Summary

The main purpose of this work was to develop methods for autonomous navigation of mobile robots in outdoor environments. The two main problems in outdoor navigation are — in contrast to indoor navigation — that illumination changes greatly affect the appearance of a scene and that robots are not limited to planar movement. The main findings of this work are briefly summarized in this section.

7.1.1 Skyline Segmentation

We extensively examined color contrast vision between the ultraviolet and green channel (UV/G) to classify pixels either as ground objects or sky (*skyline*). Besides constructing a UV/G camera setup, we suggested several methods based on linear thresholding and evaluated their classification quality. The tested thresholding methods can be divided into two categories: Methods which are based on a fixed threshold (called *global separation techniques*), and methods based on an adaptive threshold (called *local separation techniques*).

As our results show, UV/G contrast vision slightly increases the classification quality compared to UV- and green-only vision, however the improvement is only marginal. Moreover, we could show that UV-only vision clearly outperforms green-only vision. Since the construction of a UV/G camera setup is a demanding task, UV/G vision is a promising approach to model insect vision but less advantageous for robot applications; for robot applications we therefore suggest — due to its simplicity — to use UV-only vision. The evaluation of global and local separation techniques showed that global separation techniques are able to reliably separate ground objects from sky during sunny days, however with decreasing light intensities (e.g. clouds, dawn, dusk) the classification quality significantly decreases. In contrast, local separation methods showed superior and nearly constant classification quality over all tested databases despite varying illumination or weather changes. We used the obtained insights to successfully implement and use skyline segmentation on a robot to perform localization tasks, proving that the skyline — which contains considerably less information in comparison to color images — still provides sufficient information for visual navigation tasks.

7.1.2 Spherical Harmonics

We briefly introduced the basics of Fourier analysis on the rotation group $SO(3)$. The acquired tools most importantly allow us to express functions defined on the rotation group $SO(3)$ and the unit sphere S^2 in frequency domain. The bases we use to represent functions on $SO(3)$ and S^2 are the Wigner-D matrices and spherical harmonics (SH), respectively. Since panoramic images can be expressed as functions on the unit sphere, the basis of SH allows us to work on panoramic

images directly in the frequency domain. By representing functions in the frequency domain, we have access to important descriptive information such as the amplitude spectrum and bispectrum. Moreover, the SH are a natural choice to represent functions on the unit sphere; in contrast, panoramic images represented as rectangular images are strongly distorted. Our most important contributions are the derivation of sparsity relations for real Wigner-D matrices around the X-/Y- and Z-axis, the use of symmetries and weighting functions for SH to work with hemispherical and non-hemispherical panoramic images, and the derivation of formulas to approximate translations directly in the basis of SH. Finally, we implemented the theoretical findings and methods in a C++ library with a focus on visual navigation.

7.1.3 Localization

By combining our insights about skyline segmentation with the amplitude spectrum obtained by using spherical harmonics, we show that localization can be performed even in challenging outdoor environments: Our suggested method calculates the amplitude spectrum (rotation-invariant) of the skyline (illumination-invariant) as a highly sparse scene descriptor. To increase the information value of these scene descriptors, we use seqSLAM (Milford and Wyeth, 2012), a sequence-based method for localization. Even though the single spectra contain a comparably small amount of information compared to common panoramic images, the combination with seqSLAM allows robust localization despite illumination changes and tilt applied to the robot. Moreover, our method does not suffer from blur, allowing the usage on even fast-moving robots. We compared our method with vanilla seqSLAM and the feature-based method FABMAP and could outperform both on challenging tracks.

7.1.4 Holistic Visual 3D-Compass

The visual compass (Zeil et al., 2003) is a simple holistic method to rotationally align two panoramic images. For example, the panoramic image captured at the current robot location (current view) could be required to be aligned with a previously stored panoramic image (snapshot). By systematically rotating the current view and searching for the best match with the snapshot (by minimizing some image distance measure), the rotational offset can be determined. However, the visual compass is commonly limited to rotations around a single axis. We showed that a real-time visual 3D compass can be realized in the frequency domain using the basis of spherical harmonics. Since we commonly do not have access to full-spherical panoramic images in robot applications, we enhanced the visual 3D compass: On the one hand, we exploit symmetries of the spherical harmonics (hemispherical continuation) and, on the other hand, introduced weighting functions for panoramic images with arbitrary opening angles. By performing extensive tests of the visual 3D compass on various databases, we searched for optimal parameters and preprocessing steps to increase the accuracy of the visual 3D compass. Moreover, we examined the influence of illumination changes and translational offsets between the current view and snapshot. As our results show, the optimal choice of parameters and preprocessing techniques depends on the desired task. Finally, we compared the visual 3D compass with feature-based methods and could show that for current views and snapshots captured at the same location the visual 3D compass achieves similar performance while requiring only a fraction of the computational power.

7.1.5 3D-Warping

A common task in visual navigation is to determine a vector pointing from the current robots location (current view) towards a previously visited goal location (snapshot) using panoramic images only. This vector is referred to as home vector and can be used for homing or localization of the robot. Several approaches have been suggested to determine the home vector, including holistic methods (which use the complete panoramic image for pixel-wise comparisons) and feature-based methods (which extract and match visual features in both images to estimate the camera motion). As could be shown by Fleer and Möller (2017), the holistic method min-warping and its predecessor 2D-warping (Möller et al., 2010) reliably estimate home vectors for robots limited to planar

movement. The accuracy achieved by min-warping is similar to the computationally expensive feature-based methods and also robust against illumination changes. However, the limitation to planar movement restricts the applicability of min-warping for outdoor navigation. To overcome this limitation, we suggested 3D-warping as generalization of 2D-warping for arbitrary movements. As our experiments show, 3D-warping performs commonly worse than 2D- and min-warping for planar movement and on standard RGB images. However, using skyline segmented images, the accuracy of 3D-warping increases. As soon as the robot is tilted, 3D-warping outperforms 2D- and min-warping. Finally, we could show that — by priorly applying our visual 3D compass — our 3D-warping method is robust against tilt up to 45°.

7.2 Discussion

Throughout this work we introduced a wide variety of tools for outdoor navigation. This includes the extraction of the skyline as visual feature (chapter 2), a theoretical framework to represent functions defined on the unit sphere (chapter 3), a localization method based on the amplitude spectrum of the skyline (chapter 4), a method to rotationally align panoramic images (chapter 5), and a generalization of 2D-warping for arbitrary movement (chapter 6). While each topic was discussed individually in their corresponding chapters, we aim in this section to discuss overall implications.

7.2.1 Alternative Approaches for Skyline Extraction

In chapter 2, we examined and compared several methods to classify each pixel in an image either as ground object or sky (the resulting binary image is called *skyline*). As our results show, the skyline provides valuable information for visual navigation in outdoor environments: In chapter 4 we proposed a localization method based on the amplitude spectrum of the skyline as rotation- and illumination-invariant scene descriptor. Furthermore, we showed in chapter 6 that the skyline can be used to compute home vectors towards a previously visited goal location. Both findings underline the importance of the skyline for outdoor navigation.

Even though the skyline is — at least theoretically — completely illumination-invariant, it discards many important information, i.e. brightness values. Therefore the follow-up question "What information are important for visual outdoor navigation?" arises. Only the sky region of the image — not to be confused with the skyline — does not provide visual landmarks which can be used for visual navigation. Even worse, the appearance of the sky region changes strongly with varying weather and illumination conditions. In contrast, ground objects as for example houses or trees provide valuable information for visual navigation. Therefore the skyline could also be used to mask out sky regions from images (sky removal) to decrease the influence of weather and illumination in outdoor environments (Pepperell et al., 2014).

However, using an additional UV-only camera for skyline segmentation increases the hardware requirements drastically, especially if further cameras need to be calibrated with the UV-only camera. For industrial use, e.g. in lawn-mower robots, these additional hardware requirements are untenable. One possibility could be to develop specialized cameras which are able to see both UV-only and visible light. This could be realized using the same approach as for color cameras using a Bayer-pattern as filter matrix. However, the development of such hardware would require massive resources.

Using common color cameras, the skyline could also be extracted using machine learning on the complete image. The additional spatial information (the complete image can be used as input) could reveal additional information for skyline segmentation. As shown by Badrinarayanan et al. (2015), a convolutional deep neural network can be trained to classify different image regions, for example the sky, trees, houses, and street signs in images. A specialized neural network could be trained to only perform a classification between ground objects and sky, however the training would require a large set of labeled training data as ground truth. Moreover, the ground truth would need to be labeled manually or require supervision. Recalling our results from chapter 2,

using a calibrated pair of color and a UV-only camera could be used to simultaneously collect training data for the neural network and the corresponding ground truth without supervision.

7.2.2 Biological Plausibility

The skyline segmentation from chapter 2 uses a color contrast between UV- and green light and is inspired by anatomical insect studies. As for all methods suggested in this work, we focus on designing methods with a low computational complexity for two reasons: First, it allows us to run these methods on low-cost hardware in real-time, which is a crucial factor for autonomous navigation of robots with limited computational power. Second, if our methods show reliable performance for navigational tasks in outdoor environments, these methods can also be used to propose biological models of insect navigation. We discussed the biological plausibility of skyline segmentation based on color contrast in chapter 2, however this has been neglected for our localization method (chapter 4), visual 3D compass (chapter 5), and 3D-warping (chapter 6). As shown by Möller (2012) for the homing method min-warping, a diligent reformulation of a method can be used to propose a model feasible for insect navigation. A reformulation of the methods suggested in this work could also contribute to the study of insect navigation. A possible approach to use the visual 3D compass for insect navigation is discussed as future work in section 7.3.1.

7.2.3 How Low Can You Go?

Regarding the input image quality used for visual localization (section 1.2.1), Milford (2013) poses the question 'How low can you go?'. The author captured omnidirectional images while driving with a car in both rural and urban environments. As it turns out, using a sequence-based method — similar to the localization algorithm presented in chapter 4 — visual localization can be performed using input images with a low resolution and reduced depth (bits per pixel). By using a high number of images per sequence, the reduced information in the images (downscaled panoramic images with a resolution of 8×4 and 2 bit depth) can be compensated for; it is shown that a sequence length of 50 images could still successfully be used for visual localization. In both cases, around 80% of all locations were matched correctly. This coincides with our findings throughout this work: We mostly used low frequencies (commonly around $L = 20$ bands) in our tests performed for visual localization (chapter 4), the visual 3D compass (chapter 5), and 3D-warping (chapter 6). This emphasizes one of the strengths of holistic methods in contrast to feature-based methods: Using the complete image information instead of point features, only low requirements regarding the image quality are necessary. Initial tests of our proposed methods show that for a lower number of bands the performance only decreases slightly. However, the performance of all tested methods decreased noticeably as soon as $L = 10$ bands or less are used. In future works, more extensive tests could be performed to find lower and upper band limits for our suggested methods in varying outdoor environments.

7.2.4 A Special Case: Movement in the Plane

In chapter 3 we introduced the spherical harmonics (SH) as basis for functions defined on the unit sphere and used them in subsequent chapters for various navigational tasks. Under perfect conditions (using a correctly calibrated and skywards facing fish-eye camera), panoramic images have the form of a disk ("donut image") and are symmetric around the center pixel: Each pixel in the image corresponds to a viewing direction, where the distance and angle of a pixel from the center determine the altitude and azimuth angles of its viewing direction. Analogously to the spherical harmonics, the **Zernike polynomials** form a basis for functions defined on the unit disk (Zernike, 1934). The Zernike polynomials reveal the same symmetry around the center and are often used in optics, for example to describe optical aberrations in circular pupils (Mahajan, 1994).

In contrast to outdoor navigation, for indoor environments it can be assumed that the robot is limited to planar movement. In this case, the theory presented in the chapters 3 to 6 — which

is based on SH — could be redone using Zernike polynomials. Especially the amplitude spectrum in the basis of Zernike polynomials could provide valuable information for visual localization. Analogously to SH, Zernike polynomials behave well under rotations (only around the skywards facing Z-axis, tilt around the X-/Y-axes is not possible) and could possibly be used to enhance the homing method introduced by Stürzl and Mallot (2006) from one- to two-dimensional images.

7.3 Future Work

In this section we present future working directions for extending the methods proposed throughout this work. Generally, the methods proposed in this work as well as their implementation could be polished. For example, our C++ library could be sped up by using specially designed SIMD implementations or multithreading. Moreover, we are interested in increasing the illumination-invariance and — wherever it is feasible — the rotation- and translation-invariance of our methods. In addition, our implementation of the visual 3D compass includes a wide range of functionality which was not used in this work, e.g. alternative edge-filtering algorithms or the possibility to adaptively adjust the maximal number of bands during the compass search. These enhancements could be used for further experiments to increase the accuracy of our proposed methods.

Note that most experiments in this work were conducted using image databases. Even though these image databases were recorded in outdoor environments, they cannot mimic all problems which are encountered in real-word applications (e.g. motion blur). Therefore additional practical tests of all proposed methods, for example in robotic applications, is mandatory.

7.3.1 Multi-Snapshot Model

In chapter 6, we used a snapshot captured at a goal location to determine a home vector pointing from the current location towards the goal location. An alternative approach uses multiple snapshots captured in the vicinity of the goal location, where each snapshot is facing towards the goal location (Graham et al., 2010, Narendra et al., 2013). To determine the home vector, the current view is rotationally aligned witch each snapshot and the best matching snapshot (regarding some image distance measure) is chosen. Since the orientation between the goal location and each snapshot is known, a home vector can be determined. Due to its simplicity, the multi-snapshot model is a highly convincing model of insect navigation (section 7.2.2).

To the best knowledge of the author, the multi-snapshot model was only tested for agents limited to planar movement, e.g. as model for visual navigation of ants. The visual 3D compass proposed in chapter 5 can be used to generalize the multi-snapshot model for freely moving agents as for example bees and wasps. In initial tests we used the multi-snapshot model to perform visual navigation in a 3D simulation of a meadow partially surrounded by trees (the panoramic image database *meadow* was rendered using this simulation, see appendix D). We used the complete database and performed homing for a freely moving agent (e.g. a quadcopter or bee) towards a specified goal location. The results are promising and further research seems reasonable.

7.3.2 Robot Experiments: Proof of Concept

We developed a localization method (chapter 4), a visual 3D compass (chapter 5), and a 3D homing method (chapter 6) which could be used for visual navigation of an autonomously driving robot. At this point, all tests were performed on image databases. In future works we aim to test our methods on mobile robots in real-world applications.

For this purpose, we built a small but fast and agile wheeled robot (figure 7.1) based on the chassis of the remote control car used in section 4.2. We replaced the original motor controller with a newly designed motor controller which can directly be operated by an on-board computer (*Raspberry Pi 3* by *Raspberry Pi Foundation*). We mounted a skywards facing camera (*UI-3241LE-M-GL* by *IDS*) on top of the chassis which can be equipped with two different versions of 220° fish-eye lenses: The first lens (*BF16M220DC* by *Lensagon*) has an infrared cut-off filter and is used to capture images in the visible light. The second lens (*BF16M220D* by *Lensagon*) has no infrared cut-off filter, but is equipped with a UV broadband filter (*UG11 IRB* by *ITOS*

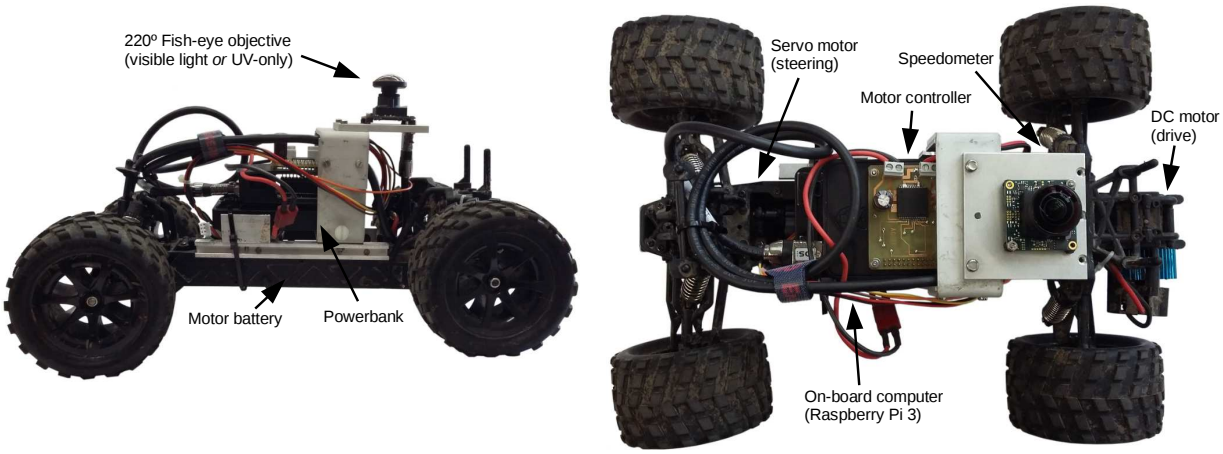


Figure 7.1: We modified the remote control car which was used in chapter 4 such that it can additionally be controlled by an on-board computer. Our low-cost outdoor robot has the advantage that it is lightweight, fast, portable, and can be used in bumpy terrain. Disadvantages are the inaccurate sensory feedback for both the steering drive motor. The camera can be equipped with a 220° fish-eye objective which is either sensible to visible light or to UV-only.

GmbH) which blocks all light with a wavelength of 390nm or higher (including IR light). The camera and the on-board computer are both powered by a powerbank (*Mobile Power Pack* by *APC*) with 20.000 mAh; both the steering and the drive motor are powered by the remote control car's battery. The maximum speed of the robot is approximately 30 km/h.

The robot has only three sensor inputs: First, the current angle of the servo motor can be read out by the on-board computer and is used passively in a software controller to set the desired steering. Second, the drive motor can be read read out by an encoder mounted on one of the back wheels (speedometer). Third, the camera image of the skywards facing camera can be used for visual navigation. The read-outs of the servo and drive motor are very imprecise and increase the challenges in performing autonomous navigation.

Currently, the algorithms developed in this work are implemented on the Raspberry Pi 3 and used to calculate timings of our algorithms. As a proof of concept, we plan to combine and enhance our algorithms to perform route following using low-cost hardware only.

7.3.3 Robot Experiments: Lawn-Mowing

The outdoor robot presented in section 7.3.2 is deliberately chosen to examine if complex navigational tasks can be solved using cheap hardware. Alternatively, more sophisticated robot platforms can be used; our group is currently modifying a designated outdoor robot platform (*VolksBot* by *Fraunhofer Institut*). It is planned that the finished robot platform will be equipped with accurate motor encoders, an inertial measurement unit, multiple cameras (including UV and color cameras), and a powerful on-board laptop. This robot platform will allow us to extensively test our methods in real-world applications. Moreover we plan to combine our expertise and framework for domestic cleaning robots with the results obtained in this work to design an efficient lawn-mower robot. The existing frameworks include methods for homing using holistic and feature-based methods (Fleer and Möller, 2017), trajectory planning (Gerstmayr-Hillen et al., 2013), and robot navigation using particle clouds (Möller et al., 2013).

7.4 Conclusion

The first central topic of this thesis suggests the skyline — an image where each pixel is either classified as ground object or sky — as an illumination- and rotation-invariant scene descriptor for visual navigation. We showed how the skyline — inspired by the study of the eyes of desert ants — can be extracted using contrast color vision between the UV and green channel. Moreover, we showed that by using a camera sensitive to UV-only nearly the same performance can be reached,

increasing the relevance of the skyline for the navigation of outdoor robots. The second central topic of this thesis is the basis of real spherical harmonics, a powerful tool to represent functions defined on the sphere directly in frequency domain. The mathematical properties of real spherical harmonics allowed us to develop elegant and highly efficient methods for real-time applications; even on low-cost hardware.

We could show that both the skyline and our suggested methods based on spherical harmonics can be combined to introduce visual navigation methods suitable for outdoor navigation: We suggested methods to perform visual localization, rotationally align panoramic images, and visual homing using the skyline as input. Even though all methods work with common panoramic images, the skyline showed superior performance for varying illumination conditions and for strong tilt of the robot; underlining the importance of the skyline for visual navigation. We hope that our results and findings will contribute to the field of visual outdoor navigation.

APPENDIX A

Proofs

A.1 Calculation Rules for Direct Sums and Kronecker Products

For the proofs given in the sections A.3 and A.7 we frequently use common calculation rules for direct sums (definition 3.3) and Kronecker products (definition 3.5):

$$[\mathbf{A} \oplus \mathbf{B}]^\dagger = \mathbf{A}^\dagger \oplus \mathbf{B}^\dagger \tag{A.1}$$

$$[\mathbf{A} \otimes \mathbf{B}]^T = \mathbf{A}^T \otimes \mathbf{B}^T \tag{A.2}$$

$$[\mathbf{AC}] \otimes [\mathbf{BD}] = [\mathbf{A} \otimes \mathbf{B}][\mathbf{C} \otimes \mathbf{D}] \tag{A.3}$$

Note that there are various other basic rules for the calculation of direct sums and Kronecker products, however these are not used during our proofs.

A.2 Clebsch-Gordan Matrix Ordering

Proof of lemma 3.6. As stated in Marinucci and Peccati (2011), remark 3.40, the Clebsch-Gordan coefficients $c^{l,m}_{l_1,m_1,l_2,m_2}$ which fulfill the triangle conditions (3.25) and (3.26) can implicitly be ordered by the relation (row inside the Clebsch-Gordan matrix)

$$(m_1, m_2) < (m_1', m_2') :\Leftrightarrow (m_1 < m_1') \ \vee \ (m_1 = m_1' \ \wedge \ m_2 < m_2') \tag{A.4}$$

and by the relation (column inside the Clebsch-Gordan matrix)

$$(l, m) < (l', m') :\Leftrightarrow (l < l') \ \vee \ (l = l' \ \wedge \ m < m'). \tag{A.5}$$

For example, a Clebsch-Gordan coefficient $c^{l,m}_{l_1,m_1,l_2,m_2}$ with $l = 2$ and $m = 0$ has a lower column index than a Clebsch-Gordan coefficient with $l = 3$ and $m = 0$. These relations do not explicitly state the column and row index of each Clebsch-Gordan coefficient. In the following we derive explicit formulas for this purpose.

By inspecting the matrix shown in example 3.3, it can be seen that (due to the ordering implied by equation A.4) only the indices m_1 and m_2 change in each row. Moreover, by substituting the indices $\tilde{m}_1 = m_1 + l_1$ and $\tilde{m}_2 = m_2 + l_2$ we can rewrite the first column as

$$\begin{pmatrix} c^{0,0}_{1,-1,1,-1} & \cdots \\ c^{0,0}_{1,-1,1,0} & \cdots \\ c^{0,0}_{1,-1,1,1} & \cdots \\ c^{0,0}_{1,0,1,-1} & \cdots \\ c^{0,0}_{1,0,1,0} & \cdots \\ c^{0,0}_{1,0,1,1} & \cdots \\ c^{0,0}_{1,1,1,-1} & \cdots \\ c^{0,0}_{1,1,1,0} & \cdots \\ c^{0,0}_{1,1,1,1} & \cdots \end{pmatrix} \underset{\text{Substitute indices}}{\Longrightarrow} \begin{pmatrix} c^{0,0}_{1,0,1,0} & \cdots \\ c^{0,0}_{1,0,1,1} & \cdots \\ c^{0,0}_{1,0,1,2} & \cdots \\ c^{0,0}_{1,1,1,0} & \cdots \\ c^{0,0}_{1,1,1,1} & \cdots \\ c^{0,0}_{1,1,1,2} & \cdots \\ c^{0,0}_{1,2,1,0} & \cdots \\ c^{0,0}_{1,2,1,1} & \cdots \\ c^{0,0}_{1,2,1,2} & \cdots \end{pmatrix}. \tag{A.6}$$

Now it can be seen that in each row the index $\tilde{m}_2$ is increased by one for $2l_2+1$ rows. After that, $\tilde{m}_2$ is set to zero and the index $\tilde{m}_1$ is increased by one. Applying the same strategy to the general case, we obtain

$$\begin{aligned} \text{row}(c^{l,m}_{l_1,m_1,l_2,m_2}) &= \tilde{m}_1(2l_2+1)+\tilde{m}_2+1 && \text{(A.7)}\\ &= (l_1+m_1)(2l_2+1)+l_2+m_2+1 && \text{(A.8)} \end{aligned}$$

where we additionally add one since the indices of matrices start by one instead of zero.

By inspecting the matrix shown in example 3.3, it can be seen that (due to the ordering implied by equation A.5) only the indices l and m change in each column. Since there are $2l+1$ tuples (l,m) with $m \in \{-l,\dots,l\}$, there is a total of $l^2 = \sum_{i=0}^{l} 2l+1$ tuples (l,m) for the first l bands. The number of tuples which appear before the first tuple $(l,m) = (l_2-l_1, -l_2+l_1)$ is $(l_2-l_1)^2$ and has to be subtracted from the total number of l^2 tuples. As before, we substitute $\tilde{m} = m+l$ and see that $\tilde{m}$ is increased by one for $2l+1$ rows and afterwards reset to zero. Together, we therefore obtain

$$\begin{aligned} \text{col}(c^{l,m}_{l_1,m_1,l_2,m_2}) &= l^2-(l_2-l_1)^2+\tilde{m}+1 && \text{(A.9)}\\ &= l^2-(l_2-l_1)^2+l+m+1 && \text{(A.10)} \end{aligned}$$

which finishes the proof.

□

A.3 Real Point-Wise Product

Proof of theorem 3.19. We start by calculating the point-wise product in the basis of SH using theorem 3.14:

$$\vec{A}^{l_1}_1 \cdot \vec{A}^{l_2}_2 \overset{\text{Th. } 3.14}{=} \tilde{\mathbf{C}}^{\dagger}_{l_1,l_2}\left[\vec{A}^{l_1}_1 \otimes \vec{A}^{l_2}_2\right] \tag{A.11}$$

Now we can apply the basis transformation from equation (3.58) and then reshape the equation using calculation rules for Kronecker products

$$\begin{aligned} &\overset{(3.58)}{=} \tilde{\mathbf{C}}^{\dagger}_{l_1,l_2}\left[\left[\mathbf{T}^T_{l_1}\vec{a}^{l_1}_1\right] \otimes \left[\mathbf{T}^T_{l_2}\vec{a}^{l_2}_2\right]\right] && \text{(A.12)}\\ &\underset{(A.3)}{\overset{(A.2)}{=}} \tilde{\mathbf{C}}^{\dagger}_{l_1,l_2}\left[\mathbf{T}_{l_1}\otimes\mathbf{T}_{l_2}\right]^T\left[\vec{a}^{l_1}_1 \otimes \vec{a}^{l_2}_2\right]. && \text{(A.13)} \end{aligned}$$

Recalling the definition of the point-wise product (section 3.5.5), we have that the Fourier coefficient vector of the result has entries for the bands $|l_2-l_1|$ to l_2+l_1. Therefore we can express the result of $\vec{A}^{l_1}_1 \cdot \vec{A}^{l_2}_2$ in the basis of RSH by applying the appropriate transformation:

$$\vec{a}^{l_1}_1 \cdot \vec{a}^{l_2}_2 = \overline{\left[\bigoplus_{|l_2-l_1|}^{l_2+l_1} \mathbf{T}_l\right]}\left[\vec{A}^{l_1}_1 \cdot \vec{A}^{l_2}_2\right] \overset{(A.13)}{=} \underbrace{\overline{\left[\bigoplus_{|l_2-l_1|}^{l_2+l_1} \mathbf{T}_l\right]}\tilde{\mathbf{C}}^{\dagger}_{l_1,l_2}\left[\mathbf{T}_{l_1}\otimes\mathbf{T}_{l_2}\right]^T}_{\overset{\text{Def. } 3.18}{=}\tilde{\mathbf{c}}^{\dagger}_{l_1,l_2}}\left[\vec{a}^{l_1}_1 \otimes \vec{a}^{l_2}_2\right] \tag{A.14}$$

By substituting the definition of the real Clebsch-Gordan matrices (definition 3.18) we finally obtain

$$\vec{a}^{l_1}_1 \cdot \vec{a}^{l_2}_2 \overset{\text{Def. } 3.18}{=} \tilde{\mathbf{c}}^{\dagger}_{l_1,l_2}\left[\vec{a}^{l_1}_1 \otimes \vec{a}^{2}_2 l_2\right] \tag{A.15}$$

which finishes the proof. □

A.4 Symmetry Theorem

Proof of theorem 3.22. To prove the theorem, we have to show which RSH are invariant under the operations κ_R, κ_M, and κ_N as well as their concatenations (e.g. $\kappa_R \circ \kappa_M$). Since these operations are self-inverse, e.g. $\kappa_R = \kappa_R^{-1}$, and commutative, e.g. $\kappa_R \circ \kappa_M = \kappa_M \circ \kappa_R$, we only have to check seven different combinations of these operations. Note that y_0^0 is constant and therefore invariant under κ_M and κ_R but not under κ_N. Thus, we will in the following only consider the case $l \geq 0$.

(i) Invariants of κ_N:
From $\kappa(y_m^l) = -y_m^l$ we have that y_m^l is invariant under κ_M if and only if $y_m^l(\vartheta, \varphi) = 0$. Therefore no RSH y_m^l is invariant under κ_N.

(ii) Invariants of κ_M and κ_{MN}:
First, recall the definition of associated Legendre polynomials from equation (3.13)

$$P_m^l(\cos\vartheta) = \frac{(-1)^m}{2^l l!}(\sin^2\vartheta)^m \frac{d^{l+m}}{d(\cos\vartheta)^{l+m}}(\cos^2\vartheta - 1)^l. \tag{A.16}$$

From equation (A.16) we have

$$P_m^l(\cos(\pi - \vartheta)) = (-1)^{l+m} P_m^l(\cos\vartheta) \tag{A.17}$$

and therefore our symmetry operation κ_M applied to y_m^l is:

$$\kappa_M(y_m^l(\vartheta, \varphi)) \overset{(3.72)}{=} y_m^l(\pi - \vartheta, \varphi) \underset{(A.17)}{\overset{(3.55)}{=}} (-1)^{l+m} y_m^l(\vartheta, \varphi) \tag{A.18}$$

As a consequence, we have that y_m^l is invariant under κ_M if and only if $l + m$ is even and that y_m^l is invariant under κ_{MN} if and only if $l + m$ is odd.

(iii) Invariants of κ_R and κ_{RN}:
For $m > 0$ it holds that

$$\kappa_R(y_m^l(\vartheta, \varphi)) \overset{(3.55)}{=} \sqrt{2} K_m^l \cos(m(\varphi + \pi)) P_m^l(\cos\vartheta) = (-1)^m y_m^l(\vartheta, \varphi). \tag{A.19}$$

Therefore we have that y_m^l is invariant under κ_R if and only if m is even and under κ_{RN} if and only if m is odd. For arbitrary m the proof is done analogously.

(iv) Invariants of κ_{RM} and κ_{RMN}:
By combining (ii) and (iii), we have

$$\kappa_{RM}(y_m^l(\vartheta, \varphi)) = (-1)^{l+m}(-1)^m y_m^l(\vartheta, \varphi) = (-1)^l y_m^l(\vartheta, \varphi). \tag{A.20}$$

from which we directly obtain that y_m^l is invariant under κ_{RM} if l is even and under κ_{RMN} if l is odd.

□

A.5 Rotation Theorems

In this section we present the proofs for the rotation theorems 3.23, 3.24, 3.25, and 3.26. Prior to that, we prove lemma 3.11, which gives some general information about Wigner-D matrices. This proof is to some extent redundant — which might be the reason why we could not find it in other literature — since the SH are actually *designed* to fulfill these properties. For a better understanding of SH, we nevertheless decided to explicitly prove these properties.

Proof of lemma 3.11. To prove (ii), we use the definition of the Wigner-D matrices (definition 3.4) and obtain

$$\begin{aligned} D^l_{mn}(\mathbf{R}) &\overset{(3.12)}{=} e^{-im\gamma}\lambda^l_{mn}(\beta)e^{-in\alpha} = \overline{e^{im\gamma}\lambda^l_{mn}(\beta)e^{in\alpha}} \\ &\overset{(3.13)}{=} \overline{e^{-in(-\alpha)}\lambda^l_{nm}(-\beta)e^{-im(-\gamma)}} \overset{(3.12)}{=} \bar{D}^l_{nm}(\mathbf{R}^T) \end{aligned} \tag{A.21}$$

which gives us $\left[\mathbf{D}^l(\mathbf{R})\right]^\dagger = \mathbf{D}^l(\mathbf{R}^T)$. Recalling definition 3.17 we have that the real Wigner-D matrices only differ by a basis transformation such that we also have for real Wigner-D matrices $\left[\mathbf{d}^l(\mathbf{R})\right]^\dagger = \mathbf{d}^l(\mathbf{R}^T)$.

Now we can use (ii) to prove (i): In section 3.7 we showed that rotations in the basis of SH can be expressed via equation (3.42) as $\mathbf{R} \circ Y^l_m(\vec{p}) = \sum_{k=-l}^{l} \bar{D}^l_{mk}(\mathbf{R}) Y^l_k(\vec{p})$. Therefore we have

$$\begin{aligned} \left\langle \mathbf{R} \circ Y^l_m(\vec{p}), Y^l_n(\vec{p}) \right\rangle &\overset{(3.42)}{=} \left\langle \sum_{k=-l}^{l} \bar{D}^l_{mk}(\mathbf{R}) Y^l_k(\vec{p}), Y^l_n(\vec{p}) \right\rangle = \sum_{k=-l}^{l} \bar{D}^l_{mk}(\mathbf{R}) \left\langle Y^l_k(\vec{p}), Y^l_n(\vec{p}) \right\rangle \\ &\overset{(3.34)}{=} \bar{D}^l_{mn}(\mathbf{R}), \end{aligned} \tag{A.22}$$

where we use in the last step that SH are orthonormal. Analogously, we obtain

$$\begin{aligned} \left\langle Y^l_m(\vec{p}), \mathbf{R}^T \circ Y^l_n(\vec{p}) \right\rangle &\overset{(3.42)}{=} \left\langle Y^l_m(\vec{p}), \sum_{k=-l}^{l} \bar{D}^l_{nk}(\mathbf{R^T}) Y^l_k(\vec{p}) \right\rangle = \sum_{k=-l}^{l} D^l_{nk}(\mathbf{R^T}) \left\langle Y^l_m(\vec{p}), Y^l_k(\vec{p}) \right\rangle \\ &\overset{(3.34)}{=} D^l_{nm}(\mathbf{R}^T) \end{aligned} \tag{A.23}$$

From (ii) we have that $\bar{D}^l_{mn}(\mathbf{R}) = D^l_{nm}(\mathbf{R}^T)$ which finishes the proof. The same proof also holds for the case of RSH.

□

A.5.1 Z-Axis Rotations

Proof of theorem 3.23. While this theorem and its proof are well-known (e.g. Green (2003) p. 23), we show it for the sake of completeness. We make a case distinction for the different possible values of m:
For $m > 0$ we have

$$\begin{aligned} y^l_m(\mathbf{R}\vec{p}) &= y^l_m(\vartheta, \varphi + \alpha) = \sqrt{2} K^l_m \cos(m(\varphi + \alpha)) P^l_m(\cos\vartheta) \\ &= \sqrt{2} K^l_m (\cos(m\varphi)\cos(m\alpha) - \sin(m\varphi)\sin(m\alpha)) P^l_m(\cos\vartheta) \\ &= \cos(m\alpha) y^l_m(\vartheta, \varphi) + \sin(m\alpha) y^l_{-m}(\vartheta, \varphi). \end{aligned} \tag{A.24}$$

For $m < 0$ we have

$$\begin{aligned} y^l_m(\mathbf{R}\vec{p}) &= y^l_m(\vartheta, \varphi + \alpha) = \sqrt{2} K^l_m \sin(-m(\varphi + \alpha)) P^l_{-m}(\cos\vartheta) \\ &= \sqrt{2} K^l_m (-\sin(m\varphi)\cos(m\alpha) - \cos(m\varphi)\sin(m\alpha)) P^l_{-m}(\cos\vartheta) \\ &= \cos(m\alpha) y^l_m(\vartheta, \varphi) - \sin(m\alpha) y^l_{-m}(\vartheta, \varphi). \end{aligned} \tag{A.25}$$

For $m = 0$ the rotation has no effect on y^l_m. □

A.5.2 Y-Axis Rotations

In this section we use the terms **axial symmetric** ($y_m^l \in I_M$) and **point symmetric** ($y_m^l \in I_{MN}$) based on the symmetries from section 3.6.2. Note that each RSH is either point or axial symmetric and that this distinction only depends on the parity of $l+m$ (theorem 3.22). In order to prove theorem 3.24, we first have to introduce the following lemma.

Lemma A.1. *Let $\mathbf{R} := \mathbf{R}_{Y,\alpha} \in SO(3)$ be a rotation matrix around the Y-axis by angle α, then it holds*

$$\mathbf{R}(\vartheta,\varphi) = (\vartheta',\varphi') \Rightarrow \mathbf{R}^T(\pi-\vartheta,\varphi) = (\pi-\vartheta',\varphi'). \tag{A.26}$$

If furthermore $f \in L^2(S^2)$ is an axial symmetric ($y_m^l \in I_M$) or point symmetric ($y_m^l \in I_{MN}$) function, we have

$$f(\mathbf{R}(\vartheta,\varphi)) = \quad f(\mathbf{R}^T(\pi-\vartheta,\varphi)) \qquad (\textit{axial symmetric}) \tag{A.27}$$
$$f(\mathbf{R}(\vartheta,\varphi)) = -f(\mathbf{R}^T(\pi-\vartheta,\varphi)) \qquad (\textit{point symmetric}). \tag{A.28}$$

Proof of lemma A.1. For the first part denote by

$$\mathbf{R} := \begin{pmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{pmatrix} \tag{A.29}$$

the entries of the rotation matrix. By mapping the spherical coordinates to Cartesian coordinates via equation (3.30), applying $\mathbf{R}$, and mapping back to spherical coordinates via equation (3.29), we obtain that $\mathbf{R}(\vartheta,\varphi) = (\vartheta',\varphi')$ and $\mathbf{R}^T(\pi-\vartheta,\varphi) = (\vartheta'',\varphi'')$ with

$$\vartheta' = \operatorname{acos}(\sin\vartheta\cos\varphi\sin\alpha + \cos\vartheta\cos\alpha) \tag{A.30}$$
$$\vartheta'' = \pi - \operatorname{acos}(\sin\vartheta\cos\varphi\sin\alpha + \cos\vartheta\cos\alpha) \tag{A.31}$$
$$\varphi' = \operatorname{atan}\left(\frac{\sin\vartheta\sin\varphi}{\sin\vartheta\cos\varphi\cos\alpha - \cos\vartheta\sin\alpha}\right) = \varphi''. \tag{A.32}$$

Therefore we have $\vartheta'' = \pi - \vartheta'$ and $\varphi'' = \varphi'$ which proves the first part. The second part follows for axial symmetric functions from

$$f(\mathbf{R}(\vartheta,\varphi)) = f(\vartheta',\varphi') \overset{\kappa_M(f)=f}{=} f(\pi-\vartheta',\varphi') = f(\mathbf{R}^T(\pi-\vartheta,\varphi)), \tag{A.33}$$

where we directly apply the result from the first part. The point symmetric case is proven analogously. □

Using lemma A.1, we are now able to prove theorem 3.24.

Proof of theorem 3.24. Without loss of generality we first assume that y_m^l and y_n^l are both axial symmetric, that is $\kappa_M(y_m^l) = y_m^l$ and $\kappa_M(y_n^l) = y_n^l$. Then we have

$$y_m^l(\vartheta,\varphi) = y_m^l(\pi-\vartheta,\varphi) \tag{s1}$$
$$y_n^l(\vartheta,\varphi) = y_n^l(\pi-\vartheta,\varphi) \tag{s2}$$

and can derive

$$d^l_{mn} = \left\langle \mathbf{R} \circ y^l_m, y^l_n \right\rangle \overset{\text{le.3.11}(i)}{=} \left\langle y^l_m, \mathbf{R}^T \circ y^l_n \right\rangle \tag{A.34}$$

$$= \int_0^{2\pi}\int_0^{\pi} y^l_m(\vartheta,\varphi) y^l_n(\mathbf{R}^T(\vartheta,\varphi)) \sin\vartheta d\vartheta d\varphi \tag{A.35}$$

$$\underset{(s2)}{\overset{\text{le.A.1}}{=}} \int_0^{2\pi}\int_0^{\pi} y^l_m(\vartheta,\varphi) y^l_n(\mathbf{R}(\pi-\vartheta,\varphi)) \sin\vartheta d\vartheta d\varphi \tag{A.36}$$

$$\overset{(s1)}{=} \int_0^{2\pi}\int_0^{\pi} y^l_m(\pi-\vartheta,\varphi) y^l_n(\mathbf{R}(\pi-\vartheta,\varphi)) \sin\vartheta d\vartheta d\varphi \tag{A.37}$$

Now we integrate by substituting $f(\vartheta) = \pi - \vartheta$ and finally obtain

$$= \int_0^{2\pi}\int_0^{\pi} y^l_m(\vartheta,\varphi) y^l_n(\mathbf{R}(\vartheta,\varphi)) \sin\vartheta d\vartheta d\varphi = \left\langle y^l_m, \mathbf{R} \circ y^l_n \right\rangle = d^l_{nm}. \tag{A.38}$$

If y^l_m or y^l_n are point symmetric, the result is multiplied by (-1) in both cases; the corresponding equations are marked with (s1) and (s2). Therefore we have

$$d^l_{mn} = d^l_{nm} \Leftrightarrow m+n \text{ even} \quad \text{and} \quad d^l_{mn} = -d^l_{nm} \Leftrightarrow m+n \text{ odd} \tag{A.39}$$

We prove the second part by induction over the band l. For $l = 0$ and $l = 1$ the assumption holds true. This can be verified by inspecting the matrix $\mathbf{d}^1$ from equation (3.71) with $\alpha = \gamma = 0$:

$$\mathbf{d}^1(0,\beta,0) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{pmatrix} \tag{A.40}$$

For the induction step we use the relations stated in theorem 3.21 and make case distinctions depending on m and n:

(i) $m = 0$	(ii) $m = 1$	(iii) $m > 1$
(I) $0 > n > -l$	(II) $n = -l$	

The approach is in all six possible combinations (small vs. capital greek numbers) the same: We use the induction hypothesis, the zero entries in the matrix $\mathbf{d}^1$, or that w^l_{mn} or v^l_{mn} is zero. For each term always one of these conditions applies, as example the expression is explicitly evaluated for the combination (i) and (I):

$$d^l_{mn} = u^l_{0,n}U^l_{0,n} + v^l_{0,n}V^l_{0,n} + \underbrace{w^l_{0,n}}_{=0} W^l_{0,n} \tag{A.41}$$

$$= u^l_{0,n0}P^l_{0,n} + v^l_{0,n}({}_1P^l_{1,n} + {}_{-1}P^l_{-1,n}) \tag{A.42}$$

$$= u^l_{0,n}d^1_{0,0} \underbrace{d^{l-1}_{0,n}}_{=0 \text{ ind.hyp.}} + v^l_{0,n}(d^1_{1,0} \underbrace{d^{l-1}_{1,n}}_{=0 \text{ ind.hyp.}} + \underbrace{d^1_{-1,0}}_{=0} d^{l-1}_{-1,n}) = 0 \tag{A.43}$$

This shows that $d^l_{mn} = 0$ and from lemma 3.11 we also have that $d^l_{nm} = 0$. □

A.5.3 X-Axis Rotations

Proof of theorem 3.25. A rotation around the X-axis can be expressed in terms of a ZYZ Euler rotation as

$$\mathbf{R}_{X,\alpha} = \mathbf{R}_{Z,-\frac{\pi}{2}}\mathbf{R}_{Y,\alpha}\mathbf{R}_{Z,\frac{\pi}{2}}. \tag{A.44}$$

In the following we denote by $\mathbf{a}^l := \mathbf{d}^l\left(\mathbf{R}_{Z,-\frac{\pi}{2}}\right)$ and $\mathbf{b}^l := \mathbf{d}^l(\mathbf{R}_{Y,\alpha})$ the real Wigner-D matrices corresponding to the rotation matrices $\mathbf{R}_{Z,-\frac{\pi}{2}}$, $\mathbf{R}_{Y,\alpha}$ and $\mathbf{R}_{Z,\frac{\pi}{2}}$. From theorem 3.23 we have

$$a^l_{m,n} = \begin{cases} 1, & \text{if } n = m \ \wedge \ m \equiv 0 \bmod 4; \\ -1, & \text{if } n = m \ \wedge \ m \equiv 2 \bmod 4; \\ 1, & \text{if } n = -m \ \wedge \ m \equiv 1 \bmod 4; \\ -1, & \text{if } n = -m \ \wedge \ m \equiv 3 \bmod 4; \\ 0, & \text{else} \end{cases} \tag{A.45}$$

and $a^l_{m,m} = a^l_{-m,-m}$, and $a^l_{m,-m} = -a^l_{-m,m}$. By concatenating the rotations

$$\mathbf{d}^l := \mathbf{d}^l(\mathbf{R}_{\mathbf{X},\alpha}) = \mathbf{d}^l\left(\mathbf{R}_{Z,-\frac{\pi}{2}}\right)\mathbf{d}^l(\mathbf{R}_{Y,\alpha})\mathbf{d}^l\left(\mathbf{R}_{Z,\frac{\pi}{2}}\right) = \mathbf{a}^l\mathbf{b}^l{\mathbf{a}^l}^T$$

we have

$$d^l_{m,n} = \sum_{q=-l}^{l}\left(\sum_{p=-l}^{l} a^l_{m,p}b^l_{p,q}\right)a^l_{n,q} = \sum_{\substack{p\,\in\,\{m,-m\}\\ q\,\in\,\{n,-n\}}} a^l_{m,p}b^l_{p,q}a^l_{n,q} \tag{A.46}$$

$$= a^l_{m,m}b^l_{m,n}a^l_{n,n} + a^l_{m,-m}b^l_{-m,n}a^l_{n,n} + a^l_{m,m}b^l_{m,-n}a^l_{n,-n} + a^l_{m,-m}b^l_{-m,-n}a^l_{n,-n}. \tag{A.47}$$

Now let $m+n$ be even, then we have from equation (A.45) that either $a^l_{m,-m} = 0$ or $a^l_{n,n} = 0$ (the same yields for $a^l_{n,-n}$ and $a^l_{m,m}$). As a consequence the two terms in the middle of equation (A.47) vanish. Furthermore, we have from theorem 3.24 that $b^l_{m,n} = b^l_{n,m}$, thus by reordering we obtain

$$d^l_{m,n} = a^l_{m,m}b^l_{m,n}a^l_{n,n} + a^l_{m,-m}b^l_{-m,-n}a^l_{n,-n} = a^l_{n,n}b^l_{n,m}a^l_{m,m} + a^l_{n,-n}b^l_{-n,-m}a^l_{m,-m} = d^l_{n,m} \tag{A.48}$$

which proves the first assumption. The same argument works for the case $m+n$ is odd.

To prove the second statement, it is sufficient to insert each combination from the table shown in theorem A.5.3 to equation (A.47) and check that each single term of the sum is equal to zero. For example, we explicitly prove the case $m<0$, $n<0$, and $m+n$ odd:

$$d^l_{mn} = a^l_{m,m}b^l_{m,n}a^l_{n,n} + a^l_{m,-m}\underbrace{b^l_{-m,n}}_{=0 \text{ th.3.24}}a^l_{n,n} + a^l_{m,m}\underbrace{b^l_{m,-n}}_{=0 \text{ th.3.24}}a^l_{n,-n} + a^l_{m,-m}b^l_{-m,-n}a^l_{n,-n} \tag{A.49}$$

$$= \underbrace{a^l_{m,m}a^l_{n,n}}_{=0,\ (A.45)}b^l_{m,n} + \underbrace{a^l_{m,-m}a^l_{n,-n}}_{=0,\ (A.45)}b^l_{-m,-n} = 0 \tag{A.50}$$

Here we use again in equation (A.50) that, due to $m+n$ odd, either $a^l_{m,m}$ or $a^l_{n,n}$ is zero (the same yields for $a^l_{m,-m}$ and $a^l_{n,-n}$). □

A.5.4 Rotations of $\pm 90°$

For rotations around the Z-axis the proof of theorem 3.26 is trivially given by substituting $\alpha = \pm 90°$ into theorem 3.23. For X-axis and Y-axis rotations, it is likely that the proof can be done similar to the proofs of the theorems 3.23, 3.24, and 3.25. However, the higher number of case distinctions makes it a poor approach. Alternatively, we prove the theorem by taking advantage of symmetries, partially using similar arguments as in the earlier proofs of this section. We only need to introduce the following lemma:

Lemma A.2. *Let f be a real-valued function defined on the unit sphere with Fourier coefficient vector $\vec{a}^L$. Moreover, let f be invariant under κ_M. Then we have for $l > 0$*

$$\kappa_{MN}(y_m^l) = y_m^l \;\Rightarrow\; a_m^l = 0. \tag{A.51}$$

Note that the role of κ_M and κ_{MN} can be exchanged.

Proof. From theorem 3.22 we have that the RSH can be divided — except for y_0^0 — into two disjoint sets, namely I_M ($l+m$ is even) and I_{MN} ($l+m$ is odd). Since the first band $l=0$ is not affected by rotations at all, we only consider bands $l \geq 1$. We prove the theorem for the case that f is invariant under κ_M, the case that f is invariant under κ_{MN} is proven analogously. The function f can be written as a linear combination:

$$f = \sum_{l=1}^{L} \sum_{m=-l}^{l} a_m^l y_m^l = \sum_{l=1}^{L} \left(\sum_{\substack{m=-l \\ l+m \text{ even}}}^{l} a_m^l y_m^l + \sum_{\substack{m=-l \\ l+m \text{ odd}}}^{l} a_m^l y_m^l \right) \tag{A.52}$$

Applying κ_M to f we have:

$$\kappa_M(f) = \sum_{l=1}^{L} \left(\sum_{\substack{m=-l \\ l+m \text{ even}}}^{l} a_m^l \kappa_M(y_m^l) + \sum_{\substack{m=-l \\ l+m \text{ odd}}}^{l} a_m^l \kappa_M(y_m^l) \right) = \sum_{l=1}^{L} \left(\sum_{\substack{m=-l \\ l+m \text{ even}}}^{l} a_m^l y_m^l - \sum_{\substack{m=-l \\ l+m \text{ odd}}}^{l} a_m^l y_m^l \right) \tag{A.53}$$

From the assumption that f is invariant under κ_M, i.e. $f = \kappa_M(f)$, we therefore have

$$f = \sum_{l=1}^{L} \left(\sum_{\substack{m=-l \\ l+m \text{ even}}}^{l} a_m^l y_m^l + \sum_{\substack{m=-l \\ l+m \text{ odd}}}^{l} a_m^l y_m^l \right) = \sum_{l=1}^{L} \left(\sum_{\substack{m=-l \\ l+m \text{ even}}}^{l} a_m^l y_m^l - \sum_{\substack{m=-l \\ l+m \text{ odd}}}^{l} a_m^l y_m^l \right) = \kappa_M(f) \tag{A.54}$$

and it directly follows that

$$\sum_{l=1}^{L} \sum_{\substack{m=-l \\ l+m \text{ odd}}}^{l} a_m^l y_m^l = 0. \tag{A.55}$$

Since the RSH form a basis, all coefficients a_m^l with $l+m$ odd have to be zero. From theorem 3.22 we have that their corresponding real spherical harmonics y_m^l are invariant under κ_{MN}, which finishes the proof. □

The theorem can now be proven for X-axis rotations as follows: First, we define a new symmetry κ_W and examine for each RSH y_m^l if it is either an element of I_W or I_{WN}. Then we can show that the rotated Fourier coefficient vector $\mathbf{R} \circ y_m^l$ is either an element of I_M or I_{MN}. From lemma A.2 we have that many entries of the Fourier coefficient vector $\mathbf{R} \circ y_m^l$ are zero. By using theorem 3.22, we can finally identify the entries d_{mn}^l which are zero.

Proof of theorem 3.26. The single steps of the proof for X-axis rotations are in detail:

(i) Define a new operation $\kappa_W(y_m^l(\vartheta, \varphi)) := y_m^l(\vartheta, -\varphi)$ for RSH. This operation mirrors a point on the hyperplane spanned by the X-axis and Z-axis, i.e. $(x, y, z)^T \mapsto (x, -y, z)^T$ (figure 3.4).

(ii) Then we can show alike in the proof of theorem 3.22 that the RSH can be divided into two classes depending on the index m:

$$m \geq 0 \;\Rightarrow\; y_m^l \in I_W \quad \text{and} \quad m < 0 \;\Rightarrow\; y_m^l \in I_{WN}. \tag{A.56}$$

(iii) Analogously to the proof of lemma A.1, we show that

$$y_m^l \in I_W \quad \Rightarrow \quad \mathbf{R} \circ y_m^l \in I_M \tag{A.57}$$
$$y_m^l \in I_{WN} \quad \Rightarrow \quad \mathbf{R} \circ y_m^l \in I_{MN}. \tag{A.58}$$

This relation can be seen in figure 3.4, where a rotation around the X-axis by 90° transfers the operation κ_W into κ_M.

(iv) Since the columns of a transformation matrix hold the images of the basis vectors, we have that the n-th column of $\mathbf{d}^l$ contains the linear coefficients which fulfill $\mathbf{R} \circ y_n^l = \sum_{m=-l}^{l} d_{mn}^l y_m^l$. From lemma A.2 we have that a function $f \in L^2(S^2)$, which satisfies $\kappa_M(f) = f$, can be written as a linear combination of RSH which also satisfy $\kappa_M(y_m^l) = y_m^l$.

(v) Now we use theorem 3.22 to derive the desired sparsity relations. This can be done analogously for all cases; here we show it explicitly for the case $n \geq 0$ and l is even: We have from (ii) that $y_n^l \in I_W$ and from (iii) that $\mathbf{R} \circ y_n^l \in I_M$. From (iv) it follows that each entry of the sum $\mathbf{R} \circ y_n^l = \sum_{m=-l}^{l} d_{mn}^l y_m^l$ has to be invariant under κ_M. From theorem 3.22 we have that only y_m^l with m even are invariant under κ_M. As a consequence, all coefficients d_{mn}^l in the linear combination $\mathbf{R} \circ y_n^l = \sum_{m=-l}^{l} d_{mn}^l y_m^l$ with m odd have to be zero.

For Y-axis rotations we define the symmetry $\kappa_V(y_m^l(\vartheta, \varphi) := y_m^l(\vartheta, \pi - \varphi)$. This symmetry mirrors a point on the hyperplane spanned by the Y-axis and Z-axis, i.e. $(x, y, z)^T \mapsto (-x, y, z)^T$ and can be used to split the RSH into the two sets I_V and I_{VN}. The remaining proof can then be done analogously. □

A.6 Translations

Proof of theorem 3.30. We will use equation (3.93) and express the translation in the opposite direction as a product:

$$\mathbf{T}' = \mathbf{R}_{Y,-\pi} \mathbf{T} \mathbf{R}_{Y,\pi} \tag{A.59}$$

In order to calculate the product, we need to calculate $\mathbf{R}_{Y,\pi}$. Note that the rotation of a single point in spherical coordinates is given by $\mathbf{R}_{Y,\pi} \circ (\vartheta, \varphi) = (\pi - \vartheta, \pi - \varphi)$. Therefore we have that for $m > 0$ the translation of a single RSH y_m^l is given by

$$\mathbf{R}_{Y,\pi} \circ y_m^l(\vartheta, \varphi) = y_m^l(\pi - \vartheta, \pi - \varphi) = \sqrt{2} K_m^l \cos(m(\pi - \varphi)) P_m^l(\cos(\pi - \vartheta)) \tag{A.60}$$
$$= \sqrt{2} K_m^l (-1)^m \cos(\varphi) (-1)^{l+m} P_m^l(\cos\vartheta) = (-1)^l y_m^l \tag{A.61}$$

Analogously, we obtain for the remaining cases ($m = 0, m < 0$):

$$\mathbf{R}_{Y,\pi} \circ y_m^l(\vartheta, \varphi) = \begin{cases} (-1)^l \quad y_m^l, & \text{for } m \geq 0 \\ (-1)^{l+1} y_m^l, & \text{for } m < 0 \end{cases} \tag{A.62}$$

From this equation, we directly obtain that the corresponding Wigner-D matrix is diagonal and each entry is either $+1$ or -1. Now we will calculate the entries of the matrix $\mathbf{T}'$. Note that we will use the enhanced notation for arbitrary transformation matrices from equation (3.91) to denote the entries of the Wigner-D matrices $\mathbf{D}$ corresponding to the rotation matrices $\mathbf{R}_{Y,\pi} = \mathbf{R}_{Y,-\pi}$. Furthermore, we will repeatedly use that $D_{m_1,m_2}^{l_1,l_2} = 0$ for $l_1 \neq l_2$ (section 3.7) and $T_{m_1,m_2}^{l_1,l_2} = 0$ for $m_1 \neq m_2$ (theorem 3.29). The entries of the product of the first two matrices $\hat{\mathbf{T}} := \mathbf{T}\mathbf{R}_{Y,\pi}$ is element-wise given by

$$\hat{T}_{m_1,m_2}^{l_1,l_2} = \sum_{l=0}^{L-1} \sum_{m=-l}^{l} T_{m_1,m}^{l_1,l} D_{m,m_2}^{l,l_2} = T_{m_1,m_1}^{l_1,l_2} D_{m_1,m_2}^{l_2,l_2}. \tag{A.63}$$

The matrix $\mathbf{T}'$ is therefore given by

$$T'^{l_1,l_2}_{m_1,m_2} = \sum_{l=0}^{L-1} \sum_{m=-l}^{l} D^{l_1,l}_{m_1,m} \hat{T}^{l,l_2}_{m,m_2} = \sum_{l=0}^{L-1} \sum_{m=-l}^{l} D^{l_1,l}_{m_1,m} T^{l,l_2}_{m,m} D^{l_2,l_2}_{m,m_2} \tag{A.64}$$

$$= D^{l_1,l_1}_{m,m} T^{l_1,l_2}_{m,m} D^{l_2,l_2}_{m,m}. \tag{A.65}$$

From equation (A.62) we have that $D^{l_1,l_1}_{m,m}$ and $D^{l_2,l_2}_{m,m}$ are either 1 or -1. By checking the different cases for m ($m \geq 0$ or $m < 0$) and l_1, l_2 (even or odd), we can finally derive the function σ^{l_1,l_2}_m from the theorem. □

A.7 Bispectrum for Real Spherical Harmonics

Proof of theorem 3.34. We start by substituting our definition of the real point-wise product (theorem 3.19) and the real coupling matrix (equation (3.62)) to obtain

$$\vec{a}_i^\dagger \left[\vec{a}_1^{l_1} \cdot \vec{a}_2^{l_2}\right] \overset{\text{Th. 3.19}}{=} \vec{a}_i^\dagger \tilde{\mathbf{c}}^\dagger_{l_1,l_2} \left[\vec{a}_1^{l_1} \otimes \vec{a}_2^{l_2}\right] \tag{A.66}$$

$$\overset{(3.62)}{=} \vec{a}_i^\dagger \left[\overline{[\mathbf{T}_{l_1} \otimes \mathbf{T}_{l_2}]} \tilde{\mathbf{C}}_{l_1,l_2} \left[\bigoplus_{j=l_2-l_1}^{l_2+l_1} \mathbf{T}_j\right]^T\right]^\dagger \left[\vec{a}_1^{l_1} \otimes \vec{a}_2^{l_2}\right] \tag{A.67}$$

Now we can reshape the equation using standard rules for direct sums, Kronecker products, etc.:

$$= \vec{a}_i^\dagger \overline{\left[\bigoplus_{j=l_2-l_1}^{l_2+l_1} \mathbf{T}_j\right]} \tilde{\mathbf{C}}^\dagger_{l_1,l_2} \left[\mathbf{T}_{l_1} \otimes \mathbf{T}_{l_2}\right]^T \left[\vec{a}_1^{l_1} \otimes \vec{a}_2^{l_2}\right] \tag{A.68}$$

$$\overset{(A.1)}{\underset{(A.2)}{=}} \left[\left[\bigoplus_{j=l_2-l_1}^{l_2+l_1} \mathbf{T}_j\right]^T \vec{a}_i\right]^\dagger \tilde{\mathbf{C}}^\dagger_{l_1,l_2} \left[\mathbf{T}^T_{l_1} \otimes \mathbf{T}^T_{l_2}\right] \left[\vec{a}_1^{l_1} \otimes \vec{a}_2^{l_2}\right] \tag{A.69}$$

$$\overset{(A.3)}{=} \left[\left[\bigoplus_{j=l_2-l_1}^{l_2+l_1} \mathbf{T}_j\right]^T \vec{a}_i\right]^\dagger \tilde{\mathbf{C}}^\dagger_{l_1,l_2} \left[\left[\mathbf{T}^T_{l_1} \vec{a}_1^{l_1}\right] \otimes \left[\mathbf{T}^T_{l_2} \vec{a}_2^{l_2}\right]\right] \tag{A.70}$$

This allows us to apply the transformation from the basis of RSH to SH (equation (3.58)) to finally obtain

$$\overset{(3.58)}{=} \vec{A}_i^\dagger \tilde{\mathbf{C}}^\dagger_{l_1,l_2} \left[\vec{A}_1^{l_1} \otimes \vec{A}_2^{l_2}\right] = \vec{A}_i^\dagger \left[\vec{A}_1^{l_1} \cdot \vec{A}_2^{l_2}\right] \tag{A.71}$$

Therefore we have $\mathcal{BS}_f(l_1, l_2, i) = \vec{a}_i^\dagger \left[\vec{a}_1^{l_1} \cdot \vec{a}_2^{l_2}\right] = \vec{A}_i^\dagger \left[\vec{A}_1^{l_1} \cdot \vec{A}_2^{l_2}\right]$ what finishes the proof. □

APPENDIX B

UVG: Details

B.1 HDR Algorithm Modifications

The HDR algorithm in this work is based on the one suggested by Debevec and Malik (1998) and slightly modified. The original algorithm states that the optical density D_i, which is measured by the camera, is mapped by a strong monotonically increasing function f to a brightness value Z_{ij}, where i is the index of the corresponding pixel on the camera sensor and j the index of the current exposure time $\triangle t_j$. If we assume that the optical density

$$D_i = E_i \triangle t_j \tag{B.1}$$

is a product of the irradiance E_i and the exposure time $\triangle t_j$, then the mapping has to fulfill

$$Z_{ij} = f(E_i \triangle t_j). \tag{B.2}$$

Let $Z = \{0, 1, \ldots, 255\}$ be the set of brightness values which can be produced by the camera, then the mapping f is discrete and we can rewrite the terms of equation (B.2) to obtain

$$g(Z_{ij}) := \log f^{-1}(Z_{ij}) = \log E_i + \log \triangle t_j. \tag{B.3}$$

The function g is called the camera responsive curve. Since camera sensors are prone to errors for (nearly) under- and overexposed pixels, an additional weighting function

$$w(z) = \begin{cases} z, & \text{if } z \leq 127 \\ 255 - z, & \text{else} \end{cases}, \quad z \in Z \tag{B.4}$$

can be considered to reformulate equation (B.3) into the following least-square optimization problem

$$\begin{aligned} \mathcal{O} = &\sum_i \sum_j w(Z_{ij})(g(Z_{ij}) - \log E_i - \log \triangle t_j)^2 \\ &+ \lambda \sum_z w(z) g''(z), \end{aligned} \tag{B.5}$$

where λ is a coefficient which can be adjusted to smooth the function g. Since the data are discrete, we use the Taylor approximation $g''(z) = g(z-1) - 2g(z) + g(z+1)$. By rewriting $\mathcal{O}$ as matrix equation, a QR factorization is applied to solve for g. After the function g has been computed once, a weighted estimation of the irradiance E_i can be calculated as

$$\log E_i = \frac{\sum_j w(Z_{ij})(g(Z_{ij}) - \log \triangle t_j)}{\sum_j w(Z_{ij})}. \tag{B.6}$$

In our implementation, several changes are made: First, the irradiance values $\log E_i$ are discretized to the values $\log e_i \in \{0, 1, \ldots, 255\}$. To achieve this, the minimal and maximal values

of $\log E_i$ must be known, which is the case as soon as all images for the database are recorded. Since our databases cover complete days and a wide range of different illumination conditions, we can assume that (at least in our temperate zone) these limits are valid for newly collected data. Second, the estimate $\log E_i$ cannot be calculated if the corresponding pixel values Z_{ij} are under- and overexposed in all images. As mentioned by Aguerrebere et al. (2014), the sensor noise will in this case dominate the estimation of $\log e_i$. As a consequence we decided to set the final irradiance estimation $\log e_i$ as

$$\log e_i = \begin{cases} 0, & \text{if } \overline{Z_{ij}} < \alpha \\ 255, & \text{if } \overline{Z_{ij}} > 255 - \alpha \\ \log E_i, & \text{else,} \end{cases} \tag{B.7}$$

where $\overline{Z_{ij}}$ is the mean of the values Z_{ij} for varying exposure times j and α a threshold value. The value α (in our experiments $\alpha = 1.5$) is used to avoid problems induced by sensor noise, especially for underexposed pixels. Finally we modified the optimization problem (B.5) by the following

$$\begin{aligned} \mathcal{O}' = &\sum_i \sum_j w(Z_{ij})(g(Z_{ij}) - \log E_i - \log \triangle t_j)^2 \\ &+ \lambda \sum_z w(z)^\mu g''(z) \end{aligned} \tag{B.8}$$

to improve the smoothening of the function g. In the following $g_{\lambda,\mu}$ denotes the result of optimizing equation (B.8) for the specified parameters. Note that (B.8) with $\mu = 1$ is equivalent to the original equation (B.5).
If λ is small in the original equation (B.5), the resulting irradiance images of a scene may suffer from color banding (example B.2). This effect may occur for mappings between two color spaces if the resolution of these color spaces differs or the mapping is not 'smooth'. While this is barely visible to the naked eye, this effect can be observed in the histogram of the corresponding picture (example B.2). Choosing a large smoothening coefficient $\lambda \gg 0$ prevents this effect, however in this case the extreme regions around 0 and 255 of g will not match well with the raw data anymore (e.g. visible for the range around 255 in example B.1). In our modified version of the optimization problem (B.8), the additional parameter μ allows to smooth the function strongly in the center while the extreme regions are preserved (in our experiments we use $g_{1.5,2}$, see figure B.1 (b)).

B.2 Efficiency on Generalized Data Sets

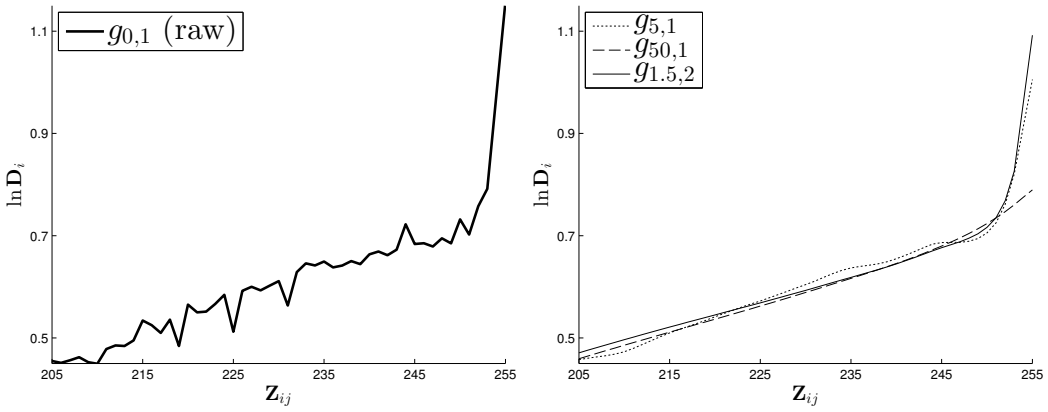
In section 2.2.8 and 2.2.9 it is described how global separation methods are trained on the data sets X_{8-19} and X_{7-20}. However, the same data sets are used to evaluate the performance of these separators. Therefore we investigate in this section if trained separators generalize well to new data sets. Recall that the sample X^i contains data from the forest/suburban database collected at day i. Now we define $\hat{X}^i$ as the sample containing data from all seven recorded days, except the i-th. Then the classification rates $R_{\hat{X}^i}(X^j)$ (section 2.2.7) for a separator trained on the dataset $\hat{X}^i$ (6 days) and tested on day X^j can be calculated. This allow us to compare the efficiency of the separators for the case that a separator is tested and trained on different data sets ($i = j$) or on the same data set ($i \neq j$). Now it is of interest if the difference of the classification rates of these two samples may be caused by random fluctuations, only in this case we can expect that this separator generalizes well to new data. Denote by

$$T_1 = \{R_{\hat{X}^i}(X^i) \mid i = 1, \ldots, 7\} \tag{B.9}$$

the set containing all classification rates from global separation methods which were trained on six days and tested on the remaining seventh day. Contrary the set

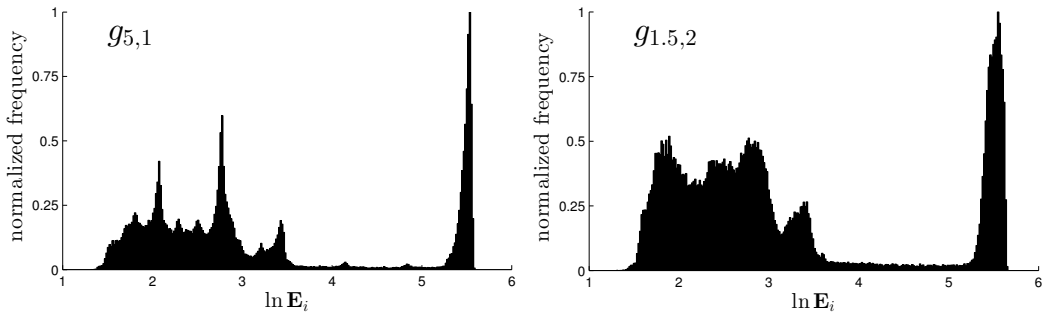
$$T_2 = \{R_{\hat{X}^i}(X^j) \mid i, j = 1, \ldots, 7, i \neq j\} \tag{B.10}$$

Example B.1: Modified Optimization Problem



The left plot shows the raw data of the function g for the region around $Z_{ij} = 255$ for an example image from the database. To find a smooth approximation of the raw data, different values for the smoothing terms of $g_{\lambda,\mu}$ are tested; the results are shown in the right plot. While $g_{5,1}$ is rippled which results in strong color banding, $g_{50,1}$ is smooth but suffers from discrepancies at the extreme values. In contrast, the function $g_{1.5,2}$ computed by our approach is smooth and fits the raw data well.

Example B.2: Smoothed Histogram



Example histograms of an image from the collected database (01. Sept. 2014, 8:00, UV-only) after applying the HDR algorithm with two different camera responsive curves $g_{5,1}$ and $g_{1.5,2}$. Using the original smoothing term (left) with $g_{5,1}$, color banding effects (strong peaks) are visible. Contrary, the histogram of the same scene with the modified smoothness term (here: $g_{1.5,2}$) is smoother and shows less color banding effects.

contains all classification rates from global separation methods which were trained on six days and also tested on one of these days.

In general it is not possible to check the alternative hypothesis H_1 that 'T_1 and T_2 are samples from the same distribution'. Therefore we apply standard statistical tests with the alternative hypothesis H_1 : 'T_1 and T_2 are samples from different distributions', to check if the samples differ significantly[1]. If this is the case, the corresponding p-values would be small, e.g. $p \leq 0.05 = \alpha$ for a 95% level of significance (type I error). However, large p-values indicate that the difference

[1] Actually we are interested in the β error (or type II error) to verify that both samples could be derived from the same distribution. Unfortunately, without further knowledge of the real underlying distribution, this error cannot be calculated.

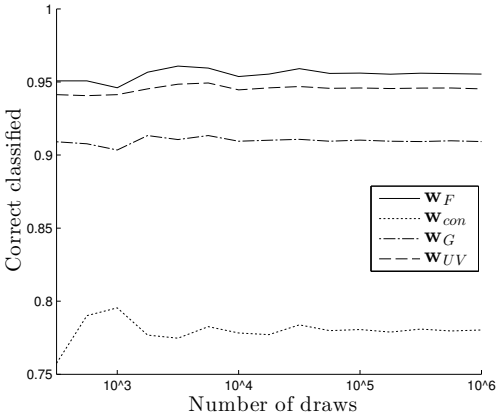


Figure B.1: The classification rate for the sample X_{8-19} depends on the number of draws n performed to obtain the sample. As can be seen, the classification rate is stable for values $n > 10^4$.

between the samples is small, such that the possibility to accept the null hypothesis H_0 : 'T_1 and T_2 are samples from the same distributions' increases. Even though the direct correlation between α and β cannot be stated (and thus p-values can be misleading), large p-values indicate that the β error is small.

We apply two different statistical tests, the *Wilcoxon signed-rank test* (WSR) and the independent two-sample *t-test*. Both tests calculate the p-value for the specified problem stated above, however the t-test presumes that T_1 and T_2 are normally distributed, while the WSR test can be applied to arbitrary distributions. In order to verify that T_1 and T_2 are normally distributed, a goodness-of-fit test needs to be applied, however the sample sizes are too small ($\#T_1 = 7, \#T_2 = 42$) for this kind of tests. Based on the experience that error values are often normally distributed, we still applied both tests for a better comparison. The results are shown in table 2.5. As can be seen, the WSR and t-test produce similar results.

Global separation techniques are trained on the union of all images over several days. This training set may differ strongly for different days as can be seen in figure 2.7. Contrary, local separation techniques are trained on single images which only differ slightly in comparison: Even for images captured from different scenes and under different illuminations, the sky and ground classes are distinguishable, only their absolute positions inside the logarithmic UV/G plot differ. However, contrary to the global separation techniques, the local separation techniques do not depend on the absolute positions since the threshold value is estimated for each single image. As a result, the set of selected training days only has a small influence on the training of the local separation techniques.

B.3 Numeric Stability

The samples X used to evaluate the different separation techniques are simulated by a probability experiment with n draws (section 2.2.5). To ensure that the amount of draws does not influence the calculated classification rates, the classification rates for X_{8-19} were calculated for varying numbers of draws. As shown in figure B.1, the results are stable for $n > 10^4$. To offer a safe guarantee regarding the classification rates, we therefore decided to set the number of draws to $n = 10^5$.

APPENDIX C

Spherical Harmonics & Applications

C.1 Code: Computation of Clebsch-Gordan Matrices

In section 3.5.2 we introduced the theory necessary to calculate Clebsch-Gordan matrices $\mathbf{C}_{l_1,l_2}$. In this section we show a Matlab (MATLAB, 2012) example code to calculate these matrices. Note that we use Matlab as programming language — instead of C++ as used for the implementation in our library libSHC (section 3.10) — to keep the code simple and lucid.

```
function [row, col] = cg_indices(l, m, l1, m1, l2, m2)

  row = (l1+m1)*(2*l2+1)+l2+m2+1;
  col = l^2-(l2-l1)^2+l+m+1;

end
```

The function *cg_indices()* calculates the column and row index of a Clebsch-Gordan coefficient $c^{l,m}_{l_1,m_1,l_2,m_2}$ by applying lemma 3.6.

```
function [rows, cols, cg] = cg_coefficients(l, m, l1, l2)

  % Calculate all CG-coefficients with m=m1+m2;
  % adapted from Straub (2014).
  mm = (m-l1-l2+abs(l1-l2+m))/2;
  n  = (m+l1+l2-abs(l1-l2-m))/2-mm+1;
  B = zeros(2*n,1); C = zeros(n+1,1);

  count = 0;
  C(n)  = 1;
  for x = n-1:-1:1
    B(2*x)   = l1*(l1+1)+l2*(l2+1)+2*(mm+x)*(m-mm-x)-l*(l+1);
    B(2*x-1) =  sqrt((l1*(l1+1)-(mm+x)*(mm+x-1))*(l2*(l2+1) ...
                -(m-mm-x)*(m-mm-x+1)));
    C(x)     = -(B(2*x)*C(x+1)+B(2*x+1)*C(x+2))/B(2*x-1);
    count    = count + C(x).^2;
  end

  C(n) = sqrt(1/(count+1));
  for x = n-1:-1:1
    C(x)  = -(B(2*x)*C(x+1)+B(2*x+1)*C(x+2))/B(2*x-1);
    count = count + C(x).^2;
  end

  % Store the calculated CG-coefficients in a vector.
  % The entries are ordered by ascending values of m1.
  cg = C(1:n);

  % Find all coefficients with m=m1+m2 ...
  rows = zeros(n,1); cols = zeros(n,1);
  count = 1;
```

```
32    for m1=-l1:l1
33      if abs(m-m1) <= l2
34        % ... and calculate their row and column indices.
35        [rows(count), cols(count)] = cg_indices(l,m,l1,m1,l2,m-m1);
36        count = count+1;
37      end
38    end
39
40  end
```

The main part of the function *cg_coefficients()* (lines 3-27) is an adaption of the code presented in Straub (2014). The function calculates all non-zero Clebsch-Gordan coefficients $c^{l,m}_{l_1,m_1,l_2,m_2}$ for fixed values l, m, l_1, l_2 and varying values m_1, m_2 which fulfill both the triangle condition (equation (3.25)) and $m = m_1 + m_2$ (equation (3.26)). The result is stored in the vector $\vec{cg}$ (line 27), whereby the entries of $\vec{cg}$ are ordered by ascending values of m_1. Afterwards, we iterate through all possible non-zero Clebsch-Gordan coefficients $c^{l,m}_{l_1,m_1,l_2,m_2}$ (lines 30-38) by checking the triangle condition and $m = m_1 + m_2$. For each non-zero entry, we calculate the row and column indices (line 35) using the previously defined function *cg_indices()*. Finally, the function returns the list of non-zero Clebsch-Gordan coefficients $c^{l,m}_{l_1,m_1,l_2,m_2}$ together with their row and column indices.

```
1   function CG = cg_matrix(l1, l2)
2
3     % Initialize CG-matrix with zero entries.
4     n  = (2*l1+1) * (2*l2+1);
5     CG = zeros(n,n);
6
7     % Fill CG-matrix band-wise ...
8     for l=abs(l2-l1):l2+l1
9       % ... and for an increasing value m ...
10      for m=-l:l
11        % ... with CG-coefficients.
12        [rows, cols, cg] = cg_coefficients(l,m,l1,l2);
13        for i=1:length(cg)
14          CG(rows(i),cols(i)) = cg(i);
15        end
16      end
17    end
18
19  end
```

Finally, we can construct the Clebsch-Gordan matrix $\mathbf{C}_{l_1,l_2}$ by calculating the Clebsch-Gordan indices and their corresponding row and column indices using the function *cg_coefficients()*. By iterating through all values $l \in \{|l_2 - l_1|, \ldots, l_2 + l_1\}$ we obtain all non-zero Clebsch-Gordan coefficients of $\mathbf{C}_{l_1,l_2}$. In more detail, the matrix is constructed column-wise which can be seen from the ordering defined in the proof of lemma 3.6.

C.2 Detailed Results: Visual 3D Compass

In chapter 5, we introduce the visual 3D compass and examine its performance. An exemplary C++ implementation of the visual 3D compass using our library libSHC is shown in figure 5.1. The results for the same-database experiments (section 5.5.1) are presented in table C.1. The results for the cross-database experiments (section 5.5.2) are presented in the tables C.2 to C.5. The best results are highlighted in each table (red text color). Finally, the results for the translation experiments (section 5.5.3) are presented in the figures C.1-C.5. The abbreviations used for the various parameter sets of the visual 3D compass used in the figures are described in detail in section 5.3.

		Noise: Natural										Noise: Constant											
		TD: Off					TD: On					TD: Off					TD: On						
		Preprocessing					Preprocessing					Preprocessing					Preprocessing						
		hs	hS	Hs	HS	HDR	hs	hS	Hs	HS	HDR	hs	hS	Hs	HS	HDR	hs	hS	Hs	HS	HDR		
fill		32.6% 7.8°	38.5% 6.4°	47.4% 12.9°	44.0% 8.4°	54.5% 14.7°	28.0% 8.0°	32.9% 5.9°	43.6% 13.7°	38.9% 7.6°	51.7% 14.6°	28.0% 7.7°	27.3% 5.3°	34.9% 11.8°	32.0% 6.8°	49.2% 14.7°	24.6% 7.4°	22.3% 4.9°	33.6% 11.5°	28.1% 6.0°	48.5% 13.8°		
hemi												25.5% 8.0°	32.7% 6.0°	32.0% 12.1°	33.0% 6.1°	45.3% 16.4°	27.8% 7.9°	38.3% 7.1°	35.0% 12.5°	37.8% 7.1°	47.7% 16.7°	**180°**	**Search Space: Large**
weighted		18.6% 4.7°	17.6% 4.2°	23.3% 7.5°	21.5% 5.4°	38.8% 8.6°	13.5% 5.8°	11.4% 4.9°	20.0% 8.6°	16.4% 5.9°	36.8% 9.6°	15.8% 4.2°	15.0% 3.6°	16.8% 6.6°	18.4% 4.8°	35.1% 8.1°	11.4% 4.9°	9.5% 4.1°	14.0% 6.6°	14.1% 5.1°	34.5% 8.2°		
fill		25.0% 4.7°	29.6% 3.9°	36.4% 9.4°	38.4% 6.8°	51.0% 11.4°	20.1% 4.9°	22.8% 3.8°	32.3% 9.5°	32.1% 5.7°	48.6% 11.5°	20.2% 4.4°	20.5% 3.3°	23.4% 7.4°	23.7% 4.9°	43.8% 10.5°	16.1% 4.2°	15.5% 3.0°	20.6% 7.3°	19.5% 4.4°	42.1% 9.8°		
weighted		10.9% 2.1°	11.0% 2.5°	9.5% 3.5°	12.1% 3.1°	33.8% 5.4°	5.6% 2.5°	6.3% 2.9°	6.0% 3.9°	7.6% 3.3°	31.8% 6.1°	9.4% 1.9°	10.2% 2.3°	6.4% 3.1°	10.7% 2.9°	31.7% 5.5°	4.9% 2.1°	5.8% 2.6°	3.7% 3.1°	6.6% 3.0°	31.1% 5.4°	**220°**	
complete												8.4% 1.3°	12.6% 1.7°	2.1% 1.1°	8.3% 1.7°	30.7% 4.6°	3.2% 0.9°	7.5% 1.2°	0.9% 0.9°	5.3% 1.3°	29.9% 4.0°	**360°**	
fill		9.5% 5.9°	9.4% 4.1°	18.3% 10.1°	12.9% 5.4°	9.3% 16.7°	8.0% 6.6°	7.6% 4.2°	15.2% 10.5°	10.6% 5.2°	8.8% 16.9°	5.7% 5.8°	4.8% 3.8°	7.7% 9.6°	6.2% 4.9°	4.3% 16.0°	4.5% 6.2°	3.4% 3.6°	7.7% 9.7°	4.9% 4.6°	4.4% 16.2°		
hemi												4.2% 6.2°	5.8% 4.5°	4.9% 9.0°	4.9% 4.1°	3.6% 17.1°	5.2% 6.6°	8.1% 5.0°	6.2% 9.5°	7.4% 5.0°	4.8% 17.5°	**180°**	**Search Space: Small**
weighted		2.0% 3.5°	1.9% 3.1°	3.1% 5.4°	2.5% 3.9°	0.6% 12.9°	1.2% 5.0°	1.5% 4.2°	2.5% 6.5°	2.2% 5.0°	0.5% 13.7°	1.5% 3.1°	2.1% 2.9°	1.9% 4.5°	2.1% 3.4°	0.3% 12.4°	1.0% 4.1°	1.2% 3.5°	1.3% 5.3°	1.5% 4.1°	0.2% 12.8°		
fill		6.5% 3.6°	5.6% 2.9°	13.3% 7.0°	12.1% 4.3°	7.1% 14.7°	5.2% 4.2°	4.1% 2.8°	10.9% 7.2°	9.6% 3.9°	7.0% 15.0°	3.1% 3.5°	2.5% 2.5°	4.1% 6.3°	3.8% 3.5°	2.5% 14.0°	2.4% 3.8°	1.6% 2.3°	3.8% 6.4°	3.4% 3.3°	2.5% 13.8°	**220°**	
weighted		0.7% 1.6°	0.9% 2.1°	0.5% 2.5°	1.2% 2.4°	0.1% 11.2°	0.3% 2.3°	0.7% 2.6°	0.2% 3.1°	0.8% 2.8°	0.0% 11.8°	0.6% 1.5°	1.2% 1.9°	0.4% 2.1°	1.1% 2.2°	0.0% 11.1°	0.3% 1.9°	0.6% 2.3°	0.1% 2.5°	0.7% 2.5°	0.0% 11.3°		
complete												0.5% 1.0°	0.8% 1.3°	0.1% 0.8°	0.7% 1.2°	0.0% 10.6°	0.2% 0.9°	0.6% 1.0°	0.0% 0.7°	0.2% 1.0°	0.0% 10.4°	**360°**	

Table C.1: Detailed results for the **same-database** tests on the database ***mixed_**** as described in section 5.3 and 5.5.1. The best results are highlighted (red text color).

		Noise: Natural										Noise: Constant											
		TD: Off					TD: On					TD: Off					TD: On						
		Preprocessing					Preprocessing					Preprocessing					Preprocessing						
		hs	hS	Hs	HS	HDR	hs	hS	Hs	HS	HDR	hs	hS	Hs	HS	HDR	hs	hS	Hs	HS	HDR		
fill		49.8% 10.9°	66.0% 15.0°	49.7% 11.9°	59.5% 13.2°	36.1% 9.0°	45.6% 10.8°	61.0% 13.7°	46.5% 11.8°	55.8% 11.9°	31.7% 9.2°	44.5% 10.1°	54.6% 12.8°	35.7% 9.5°	46.1% 9.7°	28.9% 7.9°	40.8% 9.9°	50.0% 11.8°	32.5% 9.3°	42.2% 8.9°	25.1% 7.8°		
hemi												45.0% 12.1°	65.8% 14.7°	26.3% 10.8°	40.7% 8.2°	30.5% 13.5°	57.1% 15.0°	72.2% 17.8°	33.2% 12.1°	48.8% 10.9°	37.0% 14.8°	**180°**	**Search Space: Large**
weighted		77.4% 14.3°	61.7% 14.3°	27.1% 9.5°	32.2% 8.6°	50.8% 10.4°	75.8% 14.4°	56.6% 13.8°	22.2% 9.6°	27.4% 8.5°	47.5% 10.8°	76.6% 13.8°	58.8% 13.4°	18.8% 8.5°	26.5% 7.7°	48.4% 9.7°	75.3% 13.5°	53.8% 12.9°	14.9% 8.3°	22.1% 7.5°	44.5% 9.9°		
fill		32.8% 6.5°	56.9% 10.6°	35.6% 7.9°	56.5% 11.1°	20.7% 5.1°	27.2% 6.5°	49.7% 9.2°	31.4% 7.7°	51.9% 9.6°	16.1% 5.4°	26.3% 5.8°	43.3% 8.5°	18.4% 5.8°	39.3% 7.5°	11.6% 4.2°	21.7% 5.6°	37.7% 7.8°	15.0% 5.7°	34.8% 6.5°	8.4% 4.2°		
weighted		69.4% 12.1°	48.8% 10.5°	9.4% 5.7°	22.0% 5.9°	33.5% 7.2°	67.7% 12.0°	42.2% 10.2°	6.0% 5.6°	17.6% 5.7°	30.3% 7.5°	69.4% 11.9°	47.3% 10.0°	6.4% 5.2°	19.3% 5.4°	32.3% 6.9°	67.8% 11.5°	41.3% 9.7°	4.1% 5.0°	15.3% 5.1°	28.6% 6.9°	**220°**	
complete												14.1% 3.1°	20.7% 4.7°	1.7% 1.7°	20.9% 3.9°	2.3% 1.9°	10.2% 3.3°	16.1% 4.2°	0.8% 1.7°	15.3% 3.0°	1.3% 2.5°	**360°**	
fill		13.6% 7.5°	30.5% 9.2°	13.4% 8.0°	23.0% 8.2°	6.7% 6.6°	12.4% 8.1°	27.5% 9.2°	11.0% 8.2°	19.7% 7.4°	6.5% 7.5°	10.7% 7.1°	21.9% 9.0°	4.3% 6.8°	10.4% 7.1°	3.6% 6.1°	10.1% 7.5°	19.8% 8.7°	3.6% 6.9°	8.4% 6.7°	3.3% 6.6°		
hemi												10.7% 8.5°	28.2% 10.7°	2.5% 7.0°	7.7% 5.8°	3.9% 11.1°	21.5% 11.0°	37.7% 12.4°	4.8% 8.2°	15.0% 7.4°	7.0% 12.3°	**180°**	**Search Space: Small**
weighted		49.2% 10.9°	25.5% 10.2°	5.7% 6.8°	6.9% 6.8°	21.2% 8.2°	48.4% 11.6°	24.4% 10.6°	5.0% 7.3°	5.8% 7.0°	20.7% 9.0°	48.8% 10.7°	23.6% 9.9°	4.0% 6.1°	5.4% 6.2°	19.3% 7.8°	48.0% 11.3°	22.7% 10.0°	3.3% 6.3°	4.8% 6.3°	18.0% 8.3°		
fill		5.2% 4.2°	21.7% 6.6°	7.8% 5.2°	21.3% 6.7°	2.8% 3.7°	4.5% 4.7°	19.3% 6.5°	6.3% 5.3°	18.5% 6.0°	2.4% 4.5°	3.7% 3.9°	13.3% 6.1°	1.9% 4.2°	8.4% 5.6°	1.1% 3.4°	3.2% 4.3°	11.9% 6.0°	1.5% 4.4°	6.4% 5.1°	0.9% 3.8°		
weighted		41.0% 9.1°	16.5% 8.1°	2.9% 4.7°	4.1% 5.2°	14.8% 6.0°	40.1% 9.7°	15.5% 8.5°	2.4% 5.0°	3.3% 5.2°	14.3% 6.6°	40.9% 9.0°	15.8% 7.8°	2.4% 4.4°	3.6% 4.9°	14.2% 5.8°	40.2% 9.5°	14.6% 8.1°	2.0% 4.5°	2.8% 4.8°	12.9% 6.2°	**220°**	
complete												1.3% 2.1°	4.0% 3.0°	0.2% 1.3°	2.7% 2.7°	0.3% 1.6°	1.0% 2.7°	3.1% 2.8°	0.1% 1.4°	1.5% 2.3°	0.2% 2.4°	**360°**	

Table C.2: Detailed results for the **cross-database** tests on the database ***lab_**** as described in section 5.3 and 5.5.2. The best results are highlighted (red text color).

	Noise: Natural										Noise: Constant											
	TD: Off					TD: On					TD: Off					TD: On						
	Preprocessing					Preprocessing					Preprocessing					Preprocessing						
	hs	hS	Hs	HS	HDR	hs	hS	Hs	HS	HDR	hs	hS	Hs	HS	HDR	hs	hS	Hs	HS	HDR		
fill	48.5% 15.7°	64.3% 16.4°	66.5% 17.2°	50.0% 14.0°	58.5% 16.6°	46.0% 15.1°	61.1% 15.1°	64.5% 16.9°	45.8% 13.0°	57.3% 16.3°	43.4% 15.0°	56.7% 14.7°	56.6% 14.5°	42.8% 12.9°	54.7% 15.5°	41.7% 14.5°	52.2% 13.8°	54.9% 14.4°	39.3% 12.0°	53.0% 15.2°		
hemi											44.8% 17.0°	66.6% 18.3°	51.9% 13.5°	38.4% 12.3°	49.3% 18.4°	50.9% 18.1°	70.8% 20.4°	55.8% 15.3°	46.8% 13.7°	54.4% 19.2°	180°	Search Space: Large
weighted	53.2% 16.9°	54.3% 14.0°	54.2% 13.7°	36.2% 11.2°	60.5% 15.5°	51.8% 16.7°	50.8% 13.3°	51.8% 13.7°	31.9% 11.0°	59.6% 15.7°	51.5% 16.7°	51.9% 13.0°	50.0% 12.5°	32.6% 10.7°	59.1% 14.8°	50.8% 16.4°	48.4% 12.7°	48.7% 12.5°	28.3% 10.4°	58.6% 14.9°		
fill	38.8% 9.7°	55.2% 13.7°	57.9% 12.5°	46.9% 12.2°	51.1% 11.2°	36.7% 9.1°	50.9% 12.4°	55.0% 12.4°	41.7% 11.0°	48.9% 10.8°	33.3% 8.8°	47.7% 12.4°	45.7% 10.0°	37.3% 10.9°	45.4% 9.7°	31.6% 8.5°	43.1% 11.4°	42.7% 9.9°	32.7% 9.9°	43.8% 9.3°	220°	
weighted	45.2% 13.3°	43.7% 10.8°	43.1% 8.7°	28.1% 7.9°	54.7% 11.4°	44.7% 12.7°	39.8% 10.0°	42.0% 8.8°	22.9% 7.7°	54.0% 11.6°	44.8% 13.1°	42.8% 10.3°	41.9% 8.2°	25.6% 7.7°	54.3% 11.0°	44.0% 12.7°	38.7% 9.6°	40.5% 8.2°	20.6% 7.4°	53.8% 11.2°		
complete											25.0% 6.0°	37.0% 10.1°	14.6% 3.0°	21.8% 7.0°	31.0% 4.5°	23.7% 6.5°	31.2% 8.7°	9.0% 3.0°	16.2% 5.6°	27.1% 5.0°	360°	
fill	29.0% 11.9°	28.1% 11.0°	36.3% 12.5°	19.5% 10.2°	36.4% 12.0°	28.5% 12.1°	25.6% 10.3°	34.6% 12.7°	16.4% 9.7°	35.3% 12.5°	25.2% 11.8°	22.0% 10.6°	25.3% 11.7°	12.3% 9.7°	31.7% 11.8°	24.2% 11.9°	20.2% 10.1°	25.1% 11.8°	10.6% 9.3°	31.2% 12.1°		
hemi											23.9% 13.7°	32.1% 13.7°	21.2% 11.1°	11.7% 9.2°	23.9% 14.8°	32.2% 14.8°	38.7% 15.1°	28.0% 12.3°	18.5% 10.5°	29.1% 16.0°	180°	Search Space: Small
weighted	38.8% 13.6°	24.0% 10.5°	31.9% 10.9°	11.6% 9.6°	46.7% 12.3°	39.0% 13.6°	23.7% 10.6°	32.1% 11.4°	10.8% 9.8°	45.3% 12.8°	37.1% 13.3°	23.3% 10.3°	30.2% 10.2°	9.7% 9.4°	46.0% 12.0°	36.5% 13.4°	22.6% 10.3°	30.2% 10.5°	9.1% 9.4°	44.9% 12.5°		
fill	21.2% 7.7°	23.9% 8.5°	28.5% 9.2°	22.6% 8.3°	28.1% 7.9°	20.7% 7.6°	20.7% 7.9°	26.3% 9.4°	18.3% 7.7°	26.7% 8.3°	17.1% 7.4°	17.6% 8.1°	17.4% 8.2°	11.7% 7.7°	21.8% 7.7°	16.6% 7.5°	15.6% 7.7°	16.3% 8.3°	10.2% 7.2°	21.3% 7.7°	220°	
weighted	33.5% 11.4°	19.3% 8.5°	24.6% 7.7°	7.9% 7.2°	42.3% 10.0°	32.9% 11.1°	18.3% 8.4°	25.2% 8.2°	6.5% 7.0°	41.4% 10.5°	32.3% 11.2°	18.5% 8.3°	24.3% 7.3°	6.9% 6.9°	42.4% 9.7°	31.9% 11.1°	18.0% 8.1°	23.9% 7.7°	5.8% 6.8°	41.2% 10.2°		
complete											10.8% 4.7°	13.1% 5.7°	2.0% 2.2°	7.2% 4.5°	7.9% 3.0°	11.9% 5.2°	10.2% 5.4°	1.2% 2.5°	4.8% 3.8°	5.9% 4.1°	360°	

Table C.3: Detailed results for the **cross-database** tests on the database ***uni_**** as described in section 5.3 and 5.5.2. The best results are highlighted (red text color).

		Hs				HS			
		early	**late**	**diffuse**	**dark**	**early**	**late**	**diffuse**	**dark**
Search Space Large	**early**		1.6% 3.8°	5.0% 6.1°	6.3% 5.6°		11.4% 4.5°	22.0% 6.3°	27.2% 7.3°
	late			3.2% 5.0°	4.2% 5.1°			12.4% 4.8°	13.8% 5.1°
	diffuse				4.1% 4.8°				4.8% 3.6°
	dark								
		early	**late**	**diffuse**	**dark**	**early**	**late**	**diffuse**	**dark**
Search Space Small	**early**		0.6% 3.4°	2.7% 5.5°	2.8% 4.9°		1.1% 4.0°	4.7% 6.0°	6.0% 6.5°
	late			2.1% 4.5°	2.0% 4.4°			2.5% 4.5°	2.0% 4.6°
	diffuse				1.8% 4.2°				0.7% 3.4°
	dark								

Table C.4: Results of the **cross-database** tests on the database ***lab__**** for single database combinations. The experiments were carried out for the *large* and *small* search space as well as for the best working preprocessing techniques *Hs* and *HS*. All tests use the best working method *weighted* on panoramic images with an opening angle of 220°, constant noise, and tangent distance enabled. The best results are highlighted (red text color).

		Hs			HS		
		winter	**early**	**late**	**winter**	**early**	**late**
Search Space Large	**winter**		58.2% 10.1°	2.1% 5.1°		32.3% 9.6°	2.5% 4.2°
	early			61.6% 13.2°			27.1% 9.6°
	late						
		winter	**early**	**late**	**winter**	**early**	**late**
Search Space Small	**winter**		30.9% 8.6°	1.4% 4.7°		8.5% 8.4°	1.1% 4.0°
	early			39.4% 11.2°			7.8% 8.2°
	late						

Table C.5: Results of the **cross-database** tests on the database ***uni__**** for single database combinations. The experiments were carried out for the *large* and *small* search space as well as for the best working preprocessing techniques *Hs* and *HS*. All tests use the best working method *weighted* on panoramic images with an opening angle of 220°, constant noise, and tangent distance enabled. The best results are highlighted (red text color).

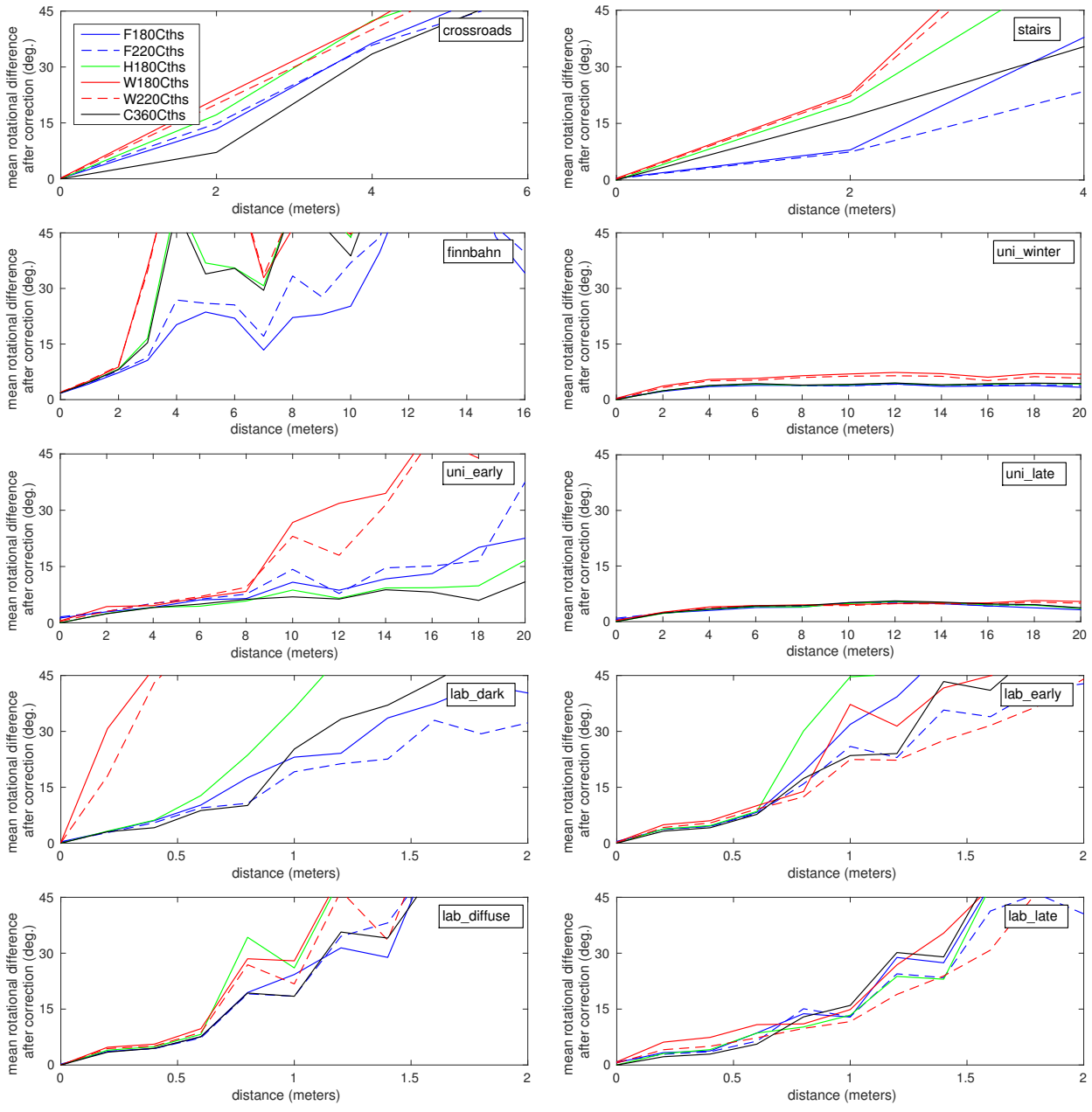


Figure C.1: Detailed results for the **translation** tests on all grid databases using the ***large* search space** with ***tangent distance* disabled**. The X-axis shows the translation in meters between two locations at which the current view and snapshot were captured and the Y-axis the mean rotational difference after correction. For details, see section 5.5.3.

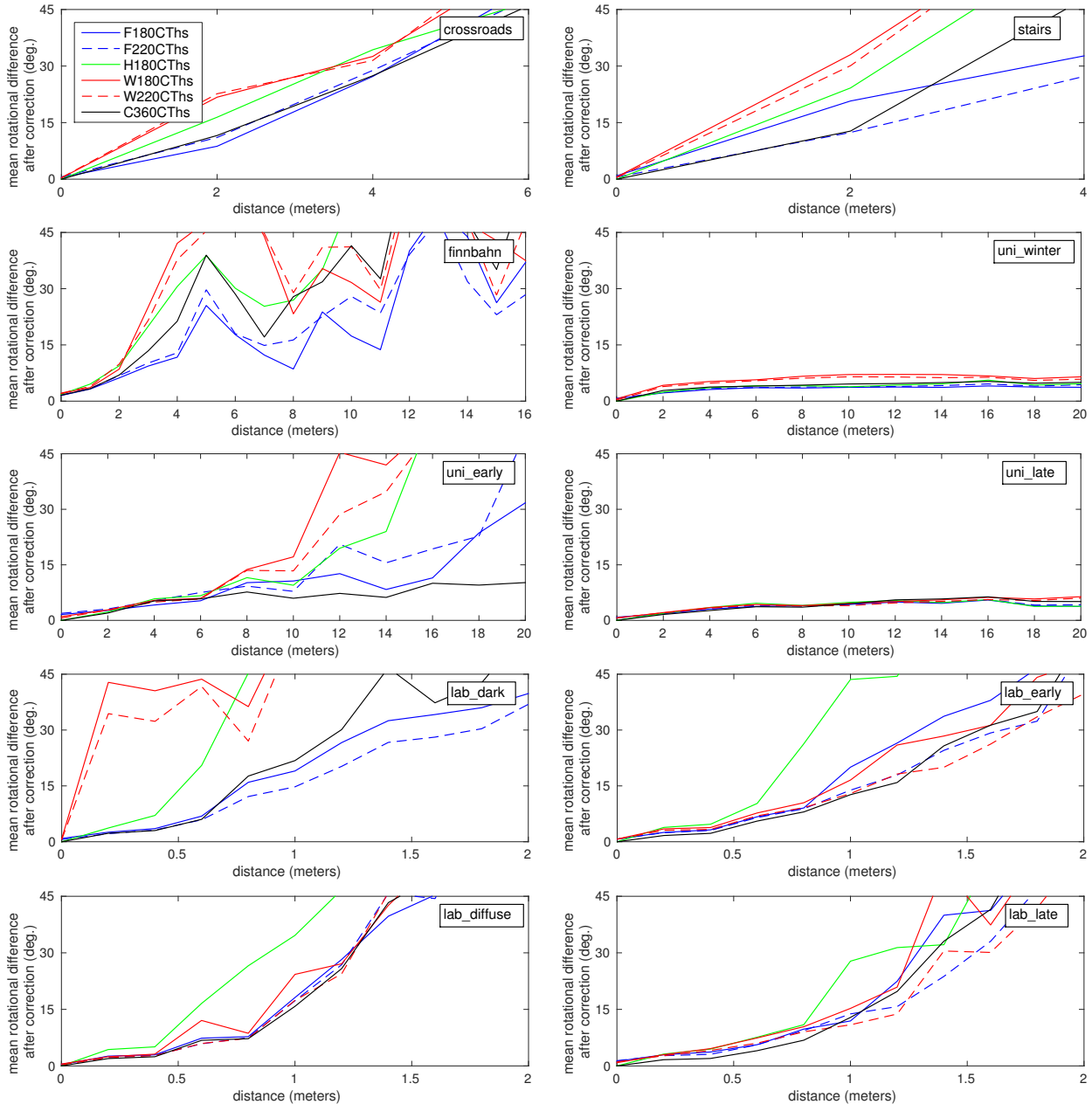


Figure C.2: Detailed results for the **translation** tests on all grid databases using the ***large* search space** with ***tangent distance* enabled**. The X-axis shows the translation in meters between two locations at which the current view and snapshot were captured and the Y-axis the mean rotational difference after correction. For details, see section 5.5.3.

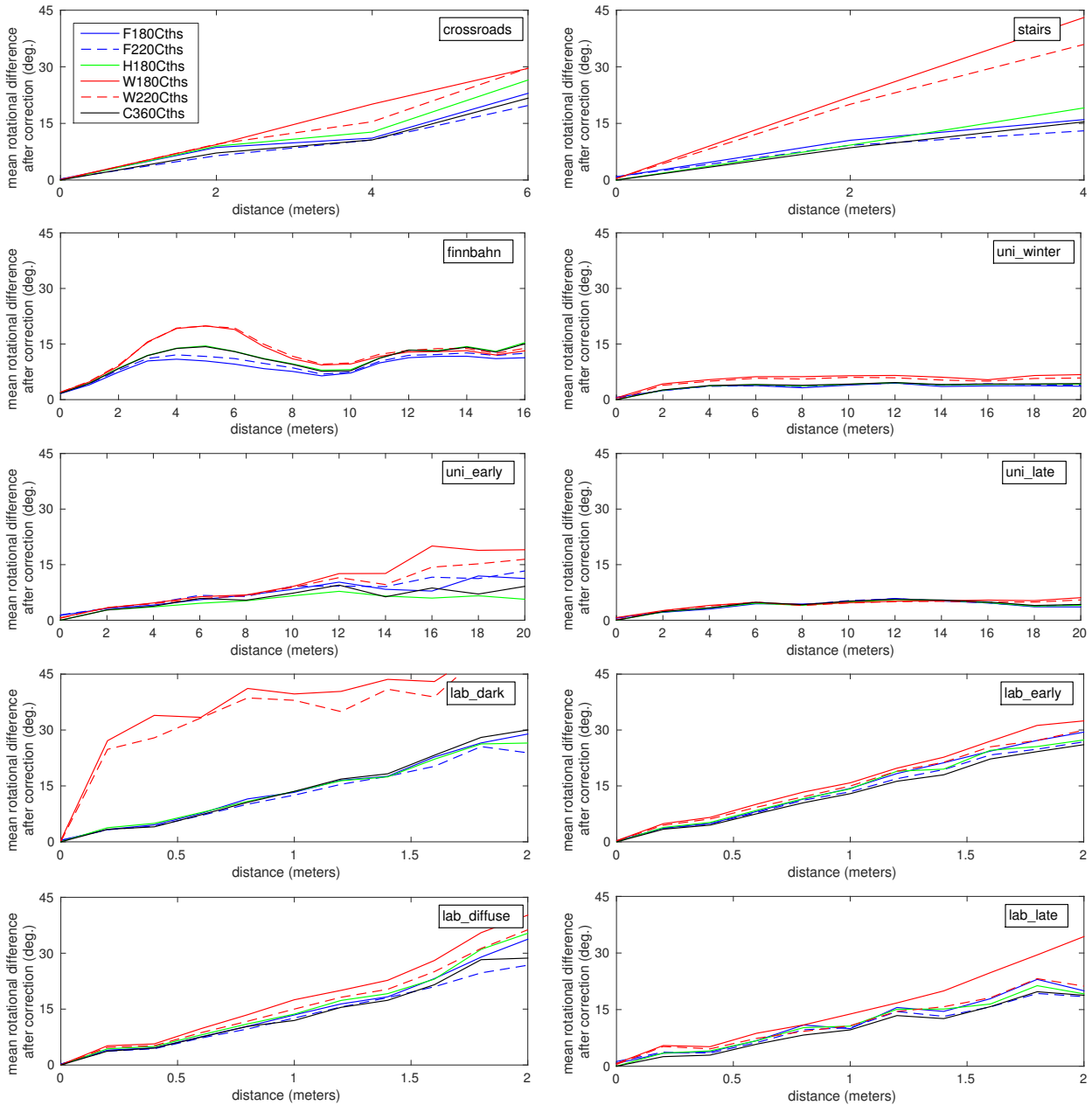


Figure C.3: Detailed results for the **translation** tests on all grid databases using the ***small* search space** with ***tangent distance* disabled**. The X-axis shows the translation in meters between two locations at which the current view and snapshot were captured and the Y-axis the mean rotational difference after correction. For details, see section 5.5.3.

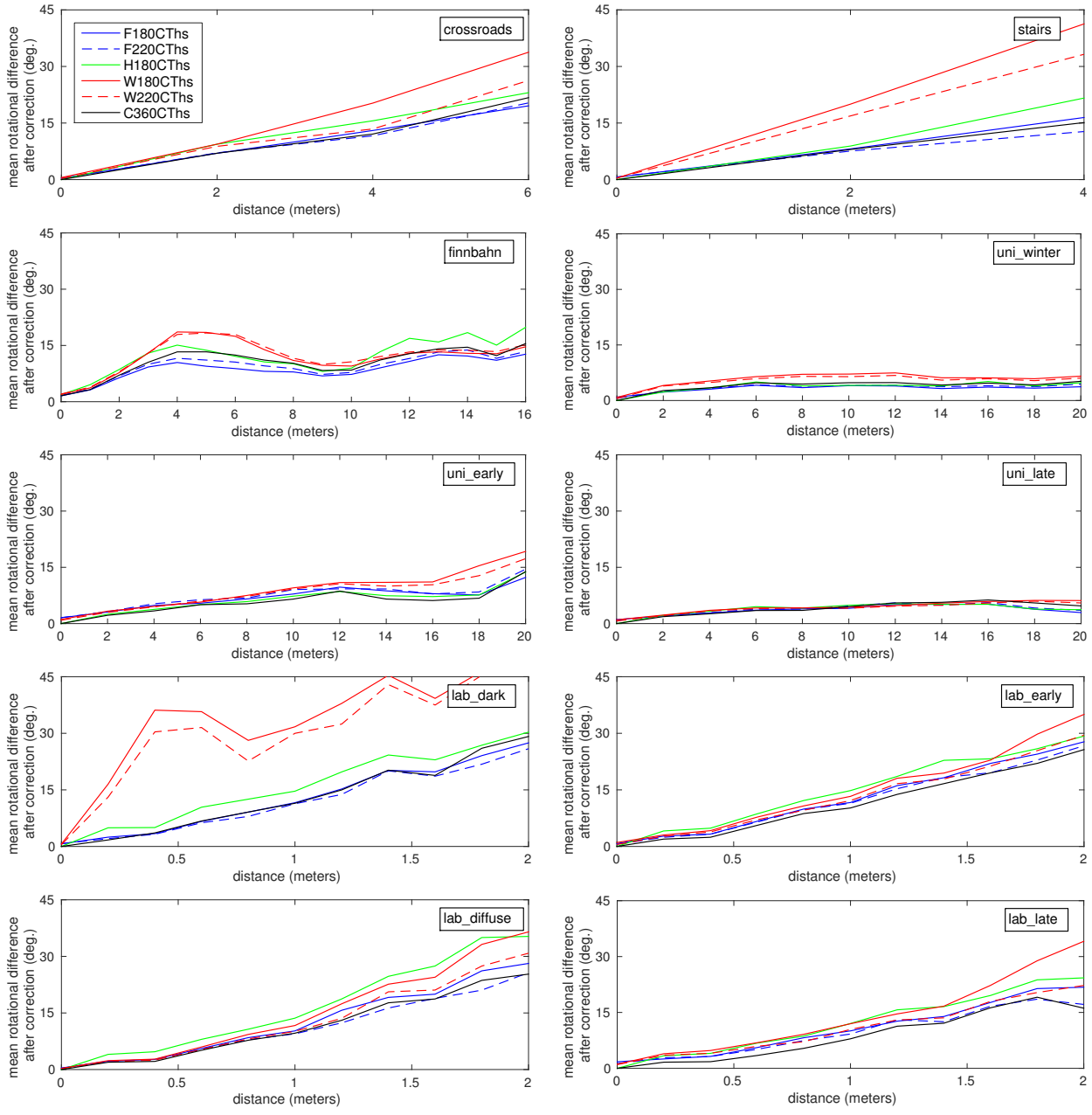


Figure C.4: Detailed results for the **translation** tests on all grid databases using the ***small* search space** with ***tangent distance* enabled**. The X-axis shows the translation in meters between two locations at which the current view and snapshot were captured and the Y-axis the mean rotational difference after correction. For details, see section 5.5.3.

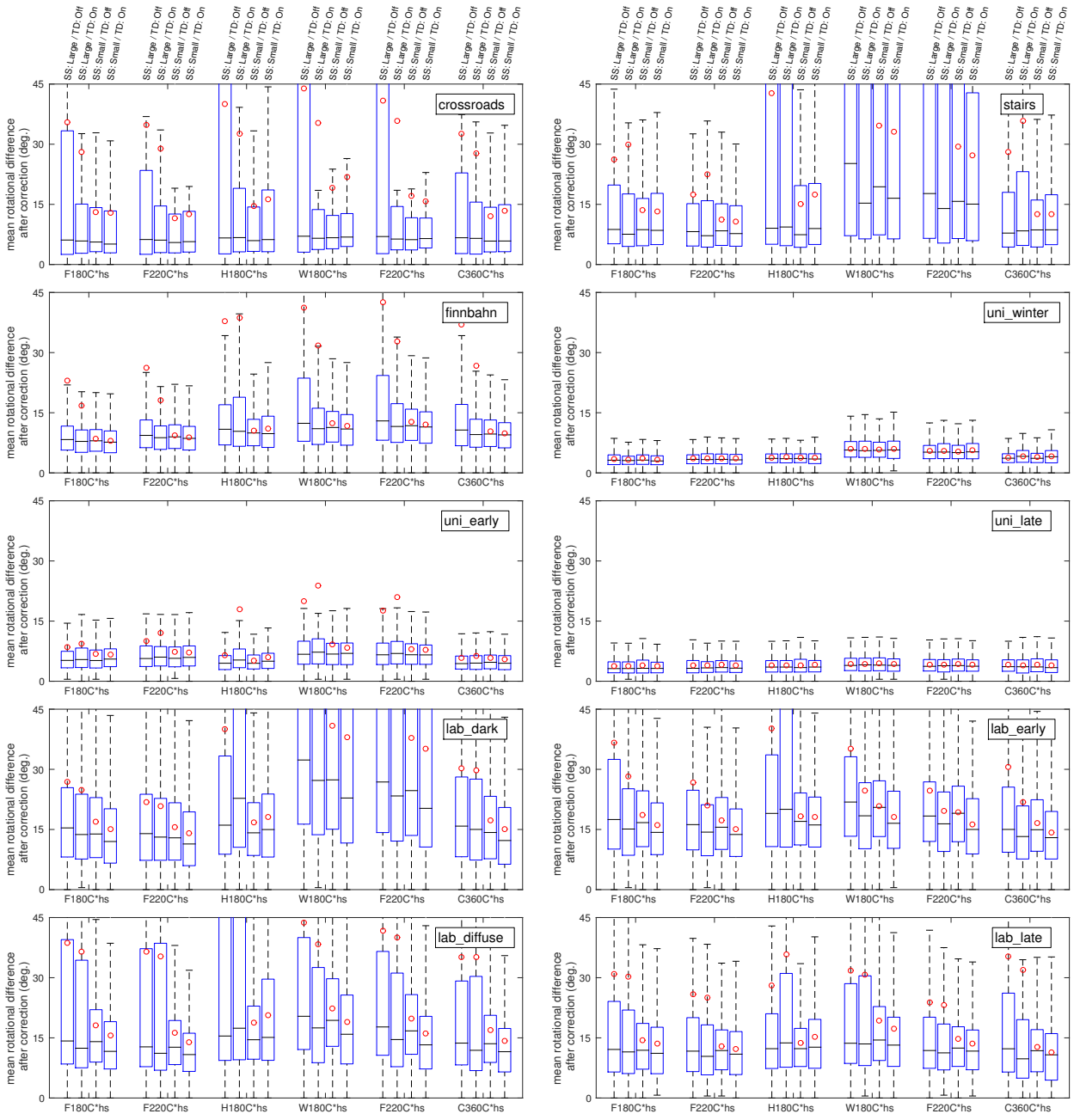


Figure C.5: This boxplots show the data presented in the previous figures, i. e. each bar refers to the data shown in one of the figures C.1-C.4. For all grid databases, the mean (red dots), median (black bars), the 25th and 75th percentiles (blue boxes), and a coverage of $3\sigma \hat{=} 97.7\%$ (black dashed lines) is shown. For details, see section 5.5.3.

C.3 Code: 3D-Warping

The following C++ code shows the implementation of 3D-warping using our *libShc* (section 3.10). The code is shortened for better readability, but all important information — especially the parameter sets used for the experiments in chapter 6 — are shown.

As usual, we first need to include the header of the *libShc*. Here we use the class *Shx* instead of *Shc* which provides additional functions for loading and saving images.

```
// include the Shc library with image loading/saving
// extension (Shx)
#include "Shx.h"

using namespace std;
using namespace shc;
```

Additionally to 3D-warping, the snapshot can be rotationally aligned with the current view using the visual 3D compass. As our results show, for strong tilt the optional use of the 3D-warping improves the performance of 2D-, min-, and 3D-warping. For more details on the visual 3D compass, see chapter 5; especially section 5.3.

```
// initialize the visual 3D compass instance
Shx init_vc() {

  Shx vc;
  // initialize coarse-to-fine approach;
  // for better readability angles are passed in degrees
  vc.init_rotations_sphere( 4°,   64°);
  vc.init_rotations_cone(   2°,   4°);
  vc.init_rotations_cone(   1°,   2°);
  // use L=16 bands and 1e4 sampling points
  vc.init_bands(16);
  vc.init_surface(1e4);
  // to fill in noise, 100 panoramic images
  // with constant noise are precalculated
  vc.init_noise(100, CONSTANT);
  // precalculate and initialize the visual 3D compass
  vc.init();

  // assuming an opening angle of 220°, we need to fill in
  // noise for all non-visible sampling points
  vc.set_noise_mask(220°);

  // calculate small translations and rotations
  // as transformations for the tangent distance
  vc.add_tangent_distance_translation(AXIS_X, 0.025);
  vc.add_tangent_distance_translation(AXIS_Y, 0.025);
  vc.add_tangent_distance_translation(AXIS_Z, 0.025);
  vc.add_tangent_distance_rotation(AXIS_X, 1°);
  vc.add_tangent_distance_rotation(AXIS_Y, 1°);
  vc.add_tangent_distance_rotation(AXIS_Z, 1°);

  // Use one-sided tangent distance
  vc.set_tangent_distance(ONESIDED);

  return vc;

}
```

The following code shows the initialization necessary for 3D-warping. A set of translations (α, d, h) is constructed and the corresponding translation matrices are precalculated. Again, we use a visual 3D compass to rotationally align the warped current views with the snapshot *after*

translation. Since the visual 3D compass is called for each translation (α, d, h), we only correct for a rather small and coarse set of rotations to reduce the computation times.

```
1
2  // initialize the 3D-warping instance
3  Shx init_warp3D() {
4
5    // initialize the compass for the inner loop of 3D-warping,
6    // compare init_vc();
7    // note that the rotation parameters are chosen differently
8    Shx w3d;
9    w3d.init_rotations_sphere( 8.0°,  16.0°);
10   w3d.init_rotations_cone(   4.0°,   8.0°);
11   w3d.init_rotations_cone(   2.0°,   4.0°);
12   w3d.init_bands(16);
13   w3d.init_surface(1e4);
14   w3d.init_noise(100, CONSTANT);
15   w3d.init();
16   w3d.set_noise_mask(220°);
17
18   // create a set of translations for 3D-warping:
19   // a -> angle                (as for 2D-/min-warping)
20   // d -> distance             (as for 2D-/min-warping)
21   // h -> height above ground (not possible for 2D-/min-warping)
22   // for better readability, we abbreviate the explizit construction
23   VectorReal a = 0°, 15°, ..., 345°;
24   VectorReal d =  0.05,  0.10, ..., 0.30;
25   VectorReal h = -0.30, -0.15, ..., 0.30;
26
27   for (int ih=0; ih<h.size(); ih++) {
28     for (int id=0; id<d.size(); id++) {
29       for (int ia=0; ia<a.size(); ia++) {
30         // calculate the 3D coordinate for the given translation
31         Coor3d translation(cos(a(ia))*d(id), sin(a(ia))*d(id), h(ih));
32         // calculate the translation matrix in the basis of RSH
33         MatrixReal m = w3d.create_matrix_warp(translation, VISUAL);
34         // add translation matrix to the w3d instance
35         w3d.add_transform(m, DENSE);
36       }
37     }
38   }
39
40   return w3d;
41
42 }
```

Next we show the 3D-warping code used to determine the home vector between a current view and snapshot. The complete 3D-warping algorithm consists of four stages: First, the visual 3D compass (chapter 5) can optionally be used to rotationally align the snapshot with the current view. Second, for each movement hypothesis the current view is warped by applying the translation (α, d, h). Third, the warped current view is rotationally aligned with the snapshot. Fourth, we search for the movement hypothesis which minimizes the integral squared error (ISE, section 3.9.1) between the warped current view and snapshot. Note that for a systematic search the third phase can also be replaced by a visual 3D compass.

```
1
2  // perform 3D-warping for a pair of CV and SS
3  Coor3d warping(Shx& vc, Shx& w3d, Shpm& cv, Shpm& ss) {
4
5    // optional: determine rotational offset between SS and CV ...
6    Xyz xyz_vc = vc.compass(ss, cv);
7    // ... and rotate the SS accordingly
```

```
  Shpm ss_rot = vc.rotate(ss, xyz_vc);

  // create vector to store the warped CV's of each movement hypothesis
  VecShpm tt(w3d.get_transform_size());

  // for each movement hypothesis i ...
  for (int i=0; i<w3d.get_transform_size(); i++) {
    // ... perform the required warp in the basis of RSH
    tt[i]       = w3d.transform(cv, i);
    // ... determine rotational offset between warped CV and SS
    Xyz xyz_w3d = w3d.compass(tt[i], ss_rot);
    // ... and rotate the warped CV accordingly
    tt[i]       = w3d.rotate(tt[i], xyz_w3d);
  }

  // compute the ISE between all warped CV's and SS
  VectorReal err = w3d.get_feature_difference(tt, ss_rot, ISE, 1);

  // find minimal ISE and return corresponding home vector
  // using an appropriate indexing function
  Coor3d homing_vector = err_to_coor(...);

  return homing_vector;

}
```

The following code finally shows how to perform 3D-warping on an image pair.

```

int main() {

  // initialize visual 3D compass instance
  Shx vc  = init_vc();
  // initialize 3D-warping instance
  Shx w3d = init_warp3D();

  // load CV and SS from image files
  Shpm cv = w3d.load_shpm("cv.bmp");
  Shpm ss = w3d.load_shpm("ss.bmp");
  warping(vc, w3d, cv, ss);

}
```

APPENDIX D

Full-Spherical Panoramic Image Databases

This appendix describes the experimental setup used to collect the panoramic images (section D.1) and gives detailed descriptions for the collected panoramic image databases (section D.2). The panoramic image databases are primarily used for visualization and navigation experiments.

D.1 Experimental Setup

The panoramic images were captured using a full-spherical camera consisting of two back-to-back mounted cameras with fish-eye lenses (figure D.1). While the cameras are able to capture a full-spherical panoramic image, the rig itself is visible and creates a blind spot. The currently usable fraction of the panoramic image is equivalent to a skywards facing camera equipped with a fish-eye objective with an opening angle of 310°. The experimental setup and the software to create the panoramic images was kindly provided by Wolfgang Stürzl and Alejandro Merello from the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR). An inertial measurement unit (IMU) is fixed at the base plate to determine the orientation of the experimental setup during the record of each database. Note that Fish-eye objectives have a decreased transmittance in the rim region of the camera image (vignetting). By the original software the vignetting was not corrected, therefore we apply a color correction by scaling the value of each pixel depending on its position in the image. The necessary correction values were determined by rotating the camera in front of a constant color emitter, here a white computer screen.

For each location, panoramic images were captured using multiple exposure times starting at 0.0125ms and then doubling in each step until 6.4 ms; the databases *lab_** have additional exposure times of up to 102.4 ms. Using these images, we additionally created high dynamic range (HDR) images as described in chapter 2.

D.2 Database Descriptions

We collected various panoramic image databases over ten months (October 2015 to July 2016) in the vicinity of Bielefeld university. The databases differ in their functionality and can mainly be assigned to one of three categories: First, databases in which the position of the experimental setup is determined by a grid (*stairs*, *crossroads*) or a line (*finnbahn*) with fixed step sizes. Due to the known ground truth positions, these databases allow to systematically test visual navigation techniques without the necessity to use a hardware platform (e.g. a wheeled robot). We refer to these as **grid databases**. Second, databases with known ground truth — as in the previous case — but collected multiple times under differing lighting conditions (*lab_**, *uni_**). We refer to these as **cross-databases**. Cross-databases can be used to test the applicability of visual navigation techniques under varying lighting conditions. Third, we collected full-spherical panoramic images in the university main building (*mixed_indoor*) and on the campus (*mixed_winter*, *mixed_summer*). For these images no positional informations are available, but allow to test the visual compass on strongly differing environments. We refer to these as **mixed databases**.

A complete list of all panoramic image databases — as well as detailed information about the

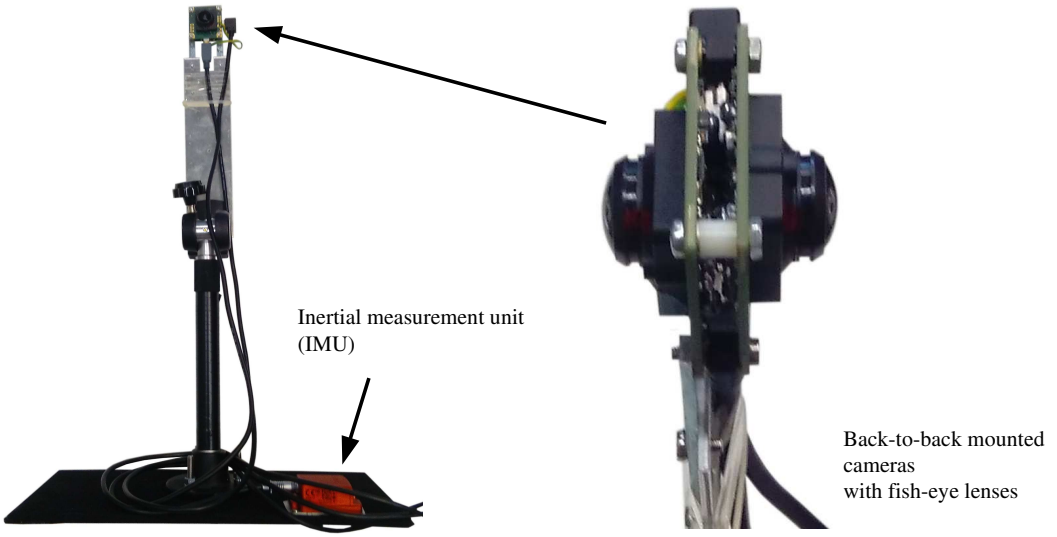


Figure D.1: The experimental setup used to capture full-spherical panoramic images. It contains two cameras with fish-eye lenses (angle of view slightly above 180°) mounted back-to-back. By mapping the images captured by both cameras onto a sphere, a full-spherical panoramic image can be created. An inertial measurement unit (IMU) is mounted to measure the orientation of the setup.

recording date and daytime, the number of collected imaged, etc. — can be found in table D.1. Afterwards, for each database an information panel is shown with exemplary images, sketches, and additional information.

Name	Abbreviation	Date	Daytime	Number of images	Dimension in meter	Ground truth	Cross-Database	Skyline	Exposure-Time
Laboratory	*lab_early*	11.04.2016	09:00	$15 \times 10 = 150$	4.2×2.7	Grid	✓	✗	12.8 ms
	lab_midday	11.04.2016	13:00						
	lab_diffuse	11.04.2016	17:00						
	lab_dark	12.04.2016	09:00						
University	*uni_early*	26.10.2015	11:00	$20 \times 4 = 80$	27.4×4.7	Grid	✓	✓	0.2 ms
	uni_late	23.10.2015	17:00						
	uni_winter	08.12.2015	14:00						
Stairs	*stairs*	31.10.2015	11:00	$5 \times 8 = 40$	$6.0 \times 10.8 \times 2.9$	Grid	✗	✗	0.05 ms
Crossroads	*crossroads*	31.10.2015	12:00	$7 \times 5 = 35$	13.7×7.8				0.1 ms
Finnbahn	*finnbahn*	28.07.2016	16:00	$201 \times 1 = 201$	20				0.8 ms
Meadow	*meadow*	21.02.2017	12:00	$13 \times 19 = 247$	13×19	Simulation	✗	✓	-
Mixed	*mixed_indoor*	08.04.2016	10:00	20		None	✗	✗	3.2 ms
	mixed_winter	08.04.2016	11:00	30					0.8 ms
	mixed_summer	27.07.2016	15:00	25					0.2 ms

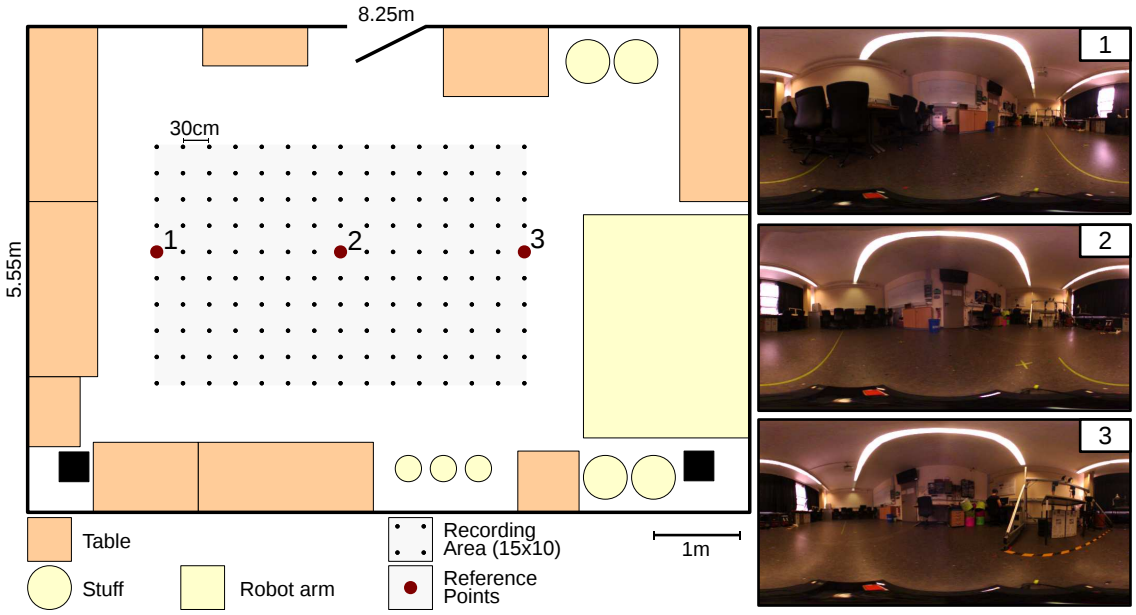
Table D.1: Overview of all collected panoramic image databases. Throughout this work, the databases are referred to by their abbreviations. Various information about the record date and daytime, the number of collected images, and the dimensions of the database are given. For each database it is noted how the ground truth was obtained: *Grid*: The records were collected on an evenly spaced grid (e.g. paving or created using tape measures). *Simulation*: The panoramic images were rendered in a simulation. *None*: The panoramic images were collected at random locations, there is no ground truth available. Databases for which a ground truth of the skyline is available are marked. If not stated otherwise, we use for all navigation experiments panoramic images captured with the given exposure time.

Panoramic Database: Laboratory

Overview:

Name:	Laboratory
Abbreviation:	*lab_**
Cross-Database:	✓
Skyline:	✗

Sketch:



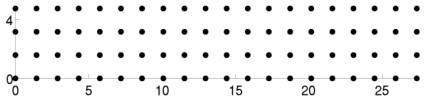
Panoramic Database: Laboratory

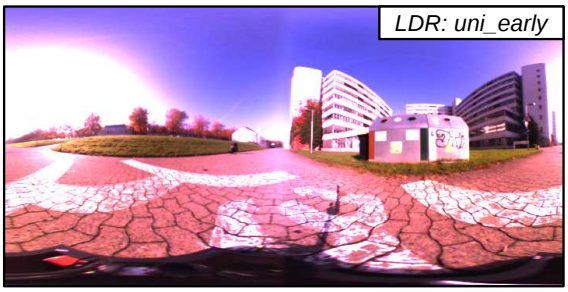
Description:
This database was recorded in the robotics lab of the faculty of technology at Bielefeld university and contains four different lighting conditions: *lab_early*: Diffuse natural lighting from outside, lights on. *lab_late*: Direct natural lighting from outside, lights on. *lab_diffuse*: Diffuse natural lighting and diffuse lighting by spotlights, lights off. *lab_dark*: Diffuse natural lighting from outside, lights off. The distance to surrounding objects is in the laboratory is in average around 3-4 meters. For better comparison, all image shown were captured with the same exposure time.

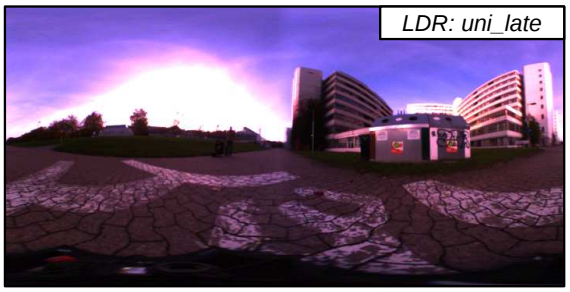
Panoramic Database: University

Overview:

Name:	University
Abbreviation:	*uni_**
Cross-Database:	✓
Skyline:	✓





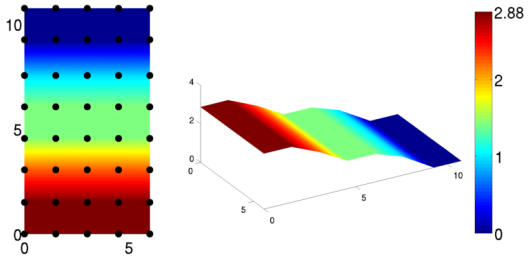


Description:
This database was recorded on a street parallel to the main building of Bielefeld university. A total of three databases were collected under different lighting conditions. Except for a dumpster, the distance to all surrounding objects is relatively large, resulting in comparable small visual changes between different recording locations. For better comparison, all image shown were captured with the same exposure time.

Panoramic Database: Stairs

Overview:

Name:	Stairs
Abbreviation:	*stairs*
Cross-Database:	✗
Skyline:	✗

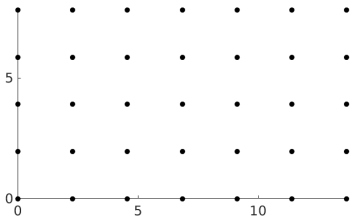


Description:
With a total of 2.88 m height difference on an area of around 6 m × 11 m, this database contains large altitude changes within the databases. Three different altitude levels are separated by two stairs and both sides of the stairs are (painted) concrete walls. From lower levels, the stairs and walls block the parts of the cameras field of view, resulting in strong occlusions.

Panoramic Database: Crossroads

Overview:

Name:	Crossroads
Abbreviation:	*crossroads*
Cross-Database:	✗
Skyline:	✗

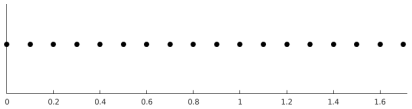


Description:
This database was collected on an area between several trees, bushes, and a hedge during autumn. There is a high amount of leaves on the ground and the sun intensity is high. Due to incident sunlight onto one of the cameras, the right side of the database images suffer from indirect lighting effects inside the fisheye lens (increased brightness).

Panoramic Database: Finnbahn

Overview:

Name:	Finnbahn
Abbreviation:	*finnbahn*
Cross-Database:	✗
Skyline:	✗

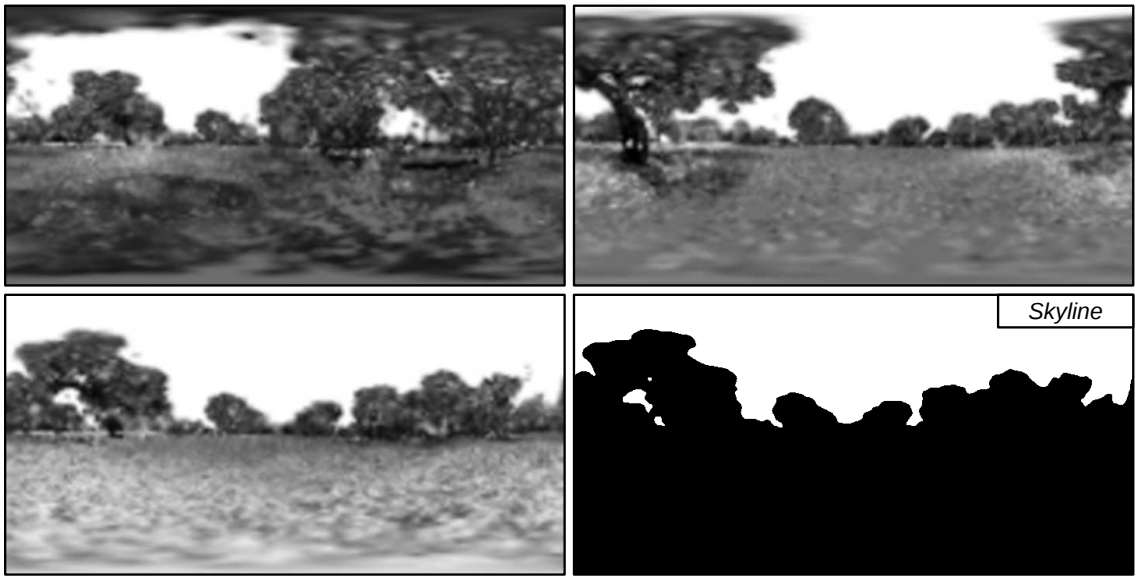


Description:
This database was collected during July on the Finnbahn close to the main building of Bielefeld university. It contains images in 10 cm steps on a straight 20 m route. The scene is dominated by bushes and trees and does contain nearly no salient landmarks or structures.

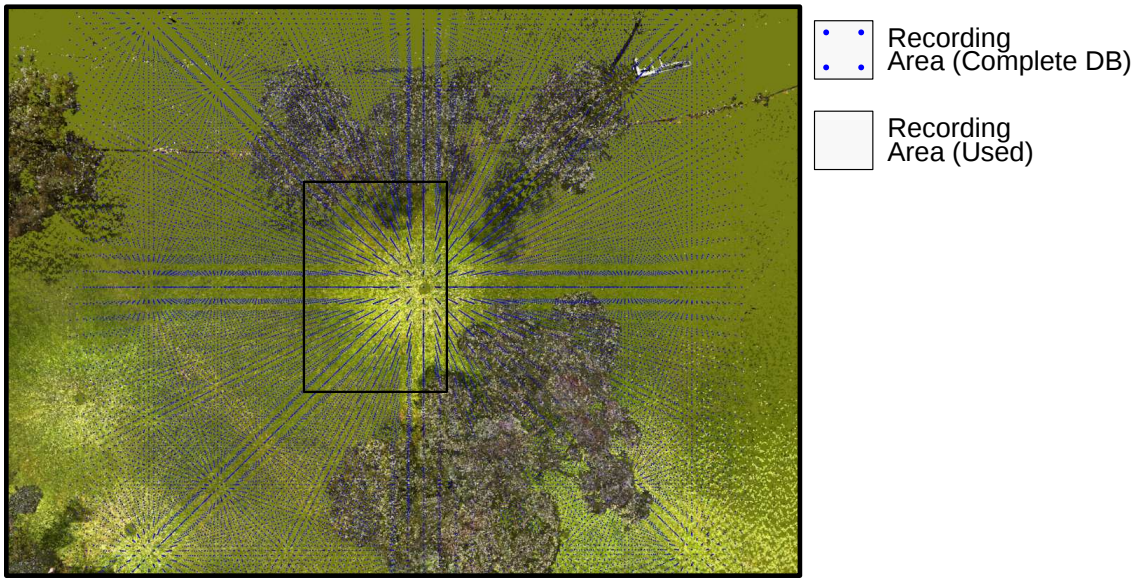
Panoramic Database: Meadow

Overview:

Name:	Meadow
Abbreviation:	*meadow*
Cross-Database:	✗
Skyline:	✓



Sketch:



Description:
This database was rendered in a 3D model of an environment recorded at Canberra, Australia (Stürzl et al., 2015); a large database of images was kindly provided by the authors. We extracted an area of 13 m × 19 m (marked) at an elevation of 1 m about ground. Since all sky pixels have maximal brightness, the skyline could simply be extracted.

Panoramic Database: Mixed

Overview:

Name: Mixed
Abbreviation: *mixed_**
Cross-Database: ✗
Skyline: ✗

Description:

We collected a mixture of indoor and outdoor panoramic images at random locations in the vicinity of Bielefeld university. The first set *mixed_indoor* was collected at several indoor locations as offices, lobbies, or staircases. The second set *mixed_winter* was collected outdoors during early April and contains panoramic images of mostly park-like environments, sometimes containing buildings. The last set *mixed_summer* was collected during July in the area around the Finnbahn close to the main building of Bielefeld university and contains panoramic images dominated by trees, bushes, and grass. Note that all images shown above are HDR images.

Bibliography

Abraham, R. and Simon, P. (2013). Review on mosaicing techniques in image processing. In *Proceedings of the International Conference on Advanced Computing and Communication Technologies (ACCT)*, pages 63–68. IEEE. (↗ p. 11)

Adarve, J. D. and Mahony, R. (2016). A filter formulation for computing real time optical flow. *Robotics and Automation Letters*, 1(2):1192–1199. (↗ p. 124)

Aguerrebere, C., Delon, J., Gousseau, Y., and Muse, P. (2014). Best algorithms for HDR image generation. A study of performance bounds. *SIAM Journal on Imaging Sciences*, 7(1):1–34. (↗ pp. 21 and 163)

Aksoy, V. and Camlitepe, Y. (2014). A behavioral analysis of achromatic cue perception by the ant *Cataglyphis aenescens* (Hymenoptera; Formicidae). *Turkish Journal of Zoology*, 38(2):199–208. (↗ p. 49)

Alahi, A., Ortiz, R., and Vandergheynst, P. (2012). FREAK: Fast retina keypoint. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 510–517. IEEE. (↗ p. 7)

Alpaydin, E. (2004). *Introduction to Machine Learning*. The MIT Press, Cambridge, Massachusetts, 1st edition. (↗ p. 26)

Ardin, P., Mangan, M., Wystrach, A., and Webb, B. (2015). How variation in head pitch could affect image matching algorithms for ant navigation. *Journal of Comparative Physiology A*, 201(6):585–597. (↗ p. 100)

Arleo, A. (2000). *Spatial learning and navigation in neuro-mimetic systems*. PhD thesis, École polytechnique fédérale de Lausanne. (↗ p. 9)

Arleo, A. and Gerstner, W. (2000). Modeling rodent head-direction cells and place cells for spatial learning in bio-mimetic robotics. *From Animals to Animats*, 6(1):236–245. (↗ p. 9)

Arroyo, R., Alcantarilla, P. F., Bergasa, L. M., Yebes, J. J., and Gámez, S. (2014). Bidirectional loop closure detection on panoramas for visual navigation. In *Proceedings of Intelligent Vehicles Symposium (IV)*, pages 1378–1383. IEEE. (↗ p. 8)

Baddeley, B., Graham, P., Husbands, P., and Andrew, P. (2012). A model of ant route navigation driven by scene familiarity. *PLOS Computational Biology*, 8(1):1–16. (↗ p. 9)

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv:1511.00561*. (↗ p. 147)

Barreto, J., Roquette, J., Sturm, P., and Fonseca, F. (2009). Automatic camera calibration applied to medical endoscopy. In *Proceedings of the British Machine Vision Conference (BMVA)*, pages 1–10. (↗ p. 125)

Basri, R. and Jacobs, D. W. (2003). Lambertian reflectance and linear subspaces. *Transactions on pattern analysis and machine intelligence*, 25(2):218–233. (↗ p. 53)

Basten, K. and Mallot, H. A. (2010). Simulated visual homing in desert ant natural environments: Efficiency of skyline cues. *Biological Cybernetics*, 102(5):413–425. (↗ pp. 18 and 46)

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer vision and image understanding*, 110(3):346–359. (↗ pp. 7 and 104)

Bazin, J.-C., Demonceaux, C., Vasseur, P., and Kweon, I. (2008). Rotation estimation and vanishing point extraction by omnidirectional vision in urban environment. *International Journal of Robotics Research*, 31(1):63–81. (↗ pp. 5, 114, and 124)

Bazin, J.-C., Kweon, I., Demonceaux, C., and Vasseur, P. (2009). Dynamic programming and skyline extraction in catadioptric infrared images. In *Proceedings of the International Conference on Robotics*

and Automation (ICRA), pages 409–416. IEEE. (↗ p. 17)

Benosman, R. and Kang, S. B. (2001). *Panoramic Vision: Sensors, theory, and applications*. Springer, New York, 1st edition. (↗ p. 11)

Bird, R. E. and Hulstrom, R. L. (1983). Terrestrial solar spectral data sets. *Solar Energy*, 30(6):563–573. (↗ p. 14)

Blanco, M. A., Florez, M., and Bermejo, M. (1997). Evaluation of the rotation matrices in the basis of real spherical harmonics. *Journal of Molecular Structure*, 419(1):19–27. (↗ pp. 68 and 70)

Bloch, C. (2008). *Das HDRI-Handbuch*. Dpunkt Verlag, Heidelberg, Germany, 1st edition. (↗ p. 21)

Bober, M., Krzysztof, K., and Skarbek, W. (2003). Face recognition by Fisher and scatter linear discriminant analysis. *Lecture Notes in Computer Science*, 2756(1):638–645. (↗ p. 26)

Booij, O., Terwijn, B., Zivkovic, Z., and Krose, B. (2007). Navigation using an appearance based topological map. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 3927–3932. IEEE. (↗ p. 3)

Borgerding, M. (2006). kissFFT v1.30. https://sourceforge.net/projects/kissfft/. [Online; accessed 10-April-2016]. (↗ p. 87)

Bormann, R., Jordan, F., Li, W., Hampp, J., and Hỳgele, M. (2016). Room segmentation: Survey, implementation, and analysis. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 1019–1026. IEEE. (↗ p. 2)

Briscoe, A. D. and Chittka, L. (2001). The evolution of color vision in insects. *Annual Review of Entomology*, 46(1):471–510. (↗ p. 15)

Bruhn, A., Weickert, J., and Schnörr, C. (2005). Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231. (↗ pp. 5, 114, and 124)

Bülow, T. (2001). Spherical diffusion for surface smoothing and denoising. Technical Report, University of Pennsylvania. (↗ p. 88)

Burel, G. and Henoco, H. (1995). Determination of the orientation of 3D objects using spherical harmonics. *Graphical Models and Image Processing*, 57(5):400–408. (↗ pp. 51 and 115)

Business Insider (2015). The robotics market report: The fast-multiplying opportunities in consumer, industrial, and office robots. http://www.businessinsider.de/growth-statistics-for-robots-market-2015-2. [Online; accessed 05-January-2017]. (↗ p. 1)

Byerly, W. E. (1893). *An elementary treatise on Fourier's series and spherical, cylindrical, and ellipsoidal harmonics, with applications to problems in mathematical physics*. Ginn & Company, London, England, 1st edition. http://www.gutenberg.org/files/29779/29779-pdf.pdf [Online; accessed 15-March-2015]. (↗ p. 65)

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). BRIEF: Binary robust independent elementary features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 778–792. Springer. (↗ p. 7)

Camlitepe, Y. and Aksoy, V. (2010). First evidence of fine colour discrimination ability in ants (Hymenoptera, Formicidae). *Journal of Experimental Biology*, 213(1):72–77. (↗ p. 49)

Carey, N. and Stürzl, W. (2011). An insect-inspired omnidirectional vision system including UV-sensitivity and polarisation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 312–319. IEEE. (↗ p. 17)

Cartwright, B. A. and Collett, T. S. (1983). Landmark learning in bees: Experiments and models. *Journal of Comparative Physiology*, 151(1):521–543. (↗ p. 14)

Chen, J. Q., Ping, J., and Wang, F. (2002). *Group representation theory for physicists*. World Scientific Publishing, Singapore, 2nd edition. (↗ pp. 52, 56, and 59)

Cheng, K. and Freas, C. A. (2015). Path integration, views, search, and matched filters: The contributions of Rüdiger Wehner to the study of orientation and navigation. *Journal of Comparative Physiology A*, 201(6):517–532. (↗ p. 13)

Chirikjian, G. S. and Kyatkin, A. B. (2001). *Engineering applications of noncommutative harmonic analysis*.

CRC Press, Boca Raton, Florida, 1st edition. (↗ pp. 52 and 56)

Chittka, L. (1996). Optimal sets of color receptors and color opponent systems for coding of natural objects in insect vision. *Journal of Theoretical Biology*, 181(2):179–196. (↗ p. 15)

Chittka, L., Beier, W., Hertel, H., Steinmann, E., and Menzel, R. (1992). Opponent colour coding is a universal strategy to evaluate the photoreceptor inputs in Hymenoptera. *Journal of Comparative Physiology A*, 171(3):545–563. (↗ p. 15)

Chittka, L., Shmida, A., Troje, N., and Menzel, R. (1994). Ultraviolet as a component of flower reflections, and the colour perception of Hymenoptera. *Vision Research*, 34(11):1489–1508. (↗ p. 15)

Choi, C. H., Ivanic, J., Gordon, M. S., and Ruedenberg, K. (1999). Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion. *Journal of Chemical Physics*, 111(19):8825–8831. (↗ p. 70)

Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., and Thrun, S. (2005). *Panoramic Vision: Sensors, Theory, and Applications*. The MIT Presss, London, England, 1st edition. (↗ p. 3)

Clark, R. N., Swayze, G. A., Wise, R., Livo, K. E., Hoefen, T. M., Kokaly, R. F., and Sutley, S. J. (2007). U.S. geological survey digital spectral library splib06a (data series 231). http://speclab.cr.usgs.gov/spectral.lib06/. [Online; accessed 09-August-2016]. (↗ p. 14)

Coemans, M. A. J. M., Vos Hzn, J. J., and Nuboer, J. F. W. (1994). The relation between celestial colour gradients and the position of the sun, with regard to the sun compass. *Vision Research*, 34(11):1461–1470. (↗ p. 43)

Cohen, T. S. and Welling, M. (2016). Group equivariant convolutional networks. *arXiv:1602.07576*. (↗ p. 56)

Collet, A., Berenson, D., Srinivasa, S. S., and Ferguson, D. (2009). Object recognition and full pose registration from a single image for robotic manipulation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 48–55. IEEE. (↗ p. 2)

Collett, T. S. and Collett, M. (2002). Memory use in insect visual navigation. *Nature Reviews Neuroscience*, 3(7):542–552. (↗ p. 13)

Collett, T. S. and Kelber, A. (1988). The retrieval of visuo-spatial memories by honeybees. *Journal of Comparative Physiology A*, 163(1):145–150. (↗ pp. 13 and 14)

Collett, T. S. and Rees, J. A. (1997). View-based navigation in Hymenoptera: Multiple strategies of landmark guidance in the approach to a feeder. *Journal of Comparative Physiology A*, 181(1):47–58. (↗ p. 13)

Collett, T. S. and Zeil, J. (1997). The selection and use of landmarks by insects. In Lehrer, M., editor, *Orientation and Communication in Arthropods*, pages 41–65. Birkhäuser Verlag, Basel, Swiss. (↗ p. 14)

Cooley, J. and Tukey, J. (1965). An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301. (↗ p. 55)

Corke, P., Paul, R., Churchill, W., and Newman, P. (2013). Dealing with shadows: Capturing intrinsic scene appearance for image-based outdoor localisation. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 2085–2092. IEEE/RSJ. (↗ pp. 3 and 95)

Corke, P., Strelow, D., and Singh, S. (2004). Omnidirectional visual odometry for a planetary rover. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, volume 4, pages 4007–4012. IEEE. (↗ p. 2)

Cummins, M. and Newman, P. (2007). Probabilistic appearance based navigation and loop closing. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 2042–2048. IEEE. (↗ p. 2)

Cummins, M. and Newman, P. (2008). FAB-MAP: Probabilistic localization and mapping in the space of appearance. *International Journal of Robotics Research*, 27(6):647–665. (↗ p. 3)

Cummins, M. and Newman, P. (2009). Highly scalable appearance-only SLAM - FAB-MAP 2.0. In *Proceedings of Robotics: Science and Systems (RSS)*, volume 5, pages 1–8. (↗ pp. 4, 8, 101, and 104)

Danos, M. and Maximon, L. C. (1965). Multipole matrix elements of the translation operator. *Journal of Mathematical Physics*, 6(5):766–778. (↗ p. 78)

Das, A. (2012). *Signal Conditioning*. Springer, Berlin, Germany, 1st edition. (↗ p. 90)

Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067. (↗ pp. 7 and 114)

Dayoub, F., Morris, T., Upcroft, B., and Corke, P. (2013). Vision-only autonomous navigation using topometric maps. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 1923–1929. IEEE/RSJ. (↗ p. 9)

Debevec, P. E. and Malik, J. (1998). Recovering high dynamic range radiance maps from photographs. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 369–378. (↗ pp. 21 and 162)

Dederscheck, D., Zahn, M., Friedrich, H., and Mester, R. (2010a). Optical rails: View-based track following with hemispherical environment model and orientation view descriptors. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 2752—2755. IEEE. (↗ pp. 95 and 119)

Dederscheck, D., Zahn, M., Friedrich, H., and Mester, R. (2010b). Slicing the view: Occlusion-aware view-based robot navigation. In Goesele, M., Roth, S., Kuijper, A., Schiele, B., and Schindler, K., editors, *Lecture Notes in Computer Science*, pages 111–120. Springer, Berlin, Germany. (↗ p. 115)

Dellaert, F., Fox, D., Burgard, W., and Thrun, S. (1999). Monte Carlo localization for mobile robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1322–1328. IEEE. (↗ p. 4)

Denuelle, A. and Srinivasan, M. V. (2016). A sparse snapshot-based navigation strategy for UAS guidance in natural environments. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 3455–3462. IEEE. (↗ pp. 7 and 9)

Deserno, M. (2004). How to generate equidistributed points on the surface of a sphere. http://www.cmu.edu/biolphys/deserno/pdf/sphere_equi.pdf. [Online; accessed 17-June-2015]. (↗ p. 89)

Differt, D. (2014). Hyperbolic mirror toolbox. http://www.ti.uni-bielefeld.de/html/research/outdoor/hmt/hyperbolic_mirror_toolbox.zip. Technical Report. (↗ pp. 11 and 20)

Differt, D. and Möller, R. (2015). Insect models of illumination-invariant skyline extraction from UV and green channels. *Journal of Theoretical Biology*, 380(7):444–462. (↗ p. 13)

Differt, D. and Möller, R. (2016). Spectral skyline separation: Extended landmark databases and panoramic imaging. *Sensors*, 16(10):1–23. (↗ p. 13)

Dillenseger, J.-L., Guillaume, H., and Patard, J.-J. (2006). Spherical harmonics based intrasubject 3D kidney modeling/registration technique applied on partial information. *Transactions on Biomedical Engineering*, 53(11):2185–2193. (↗ p. 51)

Durier, V., Graham, P., and Collett, T. S. (2003). Snapshot memories and landmark guidance in wood ants. *Current Biology*, 13(18):1614–1618. (↗ p. 14)

Dusha, D., Boles, W., and Walker, R. (2007). Attitude estimation for a fixed-wing aircraft using horizon detection and optical flow. In *Proceedings of the Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA)*, pages 485–492. IEEE. (↗ p. 17)

DWD (2014). Deutscher Wetterdienst. http://www.dwd.de/. [Online; accessed 14-Sept-2014]. (↗ p. 20)

Efron, B. and Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC press, Boca Raton, Florida, 1st edition. (↗ pp. 40 and 41)

Elkmann, N., Hortig, J., and Fritzsche, M. (2009). Cleaning automation. In Nof, S. Y., editor, *Handbook of Automation*, pages 1253–1264. Springer, Berlin, Germany. (↗ p. 1)

Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM: Large-scale direct monocular SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 834–849. Springer. (↗ pp. 7 and 56)

Engel, J., Sturm, J., and Cremers, D. (2012). Camera-based navigation of a low-cost quadrocopter. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 2815–2821. IEEE/RSJ. (↗ p. 114)

Erdmann, K. and Wildon, M. J. (2006). *Introduction to Lie algebras*. Springer, Berlin, Germany, 1st edition. (↗ p. 52)

Falcidieno, B. (2004). Aim@Shape project presentation. In *Proceedings of the International Conference on Shape Modeling and Applications (SMI)*, pages 329–335. IEEE. (↗ p. 84)

Fechner, G. T. (1860). *Elemente der Psychophysik*. Breitkopf & Härtel, Leipzig, Germany, 1st edition. (↗ pp. 15 and 21)

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395. (↗ pp. 6 and 124)

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188. (↗ p. 25)

Fleer, D. and Möller, R. (2017). Comparing holistic and feature-based visual methods for estimating the relative pose of mobile robots. *Robotics and Autonomous Systems*, 89(1):51–74. (↗ pp. 7, 134, 138, 146, and 150)

Folland, G. B. (1992). *Fourier analysis and its applications*. American Mathematical Society, Pacific Grove, California, 1st edition. (↗ p. 65)

Franz, M. O. and Mallot, H. A. (2000). Biomimetic robot navigation. *Robotics and autonomous Systems*, 30(1):133–153. (↗ p. 9)

Franz, M. O., Schölkopf, B., Mallot, H. A., and Bülthoff, H. H. (1998). Where did I take that snapshot? Scene-based homing by image matching. *Biological Cybernetics*, 79(1):191–202. (↗ pp. 6, 7, and 135)

Friedrich, H., Dederscheck, D., Krajsek, K., and Mester, R. (2007). View-based robot localization using spherical harmonics: Concept and first experimental results. In Goesele, M., Roth, S., Kuijper, A., Schiele, B., and Schindler, K., editors, *Lecture Notes in Computer Science*, pages 21–31. Springer, Berlin, Germany. (↗ p. 115)

Friedrich, H., Dederscheck, D., Rosert, E., and Mester, R. (2008). Optical rails: View-based point-to-point navigation using spherical harmonics. In Rigoll, G., editor, *Pattern Recognition*, pages 345–354. Springer, Berlin, Germany. (↗ pp. 18, 51, and 86)

Fuentes-Pacheco, J., Ruiz-Ascencio, J., and Rendón-Mancha, J. M. (2015). Visual simultaneous localization and mapping: A survey. *Artificial Intelligence Review*, 43(1):55–81. (↗ p. 3)

Fukushi, T. (2001). Homing in wood ants *Formica japonica*: Use of the skyline panorama. *Journal of Experimental Biology*, 204(12):2063–2072. (↗ p. 14)

Fukushi, T. and Wehner, R. (2004). Navigation in wood ants *Formica japonica*: Context dependent use of landmarks. *Journal of Experimental Biology*, 207(19):3431–3439. (↗ p. 14)

Gaël, G., Benoît, J., et al. (2010). Eigen v3.0. http://eigen.tuxfamily.org/. [Online; accessed 05-January-2016]. (↗ p. 87)

Garcia, J. E., Dyer, A. G., Greentree, A. D., and Spring, G. Wilksch, P. A. (2013). Linearisation of RGB camera responses for quantitative image analysis of visible and UV photography: A comparison of two techniques. *PLOS ONE*, 8(11):1–10. (↗ p. 21)

Garcia, J. E., Wilksch, P. A., Spring, G., Philp, P., and Dyer, A. (2014). Characterization of digital cameras for reflected ultraviolet photography; implications for qualitative and quantitative image analysis during forensic examination. *Journal of Forensic Sciences*, 59(1):117–122. (↗ p. 21)

Gershikov, E., Tzvika, L., and Kosolapov, S. (2013). Horizon line detection in marine images: Which method to choose? *International Journal on Advanves in Intelligent Systems*, 6(1):79–88. (↗ p. 17)

Gerstmayr-Hillen, L., Röben, F., Krzykawski, M., Kreft, S., Venjakob, D., and Möller, R. (2013). Dense topological maps and partial pose estimation for visual control of an autonomous cleaning robot. *Robotics and Autonomous Systems*, 61(5):497–516. (↗ pp. 8 and 150)

Gimbutas, Z. and Greengard, L. (2009). A fast and stable method for rotating spherical harmonic expansions. *Journal of Computational Physics*, 228(1):5621–5627. (↗ p. 71)

Goedemé, T., Tuytelaars, T., Van Gool, L. V., Vanacker, G., and Nuttin, M. (2005). Feature based omnidirectional sparse visual path following. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 1806–1811. IEEE. (↗ p. 7)

Goldhoorn, A., Ramisa, A., de Mántaras, R. D., and Toledo, R. (2007). Using the average landmark vector method for robot homing. *Frontiers in Artificial Intelligence and Applications*, 163(1):331–338. (↗ p. 9)

Goldstein, E. B. (2014). *Sensation and Perception*. Cengage Learning, Pacific Grove, California, 9th edition. (↗ pp. 15 and 21)

Goldstein, H., Poole, C. P., and Safko, J. L. (2012). *Klassische Mechanik*. Wiley, Hoboken, New Jersey, 3rd edition. (↗ p. 53)

González, D., Pérez, J., Milanés, V., and Nashashibi, F. (2016). A review of motion planning techniques for automated vehicles. *Transactions on Intelligent Transportation Systems*, 17(4):1135–1145. (↗ p. 5)

Gonzalez, R. C. and Woods, R. E. (1992). *Digital image processing*. Addison-Wesley Publishing Company, Reading, Pennsylvania, 1st edition. (↗ pp. 55 and 95)

Graham, P. (2010). Insect navigation. *Encyclopedia of Animal Behavior*, 2(1):167–175. (↗ p. 13)

Graham, P. and Cheng, K. (2009a). Ants use the panoramic skyline as a visual cue during navigation. *Current Biology*, 19(20):R935–R937. (↗ pp. 14 and 100)

Graham, P. and Cheng, K. (2009b). Which portion of the natural panorama is used for view-based navigation in the Australian desert ant? *Journal of Comparative Physiology A*, 195(7):681–689. (↗ p. 14)

Graham, P., Philippides, A., and Baddeley, B. (2010). Animal cognition: Multi-modal interactions in ant learning. *Current Biology*, 20(15):R639–R640. (↗ pp. 6 and 149)

Grant, R. H., Heisler, G. M., Gao, W., and Jenks, M. (2006). Ultraviolet leaf reflectance of common urban trees and the prediction of reflectance from leaf surface characteristics. *Agricultural and Forest Meteorology*, 120(1):127–139. (↗ p. 42)

Green, R. (2003). Spherical harmonic lighting: The gritty details. http://www.research.scea.com/gdc2003/spherical-harmonic-lighting.pdf. [Online; accessed 20-May-2015]. (↗ pp. 76 and 155)

Greiner, B. (2005). *Adaptations for nocturnal vision in insect apposition eyes*. PhD thesis, Lund University, Sweden. (↗ p. 15)

Gumbert, A., Kunze, J., and Chittka, L. (1999). Floral colour diversity in plant communities, bee colour space, and a null model. *Proceedings of the Royal Society B: Biological Sciences*, 266(1429):1711–1716. (↗ p. 15)

Hammer, B. and Villmann, T. (2002). Generalized relevance learning vector quantization. *Neural Networks*, 15(8):1059–1068. (↗ p. 113)

Hansen, P., Corke, P., and Boles, W. (2009). Wide-angle visual feature matching for outdoor localization. *International Journal of Robotics Research*, 29(2):267–297. (↗ p. 124)

Hanyk, L. (1999). *Viscoelastic response of the earth: Initial-value approach*. PhD thesis, Charles University, Czech Republic. (↗ p. 66)

Haralick, B. M., Lee, C.-N., Ottenberg, K., and Nölle, M. (1994). Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13(3):331–356. (↗ p. 3)

Heusser, D. and Wehner, R. (2002). The visual centering response in desert ants – *Cataglyphis fortis*. *Journal of Experimental Biology*, 205(5):585–590. (↗ p. 14)

Hewitt, E. and Ross, K. A. (1963). *Abstract harmonic analysis*, volume 1. Universitätsdruckerei H. Stürz AG, Würzburg, Germany, 1st edition. (↗ p. 52)

Hillen, L. (2013). *From local visual homing towards navigation of autonomous cleaning robots*. PhD thesis, Bielefeld University, Germany. (↗ pp. 4 and 5)

Homeier, H. and Steinborn, E. O. (1996). Some properties of the coupling coefficients of real spherical harmonics and their relation to Gaunt coefficients. *Journal of Molecular Structure: THEOCHEM*, 368(1):31–37. (↗ pp. 59 and 66)

Horridge, G. A. (2005). Recognition of a familiar place by the honeybee (*Apis mellifera*). *Journal of Comparative Physiology A*, 191(4):301–316. (↗ p. 13)

Horst, M. and Möller, R. (2017). Visual place recognition for autonomous mobile robots. *Robotics*, 6(2):1–40. (↗ p. 4)

Huynh, D. Q. (2009). Metrics for 3D rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164. (↗ p. 55)

Ishimaru, A. (1991). Wave propagation and scattering in random media and rough surfaces. In *Proceedings of the IEEE*, volume 79, pages 1359–1366. IEEE. (↗ p. 52)

Ivanic, J. and Ruedenberg, K. (1996). Rotation matrices for real spherical harmonics. Direct determination by recursion. *The Journal of Physical Chemistry*, 100(15):6342–6347. (↗ pp. 70 and 71)

Ivanic, J. and Ruedenberg, K. (1998). Additions and corrections – Rotation matrices for real spherical harmonics. Direct determination by recursion. *Journal of Physical Chemistry A*, 102(45):9099–9100. (↗ p. 71)

Judd, S. P. D. and Collett, T. S. (1998). Multiple stored views and landmark guidance in ants. *Nature*, 392(1):710–714. (↗ p. 14)

Judd, S. P. D., Dale, K., and Collet, T. S. (1999). On the fine structure of view based navigation in insects. In Golledge, R., editor, *Wayfinding behavior: Cognitive mapping and other spatial processes*, pages 229–258. The Johns Hopkins University Press, Baltimore, Maryland. (↗ p. 14)

Julle-Daniere, E., Schultheiss, P., Wystrach, A., Schwarz, S., Nooten, S. S., Bibost, A.-L., and Cheng, K. (2014). Visual matching in the orientation of desert ants (Melophorus bagoti): The effect of changing skyline height. *International Journal of Behavioural Biology*, 120(8):783–792. (↗ p. 14)

Kaas, H.-W., Mohr, D., Gao, P., Müller, N., Wee, D., Russell, H., Guan, M., Möller, T., Eckhard, G., Bray, G., Beicker, S., Brotschi, A., and Kohler, D. (2016). Automotive revolution – Perspective towards 2030. https://www.mckinsey.de/sites/mck_files/files/automotive_revolution_perspective_towards_2030.pdf. [Online; accessed 28-October-2016]. (↗ p. 1)

Kakarala, R. (1992). *Triple correlation on groups*. PhD thesis, University of California, USA. (↗ pp. 52 and 69)

Kakarala, R. and Mao, D. (2010). A theory of phase-sensitive rotation invariance with spherical harmonic and moment-based representations. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–112. IEEE. (↗ pp. 69 and 86)

Karami, E., Prasad, S., and Shehata, M. (2015). Image matching using SIFT, SURF, BRIEF and ORB: Performance comparison for distorted images. http://www.researchgate.net/publication/292157133. [Online; accessed 24-January-2016]. (↗ p. 124)

Kaula, W. M. (1966). *Theory of satellite geodesy*. Dover Publications, Providence, Rhode Island, 2nd edition. (↗ p. 51)

Kazhdan, M., Funkhouser, T., and Rusinkiewicz, S. (2003). Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Proceedings of the Eurographics Symposium on Geometry Processing (SGP)*, volume 6, pages 156–164. (↗ p. 86)

Kneip, L. and Furgale, P. (2014). OpenGV: A unified and generalized approach to real-time calibrated geometric vision. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE. (↗ p. 125)

Kollmeier, T., Röben, F., Schenck, W., and Möller, R. (2007). Spectral contrasts for landmark navigation. *Journal of the Optical Society of America A*, 24(1):1–10. (↗ pp. 15, 28, 31, 32, 33, 47, and 49)

Konolige, K. and Agrawal, M. (2008). FrameSLAM: From bundle adjustment to real-time visual mapping. *Transactions on Robotics*, 24(5):1066–1077. (↗ p. 2)

Kostelec, P. J. and Rockmore, D. N. (2008). FFTs on the rotation group. *Journal of Fourier Analysis and Applications*, 14(2):145–179. (↗ pp. 58 and 89)

Labbe, M. and Michaud, F. (2013). Appearance-based loop closure detection for online large-scale and long-term operation. *Transactions on Robotics*, 29(3):734–745. (↗ p. 2)

Lambrinos, D., Möller, R., Labhart, T., Pfeifer, R., and Wehner, R. (2000). A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems*, 30(1):39–64. (↗ p. 9)

Laughlin, S. B. (1989). The role of sensory adaptation in the retina. *The Journal of Experimental Biology*, 146(1):39–62. (↗ pp. 15 and 21)

Laughlin, S. B. (1994). Matching coding, circuits, cells, and molecules to signals: General principles of retinal design in the fly's eye. *Progress in Retinal and Eye Research*, 13(1):165–196. (↗ pp. 15 and 21)

Lee, J.-K. and Yoon, K.-J. (2015). Real-time joint estimation of camera orientation and vanishing points. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1866–1874.

IEEE. (↗ pp. 5, 114, and 124)

Lessig, C., de Witt, T., and Fiume, E. (2012). Efficient and accurate rotation of finite spherical harmonics expansions. *Journal of Computational Physics*, 231(2):243–250. (↗ p. 71)

Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011). BRISK: Binary robust invariant scalable keypoints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2548–2555. IEEE. (↗ p. 7)

Lourenço, M., Barreto, J. P., and Vasconcelos, F. (2012). sRD-SIFT: Keypoint detection and matching in images with radial distortion. *Transactions on Robotics*, 28(3):752–760. (↗ p. 124)

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157. IEEE. (↗ pp. 7 and 124)

Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P., and Milford, M. (2016). Visual place recognition: A survey. *Transactions on Robotics*, 32(1):1–19. (↗ pp. 3 and 99)

Maddern, W., Milford, M., and Wyeth, G. (2012). CAT-SLAM: Probabilistic localisation and mapping using a continuous appearance-based trajectory. *International Journal of Robotics Research*, 31(4):429–451. (↗ p. 4)

Maddern, W., Stewart, A., McManus, C., Upcroft, B., Churchill, W., and Newman, P. (2014). Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1–8. IEEE. (↗ p. 3)

Madl, T., Chen, K., Montaldi, D., and Trappl, R. (2015). Computational cognitive models of spatial memory in navigation space: A review. *Neural Networks*, 65(1):18–43. (↗ p. 13)

Mahajan, V. N. (1994). Zernike circle polynomials and optical aberrations of systems with circular pupils. *Applied optics*, 33(34):8121–8124. (↗ p. 148)

Makadia, A. and Daniilidis, K. (2003). Direct 3D-rotation estimation from spherical images via a generalized shift theorem. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 217–224. IEEE. (↗ p. 115)

Makadia, A., Geyer, C., and Daniilidis, K. (2007). Correspondenceless structure from motion. *International Journal of Computer Vision*, 75(3):311–327. (↗ p. 114)

Makadia, A., Patterson, A., and Daniilidis, K. (2006). Fully automatic registration of 3D point clouds. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 1297–1304. IEEE. (↗ pp. 51 and 133)

Makadia, A., Sorgi, L., and Daniilidis, K. (2004). Rotation estimation from spherical images. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, volume 3, pages 590–593. IEEE. (↗ p. 115)

Marinucci, D. and Peccati, G. (2011). *Random fields on the sphere*. Cambridge University Press, Cambridge, Massachusetts, 1st edition. (↗ pp. 52, 57, 58, 59, 61, 62, and 152)

Marinucci, D. and Piccioni, M. (2004). The empirical process on Gaussian spherical harmonics. *The Annals of Statistics*, 32(3):1261–1288. (↗ p. 52)

Maslen, D. K. (1996). Generalized FFTs – A survey of some recent results. Technical Report, Max-Planck-Institut for Mathematics. (↗ p. 88)

MATLAB (2012). *Version 8.0.0 (R2012b)*. The MathWorks Inc., Natick, Massachusetts, 1st edition. (↗ pp. 93 and 166)

Matthies, L., Xiong, Y., Hogg, R., Zhu, D., Rankin, A., Kennedy, B., Hebert, M., Maclachlan, R., Won, C., Frost, T., et al. (2002). A portable, autonomous, urban reconnaissance robot. *Robotics and Autonomous Systems*, 40(2):163–172. (↗ p. 99)

McManus, C., Upcroft, B., and Newman, P. (2014). Scene signatures: Localised and point-less features for localisation. In *Proceedings of Robotics: Science and Systems (RSS)*, pages 1–9, Berkeley, California. (↗ pp. 4 and 6)

Meguro, J.-I., Murata, T., Amano, Y., Hasizume, T., and Takiguchi, J.-I. (2008). Development of a positioning technique for an urban area using omnidirectional infrared camera and aerial survey data. *Advanced Robotics*, 22(6–7):731–747. (↗ p. 113)

Mendel, J. M. (1991). Tutorial on higher-order statistics (spectra) in signal processing and system theory: Theoretical results and some applications. *Proceedings of the IEEE*, 79(3):278–305. (↗ p. 56)

Menegatti, E., Maeda, T., and Ishiguro, H. (2004). Image-based memory for robot navigation using properties of omnidirectional images. *Robotics and Autonomous Systems*, 47(4):251–267. (↗ pp. 3 and 4)

Menzel, R. and Backhaus, W. (1991). Colour vision in insects. In Gouras, P., editor, *The Perception of Colour Vision and Visual Dysfunction*, pages 262–288. Macmillan Publishers, Oxford, England. (↗ p. 15)

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Van Gool, L. (2005). A comparison of affine region detectors. *International journal of computer vision*, 65(1-2):43–72. (↗ p. 132)

Milford, M. (2013). Vision-based place recognition: How low can you go? *International Journal of Robotics Research*, 32(7):766–789. (↗ pp. 4 and 148)

Milford, M. and Wyeth, G. (2010). Persistent navigation and mapping using a biologically inspired SLAM system. *International Journal of Robotics Research*, 29(9):1131–1151. (↗ p. 9)

Milford, M. and Wyeth, G. F. (2012). SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 1643–1649. IEEE. (↗ pp. 3, 95, 100, 101, 103, 104, and 146)

Mohlenkamp, M. J. (2016). A user's guide to spherical harmonics. http://www.ohio.edu/people/mohlenka/research/uguide.pdf. [Online; accessed 24-November-2016]. (↗ p. 65)

Möller, R. (2002). Insects could exploit UV-green contrast for landmark navigation. *Journal of Theoretical Biology*, 214(4):619–631. (↗ pp. 15, 18, 21, 28, 31, 32, 33, 46, 47, and 100)

Möller, R. (2009). Local visual homing by warping of two-dimensional images. *Robotics and Autonomous Systems*, 57(1):87–101. (↗ p. 135)

Möller, R. (2012). A model of ant navigation based on visual prediction. *Journal of Theoretical Biology*, 305(1):118–130. (↗ pp. 9 and 148)

Möller, R. (2016a). Column distance measures and their effect on illumination tolerance in MinWarping. Technical report, Bielefeld University, Faculty of Technology, Computer Engineering Group. (↗ p. 5)

Möller, R. (2016b). A SIMD implementation of the minWarping method for local visual homing. Technical report, Bielefeld University, Faculty of Technology, Computer Engineering Group. (↗ p. 136)

Möller, R., Horst, M., and Fleer, D. (2014). Illumination tolerance for visual navigation with the holistic min-warping method. *Robotics*, 3(1):22–67. (↗ pp. 5, 95, and 136)

Möller, R., Krzykawski, M., and Gerstmayr, L. (2010). Three 2D-warping schemes for visual robot navigation. *Autonomous Robots*, 29(3):253–291. (↗ pp. 3, 6, 12, 18, 114, 135, 136, and 146)

Möller, R., Krzykawski, M., Gerstmayr-Hillen, L., Horst, M., Fleer, D., and De Jong, J. (2013). Cleaning robot navigation using panoramic views and particle clouds as landmarks. *Robotics and Autonomous Systems*, 61(12):1415–1439. (↗ pp. 9 and 150)

Möller, R. and Vardy, A. (2006). Local visual homing by matched-filter descent in image distances. *Biological Cybernetics*, 95(5):413–430. (↗ p. 6)

Moore, S. (2008). SpharmonicKit v2.7. http://www.cs.dartmouth.edu/~geelong/sphere/. [Online; accessed 02-March-2015]. (↗ pp. 87 and 116)

Mote, M. I. and Wehner, R. (1980). Functional characteristics of photoreceptors in the compound eye and ocellus of the desert ant, *Cataglyphis bicolor*. *Journal of Comparative Physiology A*, 137(1):63–71. (↗ pp. 15, 19, 49, and 100)

Mount, J. and Milford, M. (2016). 2D visual place recognition for domestic service robots at night. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 4822–4829. IEEE. (↗ p. 4)

Muja, M. and Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 331–340. (↗ p. 6)

Muja, M. and Lowe, D. G. (2014). Scalable nearest neighbor algorithms for high dimensional data. *Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240. (↗ p. 132)

Nann, S. and Riordan, C. (1991). Solar spectral irradiance under clear and cloudy skies: Measurements and a semiempirical model. *American Meteorology Society*, 30(4):447–462. (↗ p. 14)

Narendra, A., Gourmaud, S., and Zeil, J. (2013). Mapping the navigational knowledge of individually foraging ants, Myrmecia croslandi. *Proceedings of the Royal Society of London B: Biological Sciences*, 280(1765):1–9. (↗ p. 149)

Neto, A. M., Victorino, A. C., Fantoni, I., and Zampieri, D. E. (2011). Robust horizon finding algorithm for real-time autonomous navigation based on monocular vision. In *Proceedings of the International Conference on Intelligent Transportation Systems (ITSC)*, pages 532–537. IEEE. (↗ p. 17)

Nistér, D., Naroditsky, O., and Bergen, J. (2006). Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20. (↗ pp. 2 and 3)

Nowrouzezahrai, D., Simari, P., and Fiume, E. (2012). Sparse zonal harmonic factorization for efficient SH rotation. *Transactions on Graphics*, 31(3):23:1–23:9. (↗ p. 71)

O'Keefe, J. (1976). Place units in the hippocampus of the freely moving rat. *Experimental neurology*, 51(1):78–109. (↗ p. 9)

Otsu, N. (1979). A threshold selection method from gray-level histograms. *Transactions on Systems, Man, and Cybernetics*, 9(1):62–66. (↗ pp. 25 and 27)

Parker, J. R. (1997). *Algorithms for image processing and computer vision*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2nd edition. (↗ p. 95)

Pepperell, E., Corke, P., and Milford, M. (2014). All-environment visual place recognition with SMART. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 1612–1618. IEEE. (↗ p. 147)

Prassler, E., Munich, M. E., Pirjanian, P., and Kosuge, K. (2016). Domestic robotics. In Siciliano, B. and Khatib, O., editors, *Handbook of Robotics*, pages 1729–1758. Springer, Berlin, Germany. (↗ p. 1)

Pratt, S. C., Brooks, S. E., and Franks, N. R. (2001). The use of edges in visual navigation by the ant *Leptothorax albipennis*. *Ethology*, 107(12):1125—1136. (↗ p. 14)

Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (1992). *Numerical recipes in C: The art of scientific computing*. Cambridge University Press, Cambridge, England, 2nd edition. (↗ p. 70)

Ren, S., Cao, X., Wei, Y., and Sun, J. (2014). Face alignment at 3000 fps via regressing local binary features. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1685–1692. IEEE. (↗ p. 2)

Rosengren, R. and Fortelius, W. (1986). Ortstreue in foraging ants of the *Formica rufa* group - Hierarchy of orienting cues and long-term memory. *Insectes Sociaux*, 33(3):306–377. (↗ p. 14)

Rossel, S. and Wehner, R. (1984). Celestial orientation in bees: The use of spectral cues. *Journal of Comparative Physiology A*, 155(5):605–613. (↗ p. 43)

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 430–443. Springer. (↗ p. 7)

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE. (↗ pp. 7 and 124)

Ruderman, D. L. and Bialek, W. (1994). Statistics of natural images: Scaling in the woods. *Physical review letters*, 73(6):814–818. (↗ p. 94)

Rudnicki-Bujnowski, G. (1975). Explicit formulas for Clebsch-Gordan coefficients. *Computer Physics Communications*, 10(4):245–250. (↗ p. 59)

Scaramuzza, D., Martinelli, A., and Siegwart, R. (2006). A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *Proceedings of the International Conference on Computer Vision Systems (ICVS)*, pages 45–45. IEEE. (↗ p. 11)

Scaramuzza, D. and Siegwart, R. (2008). Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE Transactions on robotics*, 24(5):1015–1026. (↗ p. 114)

Schmidt, J. (2014). Wetterkontor GmbH. http://www.wetterkontor.de/. [Online; accessed 14-Sept-2014]. (↗ p. 20)

Schultheiss, P., Wystrach, A., Schwarz, S., Tack, A., Delor, J., Nooten, S. S., Bibost, A.-L., Freas, C. A., and Cheng, K. (2016). Crucial role of ultraviolet light for desert ants in determining direction from the terrestrial panorama. *Animal Behaviour*, 115(1):19–28. (↗ p. 15)

Schwarz, S., Julle-Daniere, E., Morin, L., Schultheiss, P., Wystrach, A., Ives, J., and Cheng, K. (2014). Desert ants (*Melophorus bagoti*) navigating with robustness to distortions of the natural panorama. *Insect Sociaux*, 61(4):371–383. (↗ pp. 14 and 46)

Seidl, R. (1982). *Die Sehfelder und Ommatidien-Divergenzwinkel von Arbeiterin, Königin und Drohne der Honigbiene (Apis mellifica)*. PhD thesis, Technische Hochschule Darmstadt, Germany. (↗ p. 89)

Sgavetti, M., Pompilio, L., and Meli, S. (2006). Reflectance spectroscopy (0.3-2.5 μm) at various scales for bulk-rock identification. *Geosphere*, 2(3):142–160. (↗ pp. 15 and 42)

Shen, L., Farid, H., and McPeek, M. A. (2009). Modeling three-dimensional morphological structures using spherical harmonics. *Evolution*, 63(4):1003–1016. (↗ pp. 51 and 133)

Shen, Y. and Wang, Q. (2013). Sky region detection in a single image for autonomous ground robot navigation. *International Journal of Advanced Robotic Systems*, 10(10):362–375. (↗ p. 17)

Simard, P. Y., LeCun, Y. A., Denker, J. S., and Victorri, B. (1998). Transformation invariance in pattern recognition – Tangent distance and tangent propagation. In Orr, G. B. and Müller, K.-R., editors, *Neural networks: Tricks of the trade*, pages 239–274. Springer, Berlin, Germany. (↗ pp. 3, 95, and 97)

Sloan, P.-P., Kautz, J., and Snyder, J. (2002). Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 527–536. (↗ p. 51)

Stafford, S., Hillebrand, R., and Hauschild, H. (2004). *The new Nikon compendium: Cameras, lenses & accessories since 1917*. Lark Books, 1st edition. (↗ p. 11)

Stewenius, H., Engels, C., and Nistér, D. (2006). Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284–294. (↗ pp. 6 and 124)

Stone, T., Differt, D., Milford, M., and Webb, B. (2016). Skyline-based localisation for aggressively manoeuvring robots using UV sensors and spherical harmonics. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 5615–5622. IEEE. (↗ pp. 17, 18, 52, 99, and 112)

Stone, T., Mangan, M., Ardin, P., and Webb, B. (2014). Sky segmentation with ultraviolet images can be used for navigation. In *Proceedings of Robotics: Science and Systems (RSS)*. www.roboticsproceedings.org. (↗ pp. 17 and 48)

Straub, W. O. (2014). Efficient computation of Clebsch-Gordan coefficients. http://vixra.org/abs/1403.0263. [Online; accessed 28-April-2015]. (↗ pp. 59, 60, and 167)

Strübbe, S., Stürzl, W., and Egelhaaf, M. (2015). Insect-inspired self-motion estimation with dense flow fields — an adaptive matched filter approach. *PLOS ONE*, 10(8):1–35. (↗ p. 9)

Stürzl, W., Grixa, I., Mair, E., Narendra, A., and Zeil, J. (2015). Three-dimensional models of natural environments and the mapping of navigational information. *Journal of Comparative Physiology A*, 201(6):563–584. (↗ p. 187)

Stürzl, W. and Mallot, H. A. (2006). Efficient visual homing based on Fourier transformed panoramic images. *Robotics and Autonomous Systems*, 54(4):300–313. (↗ pp. 9, 115, and 149)

Stürzl, W. and Zeil, J. (2007). Depth, contrast and view-based homing in outdoor scenes. *Biological Cybernetics*, 96(5):519–531. (↗ p. 95)

Sun, D., Roth, S., and Black, M. (2014). A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137. (↗ p. 114)

Sünderhauf, N. and Protzel, P. (2011). BRIEF-Gist-Closing the loop by simple means. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 1234–1241. IEEE/RSJ. (↗ p. 8)

Tamgade, S. N. and Bora, V. R. (2009). Motion vector estimation of video image by pyramidal implementation of Lucas Kanade optical flow. In *Proceedings of the International Conference on Emerging Trends in Engineering & Technology (ICETE)*, pages 914–917. IEEE. (↗ pp. 5, 114, and 124)

Tardif, J.-P., Pavlidis, Y., and Daniilidis, K. (2008). Monocular visual odometry in urban environments

using an omnidirectional camera. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 2531–2538. IEEE. (↗ p. 2)

Tehrani, M. H., Garratt, M., and Anavatti, S. (2012a). Gyroscope offset estimation using panoramic vision-based attitude estimation and extended Kalman filter. In *Proceedings of the International Conference on Communications, Computing, and Control Applications (CCCA)*, pages 1–5. IEEE. (↗ p. 17)

Tehrani, M. H., Garratt, M. A., and Anavatti, S. (2012b). Horizon-based attitude estimation from a panoramic vision sensor. *IFAC Proceedings Volumes*, 45(1):185–188. (↗ p. 17)

The Robot Report (2016). Global and iRobot floor cleaning market. http://www.therobotreport.com/news/global-and-irobot-floor-cleaning-robots-market. [Online; accessed 05-January-2017]. (↗ p. 1)

Valgren, C. and Lilienthal, A. J. (2007). SIFT, SURF and seasons: Long-term outdoor localization using local features. In *Proceedings of European Conference on Mobile Robots (ECMR)*, pages 1–6. (↗ p. 132)

van der Schaaf, A. and van Hateren, J. H. (1996). Modelling the power spectra of natural images: Statistics and information. *Vision Research*, 36(17):2759–2770. (↗ p. 94)

van Gelderen, M. (1998). The shift operators and translations of spherical harmonics. *DEOS Progress Letter*, 98(1):57–67. (↗ p. 78)

Wallach, H. M. (2006). Topic modeling: Beyond bag-of-words. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 977–984. ACM. (↗ p. 104)

Wang, C.-C., Thorpe, C., and Thrun, S. (2003). Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, volume 1, pages 842–849. IEEE. (↗ p. 3)

Wang, J., Xu, K., Zhou, K., Lin, S., Hu, S., and Guo, B. (2006). Spherical harmonics scaling. *International Journal of Computer Graphics*, 22(9):713–720. (↗ pp. 80 and 82)

Warrant, E. J. (2006). Invertebrate vision in dim light. In Warrant, E. J. and Nilsson, D.-E., editors, *Invertebrate vision*, pages 83–126. Cambridge University Press, Cambridge, England. (↗ p. 15)

Weber, K., Venkatesh, M., and Srinivasan, M. (1999). Insect-inspired robotic homing. *Adaptive Behavior*, 7(1):65–97. (↗ p. 7)

Wehner, R. (1982). Himmelsnavigation bei Insekten. *Neujahrsblatt der Naturforschenden Gesellschaft Zürich*, 184:1–132. (↗ pp. 15 and 48)

Wehner, R., Michel, B., and Antonsen, P. (1996). Visual navigation in insects: Coupling of egocentric and geocentric information. *Journal of Experimental Biology*, 199(1):129–140. (↗ pp. 13 and 14)

Wehner, R. and Räber, F. (1979). Visual spatial memory in desert ants *Cataglyphis bicolor (Hymenoptera: Formicidae). Experientia*, 35(12):1569–1571. (↗ p. 14)

Weiß, G., Wetzler, C., and Von Puttkamer, E. (1994). Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 595–601. IEEE/SJ/GI. (↗ p. 7)

Werner, F., Sitte, J., and Maire, F. D. (2007). Automatic place determination using colour histograms and self-organising maps. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, pages 21–24. (↗ p. 4)

Wieczorek, M. (2015). SHTOOLS v3.1. http://shtools.ipgp.fr/. [Online; accessed 02-March-2015]. (↗ pp. 87 and 116)

Wolf, H. (2011). Review: Odometry and insect navigation. *Journal of Experimental Biology*, 214(4):1629–1641. (↗ p. 13)

Wu, J., Cui, Z., Sheng, V. S., Zhao, P., Su, D., and Gong, S. (2013). A comparative study of SIFT and its variants. *Measurement Science Review*, 13(3):122–131. (↗ pp. 124 and 132)

Wystrach, A., Beugnon, G., and Cheng, K. (2011). Landmarks or panoramas: What do navigating ants attend to for guidance? *Frontiers in Zoology*, 8(21):1–11. (↗ p. 14)

Wystrach, A. and Graham, P. (2012). What can we learn from studies of insect navigation? *Animal Behaviour*, 84(1):13–20. (↗ p. 13)

Yang, J., Chung, S.-J., Hutchinson, S., Johnson, D., and Kise, M. (2015). Omnidirectional-vision-based estimation for containment detection of a robotic mower. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 6344–6351. IEEE. (↗ p. 99)

Yang, K., Gao, S., Li, C., and Li, Y. (2013). Efficient color boundary detection with color-opponent mechanisms. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2810–2817. IEEE. (↗ p. 49)

Zeil, J. (2012). Visual homing: An insect perspective. *Current Opinion in Neurobiology*, 22(2):285–293. (↗ pp. 9 and 13)

Zeil, J., Hofmann, M. I., and Chahl, J. S. (2003). Catchment areas of panoramic snapshots in outdoor scenes. *Journal of the Optical Society of America A*, 20(3):450–469. (↗ pp. 9, 12, 114, 119, and 146)

Zernike, v. F. (1934). Beugungstheorie des Schneidenverfahrens und seiner verbesserten Form, der Phasenkontrastmethode. *Physica*, 1(7-12):689–704. (↗ p. 148)

Zhou, C., Wei, Y., and Tan, T. (2003). Mobile robot self-localization based on global visual appearance features. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, volume 1, pages 1271–1276. IEEE. (↗ p. 4)

Zhu, H., Chan, F. H. Y., and Lam, F. K. (1999). Image contrast enhancement by constrained local histogram equalization. *Computer Vision and Image Understanding*, 73(2):281–290. (↗ p. 95)

Zsedrovits, T., Bauer, P., Zarandy, A., Vanek, B., Bokor, J., and Roska, T. (2014). Error analysis of algorithms for camera rotation calculation in GPS/IMU/camera fusion for UAV sense and avoid systems. In *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 864–875. IEEE. (↗ p. 3)

Zuo, L., Humbert, M., and Esling, C. (1993). An effective algorithm for calculation of the Clebsch-Gordan coefficients. *Journal of Applied Crystallography*, 26(2):302–304. (↗ p. 59)