Learning and Detecting Objects in Combined Range and Color Images based on Local Feature Frames

Jens Kubacki

Submitted: March 2010
Published: April 2012

Learning and Detecting Objects in Combined Range and Color Images based on Local Feature Frames

Dissertation zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

der Technischen Fakultät der Universität Bielefeld
vorgelegt von

Jens Kubacki

12. März 2010

Acknowledgments

The work presented in this thesis would not have been possible without the support of many people who have contributed to my professional and personal development in very different ways throughout the past several years.

On the professional side, I have cooperated with researchers from various European research institutes, while carrying out my research work within the European project Cogniron and the German project Desire. I thank all the members of these past projects who interacted with me in one way or another. Also I would like to thank my colleagues at Fraunhofer IPA, especially Andreas Pott, Winfried Baum and Birgit Graf. A person that I am deeply grateful to is Gerhard Sagerer, who initiated this thesis. I am also grateful to Franz Kummert and Erwin Prassler, who volunteered as reviewers of this work.

On the personal side, I would like to thank my wife Lana and my parents who gave me their support throughout this time. Apologies go to my son Jan Lucas who, at four years of age, does not understand why a father works on such documents when he should be playing with his son.

Abstract

In this thesis an approach to single object detection is presented, aimed at use in the context of service robots. The approach integrates suggestions for sample data acquisition for real robots in typical environments and efficient algorithms for model construction and detection. A novelty is that the early fusion of 3D and color data per image pixel is explored in detail. In order to approach the challenge of sample data acquisition, this fusion is used to provide segmented training views using 3D masking shapes for segmentation. Combined with powerful feature points that have been published elsewhere in the context of 2D vision, a new data representation is derived. This representation, called *feature frame cloud*, is similar to the well-known point cloud, but makes full use of the information delivered with 3D and color data. Instead of points being the basic elements, here full coordinate frames are used. These can be computed from the 2D feature point gradient, the 3D position, and the 3D surface normal. Feature frames of object models can be described by their relative position and orientation to an intrinsic object frame. The frames can be accessed efficiently by the use of low-dimensional discrete keys consisting of stable attributes. It is shown in this work, that the complexity for frame estimation using this new data structure is low. Only two matching correspondences are required to fully determine a frame. Efficient algorithms are presented for object model construction and detection. The collection of training data in real-world environments is addressed explicitly. Measurements, tests and demonstrations in the evaluation part of the document show that the ideas presented are suited for the use on real robots. The low search complexity to estimate a single object pose hypothesis, the robustness against large fractions of background in the scene data and against object coverage, combined with the learning ability, are seen as key advantages of this approach.

Contents

List of Figures

List of Tables

Acronyms

CAD	computer aided design
CCD	charge-coupled device
DCM	direction cosine matrix
DOF	degrees-of-freedom
FFC	feature frame cloud
GPS	global positioning system
GPU	graphics processing unit
ICP	iterative closest points algorithm
PCL	point cloud library
RANSAC	random sample consensus algorithm
SIFT	scale invariant feature transform
SURF	speeded-up robust features
SVD	singular value decomposition
SVM	support vector machine
TOF	time-of-flight

Notation

In the following some of the symbols that are used in this document are listed arranged in order of the related chapters. In the appendix of this document the conventions are given.

$\mathbb{R}$	real numbers
$\mathbb{R}^3$	representation of the 3D Euclidean space
$\mathbb{F} \subset \mathbb{R}^{3\times4}$	frame space, three unit lengths orthogonal orientation vectors plus offset vector in $\mathbb{R}^3$
$W \in \mathbb{F}$	world frame
$R \in \mathbb{F}$	robot frame
$E \in \mathbb{F}$	end effector frame
$S \in \mathbb{F}$	sensor frame (in this document usually set to S=W)
$O \in \mathbb{F}$	object frame
$D \in \mathbb{F}$	detected frame (the object frame relative to the sensor frame)

$\mathcal{S} \in \mathbb{R}^{w\times h\times3\times3}$	shared image
$\mathcal{P} \in \mathbb{R}^{w\times h\times3}$	coordinate image of the shared image
$\mathcal{C} \in \mathbb{R}^{w\times h\times3}$	color image of the shared image
$w \in \mathbb{N}$	width of the shared image
$h \in \mathbb{N}$	height of the shared image
$\boldsymbol{u} = (u, v)^T \in \mathbb{N}^2$	image coordinates in the range image and shared image
$\boldsymbol{p}_{u,v} \in \mathbb{R}^3$	shared image point 3D coordinates in the sensor system S
$\mathbb{C} \subseteq \mathbb{R}^3$	color space containing red, green and blue values
$\boldsymbol{c}_{u,v} \in \mathbb{C}$	color values of a pixel position in the shared image
${}^C\boldsymbol{p}_{u,v}$	coordinates in the color imaging sensor system C at $(u, v)^T$
$\boldsymbol{T} \in \mathbb{R}^{w\times h}$	range image containing the time-of-flight values (phase shifts) per pixel

$t_{u,v} \in \mathbb{R}$	a single time-of-flight value at a pixel position in the range image
$\mathcal{C}' \in \mathbb{R}^{w_c \times h_c \times 3}$	the raw color image from the color imaging sensor
$w_c \in \mathbb{N}$	width of the raw color image
$h_c \in \mathbb{N}$	height of the raw color image
$\boldsymbol{u}_c = (u_c, v_c)^T \in \mathbb{N}^2$	image coordinates in the raw color image
$\boldsymbol{c}'_{u_c,v_c} \in \mathbb{R}^3$	color value in the raw color image
$d_{u,v} \in \mathbb{R}$	Euclidean distance of the shared image coordinates at a pixel location
$\lvert \cdot \rvert$	absolute value (Euclidean norm) of a vector
$d \in \mathbb{R}$	Euclidean distance of a point
$f_x, f_y \in \mathbb{R}$	the focal lengths of the range imaging sensor in pixel units
$c_x, c_y \in \mathbb{R}$	image plane center of the range imaging sensor
$k_1, k_2, k_3 \in \mathbb{R}$	radial distortion coefficients of the range imaging sensor
$p_1, p_2 \in \mathbb{R}$	tangential distortion coefficients of the range imaging sensor
$rnd : \mathbb{R} \to \mathbb{N}$	rounding function
$\boldsymbol{x}' \in \mathbb{R}^2$	projected point on the ideal image plane
$\boldsymbol{x}'' \in \mathbb{R}^2$	corrected point on the ideal image plane
$\boldsymbol{R}_C$	rotation matrix from S to C
$\boldsymbol{o}_C$	translation vector, base point of C in S
$f_{x\,c}, f_{y\,c} \in \mathbb{R}$	scaling parameters of the color imaging sensor
$c_{x\,c}, c_{y\,c} \in \mathbb{R}$	image plane center of the color imaging sensor
$k_{1\,c}, k_{2\,c}, k_{3\,c} \in \mathbb{R}$	radial distortion coefficients of the color imaging sensor
$p_{1\,c}, p_{2\,c} \in \mathbb{R}$	tangential distortion coefficients of the color imaging sensor
$d'_{u,v} \in \mathbb{R}$	intrinsic divisor to compute the orthogonal (z) distance from the Euclidean distance
$k, l, m \in \mathbb{R}$	parameters for the distance computation
$\leftarrow$	value assignment in pseudo codes
$\boldsymbol{Z}_{buf} \in \mathbb{R}^{w_c \times h_c}$	z-buffer storage
$\boldsymbol{z}_{max} \in \mathbb{R}$	maximal possible z-value
$\mathcal{U} \in \mathbb{R}^{w_c \times h_c \times 2}$	pixel position storage of the closest points so far
$\boldsymbol{I} \in \mathbb{R}^{w \times h}$	gray-level intensity image of the range imaging sensor

$\boldsymbol{I}_c \in \mathbb{R}^{w_c \times h_c}$	gray-level intensity image of the color imaging sensor
$\mathrm{O} \in \mathbb{R}^{3 \times k \times l}$	point set of the calibration rig coordinates
$\mathrm{I} \in \mathbb{R}^{2 \times k \times l}$	2D image point coordinates in the intensity image of the range imaging sensor
$\mathrm{I}_c \in \mathbb{R}^{2 \times k \times l}$	2D image point coordinates in the gray-level image of the raw color image
$\boldsymbol{T} \in \mathbb{R}^{w \times h}$	image containing the time-of-flight measures
$\mathrm{I}' \in \mathbb{R}^{2 \times k \times l}$	corrected 2D image point coordinates
$\mathrm{T} \in \mathbb{R}^{k \times l}$	set of time-of-flight values
$\mathrm{D} \in \mathbb{R}^{k \times l}$	set of distances
$\mathrm{P} \in \mathbb{R}^{3 \times k \times l}$	set of points computed by using the extrinsic calibration frames
$\mathrm{P}' \in \mathbb{R}^{3 \times k \times l}$	set of points computed by using the transformation equations
$E_n \in \mathbb{F}$	extrinsic coordinate frame of the nth view
$L \in \mathbb{F}$	learning frame
$\mathcal{S}_{seg} \in \mathbb{R}^{w \times h \times 3}$	segmented training image of an object view
S_{train}	training set of shared images
$s \in \mathbb{R}$	feature point scale measure
$\phi \in [0, .., 2\pi] \subset \mathbb{R}$	feature point 2D orientation
$l \in \mathbb{N}$	descriptor vector dimensionality
$\boldsymbol{d} \in \mathbb{R}^l$	descriptor vector of a feature point
F	feature frame cloud as enumerated set
$m \in \mathbb{N}$	number of different keys in a feature frame cloud
$n_i \in \mathbb{N}$	number of frames associated with the ith key of a feature frame cloud
$\mathcal{F}$	feature point tuple
$\boldsymbol{k} \in \mathbb{N}^4$	discrete description key
$i \in \{0, 1, 2\}$	image channel identifier or counting variable
$r \in \mathbb{N}$	discrete feature point radius
$l \in \{0, 1\}$	Laplacian identifier
$c \in \mathbb{N}^{n_c}$	cluster identifier, $n_c \in \mathbb{N}$ is the number of clusters
$\lfloor \cdot \rfloor$	floor operation of type $\mathbb{R} \rightarrow \mathbb{R}$
F_{train}	training set of feature frame clouds
$\boldsymbol{k}' \in \mathbb{N}^3$	description key prefix

$s' \in \mathbb{N}$	discrete size label of a feature point
G	argument feature frame cloud
$D_{\boldsymbol{k}'} \in \mathbb{R}^{l \times n}$	descriptor partition of size n associated with the key prefix $\boldsymbol{k}'$
$c_{\boldsymbol{k}'}(\boldsymbol{d}) \in \mathbb{N}$	function to retrieve the cluster identifier for descriptor $\boldsymbol{d}$ at the key prefix $\boldsymbol{k}'$
$n_c \in \mathbb{N}$	maximal cluster identifier
$t_{cls} \in \mathbb{R}$	threshold controlling the descriptor clustering
$\boldsymbol{n}_{u,v} \in \mathbb{R}^3$	surface normal at shared image position $(u, v,)^T$
$\boldsymbol{g} \in \mathbb{R}^3$	gradient vector
$\boldsymbol{h} \in \mathbb{R}^3$	orthogonal axis vector
$H_i \in \mathbb{F}$	ith hypothesis frame
$\mathrm{H} \in \mathbb{F}^n$	set n of frame hypotheses
$d_F : \mathbb{F} \times \mathbb{F} \to \mathbb{R}$	frame distance function
$t_F \in \mathbb{R}$	frame threshold
$c(H_i) : \mathbb{F} \to \mathbb{N}$	count of frames in the vicinity of t_F for the ith hypothesis H_i
H^*	best hypothesis frame
$\mathrm{H}^* \subset \mathrm{H}$	subset of H containing the hypotheses of the voting cube with highest count
$\boldsymbol{u}_F \in \mathbb{F}$	rotation axis of a frame F
$\alpha \in [0, ..\pi] \subset \mathbb{R}$	rotation angle of a frame F
$\theta \in [0, ..2\,\pi] \subset \mathbb{R}$	first component of the sphere coordinates of $\boldsymbol{u}_F$
$\phi \in [-\pi/20, .., \pi/2) \subset \mathbb{R}$	second component of the sphere coordinates of $\boldsymbol{u}_F$
M	object model feature frame cloud
S	scene feature frame cloud
$t_{surf} \in \mathbb{R}$	Hessian threshold used for SURF features
$\mathbb{R}_+$	positive reals
$e_d \in \mathbb{R}_+$	descriptor error rate
$n_{fa} \in \mathbb{N}$	number of false alarms
$n_m \in \mathbb{N}$	number of misses
$n_{neg} \in \mathbb{N}$	number of feature points outside the object region
$n_{pos} \in \mathbb{N}$	number of feature points inside the object region
$e_F \in \mathbb{R}$	frame error rate
$h \in \mathbb{R}$	hit rate
$\hat{D}$	approximated detection frame D

1 Introduction

In this chapter the context and problem that motivate this work are introduced. First, applications that can profit from sophisticated perception abilities are presented. *Service robots* are chosen to be of primary interest in this work. Then, the object detection problem is introduced as the subject of this thesis. Finally, the ideas of this work are summarized and an outline of the document is given.

1.1 Motivation for Vision

1.1.1 Pros and Cons for Vision

Today, we are surrounded by technical devices that contain mechanical, electronic and software components. Computers and devices such as mobile phones have become our permanent companions. It is expected that this trend will continue and that there will be more applications for such systems to be found in the future.

Vision systems are enablers for future applications since they can be used for a variety of perception tasks. Many applications in the modern technical world already incorporate vision systems and there will certainly be more use cases in the future. Some of the attractions of using vision are:

- Imaging sensors deliver a huge amount of information in one image frame.
- Not only is the high spatial resolution attractive, but also the time for frame retrieval is constantly decreased with the introduction of new products.
- The hardware on which vision algorithms run is constantly being accelerated and special purpose hardware speeds up processing. Also, special purpose hardware from other domains, such as *graphics processing units* (GPUs), can be used for faster computation.
- The images capture a large part of the physical environment, i.e. the *field-of-view* is large compared to other (e.g. single point) sensors.
- There are already many vision algorithms available in commercial and open

source software libraries that can be used to cover new problem domains.

However, there are also some drawbacks to vision:

- Many current algorithms in vision are not stable enough to provide the reliability needed in e.g. industrial or domestic domains.
- A vision system can be too expensive. For instance, many vacuum cleaning robots that are currently on the market do not incorporate a vision system due to high cost. Although vision sensors can be quite cheap, the processing requirements may quickly exceed those of the embedded micro controller systems integrated in such products.
- Many problems that e.g. human or animal vision systems can solve are not yet solved in computer vision. Furthermore, some problems can be better solved with specialized sensors. For instance, large area position estimation can be solved by using a *global positioning system* (GPS).

1.1.2 Vision Applications

One new technical device that could play an important role in the near future is the service robot [68]. These are devices aimed at helping humans in their daily life. Ideally they operate reliably and safely in typical household or other domestic environments to the satisfaction of their user. Typical tasks they perform may involve the physical manipulation of objects. A distinguishing attribute of the future service robot from those available today (e.g. robot vacuum cleaners) is that they will be multi-functional. Necessary capabilities are to provide many different services, learn new skills, and interact closely with the *environment*. The environment may include humans that want to interact with the robot. Furthermore, the environment contains objects such as tools that the robot could manipulate or use. In this sense the idea of such a multi-purpose robot can be seen as a first attempt to realize the idea of the *first-generation universal robot* defined in [51]. The author optimistically dated the arrival of first-generation universal robots at around 2010. In a number of projects (e.g. [59]) researchers are actively developing control components of these robots. One output of these developments is the Care-O-bot® 3 robotic hardware platform ([24] and [61]) shown in figure 1.1 (left). Another example is the technology platform of the project Desire [59], shown in figure 1.1 (right). The home robot league of the Robocup competition series [63] provides more examples.

The *hardware* of the robot consists of sensors, actuators and computation components. The robot needs sensors in order to receive what happens in its environment and actuators (e.g. motors or speakers) to change the environmental state. The actual design of the hardware depends on the task domain (e.g. needed manipulative

Figure 1.1: Two examples of service robot research platforms. On the left the Care-O-bot® 3 is shown and on the right a newer platform that includes two manipulator arms. Both robots are equipped with various sensors and actuators and were used during the work related to this document.

and perceptual capacities), type of the environment, and available sensor-actuator and computation technology.

One may split the overall *software* of the robot roughly into the basic installations (operating system, drivers for sensors and actuators, libraries, etc) and those components that constitute the *control system*. These mediate between the sensors and actuators in order to transform the inputs from the sensors into actions applied to the environment. The control system's purpose is to provide an operation ability to the robot such that it can provide services to the user. A useful working distinction is between *physical* and *informational* services. Physical services are those that involve the physical manipulation of objects or involve physical contact with the user. Informational services are those that are related to communication. For instance the robot can serve as an easy-to-use Internet access based on speech dialog.

Vision systems are used in modern robotics in order to provide data about partial information of the current state of the environment. The primary attraction of the use of vision in service robot control is the large amount of data available at fast update rates. Based on the image data and subsequent recognition processes, decisions can be made. Instant or future actions may be triggered or planned. Some possible

applications for vision systems in robots are given in the following list:

- mobility: goal following, localization (metric, symbolic), obstacle avoidance
- manipulation: object detection (position, pose, state, category), collision monitoring, object tracking
- human-robot interaction: human detection, human identification, tracking of body-parts (e.g. to support gesture recognition or safe interaction), face analysis (emotional state recognition, viewing direction), speech recognition support

This set of applications is meant to be illustrative, not exhaustive. The vision system is an important component of the control system. Other components within the control system interact with the vision system. Typically, receiving components require some high-level information from the vision system in form of e.g. symbolic (predicate logic statements) or geometric information. Then the robot's world model can be updated and suitable actions can be selected. Direct coupling between vision and action selection is also possible, e.g. for reactive obstacle avoidance or person following.

Vision systems are not only important components of robots. A variety of applications in the modern technical world already incorporate vision systems and there will certainly be more use cases in the future. Some examples are listed below:

- automation: quality control, object detection, counting, safety systems, goal following, localization, obstacle avoidance
- security systems: entry control (person detection and identification), warning/alert systems
- medical applications: assistance in diagnostics, operation guidance/assistance
- video games/virtual reality: gesture recognition/tracking, action capture
- digitalization: scanning of large areas/buildings and generation of virtual models

1.2 Object Recognition

What is object recognition? In this section two quotations are discussed which try to answer this question. The first quotation (see [57]) stems from the field of artificial intelligence while the second quotation (see [19]) is an objective of the more specialized field of computer vision. The first quotation is:

> [...] the system must, based on the camera image, identify the corresponding

> object. The problem is that objects can change their appearance because of viewing positions, photometric effects (e.g., light conditions), object setting (e.g., different backgrounds), and changes in shape (e.g., animals). The core problem in object recognition is to somehow relate the many views that one and the same object can generate. For example, a car can look very different depending on the viewing position, but it is always the same car.

It is stressed that the object itself and the general scene conditions can generate many different views. An extension to this might include the variations within general object categories. In the second quotation the problem is described in terms of requirements for an ideal (fictive) object recognition system:

> The ideal object recognition system would
>
> - recognize many different objects:
>   This is much more difficult than it sounds: To recognize large numbers of objects, we need to know how to organize them into a data structure that is easily searched given image data. In particular, we need to know what measurements can be used to distinguish between objects as opposed to distinguishing between instance (one cat may be tabby, the other gray: they are both cats).
> - recognize objects seen against many different backgrounds;
>   Again, this appears to be difficult. Ideally, an appropriate object representation would help by organizing the image into segments that might have come from an object category (without reference to a particular instance) and those that could not.
> - recognize objects at an appropriate level of abstraction.
>   Humans do not need to have seen a particular chair before they know it is a chair. Ideally, our programs would be able to recognize both leopards and cheetahs as spotted cats before drawing a distinction. Just precisely what is an appropriate level of abstraction is mysterious; at least part of the issue is tied up in the question of recognizing many different objects.

In contrast to the previous definition of the object recognition problem, here the problem of category detection is included as well. Further extensions of these capabilities are the recognition of functions of objects or the detection of the states of parameterized (articulated) objects [46].

1.2.1 Localization of Objects

In the quotations given in the previous section, the terms “recognition” and “identification” were not defined. The scenarios can be used for clarification:

- An object is presented in a segmented way or with some spatial reference (e.g.

the user points to the object) and the robot must tell "what" object it is.

- The user tells the robot which object he wants and the robot finds out "where" the object is.
- Complete scene analysis: an estimate about which can be seen and where they are.

In the "what" and "where" cases one can distinguish between different levels of object abstraction, e.g. specific objects ("my red cup"), object categories ("a cup") and more abstract categories ("dishes").

In this thesis the "where" part is the subject of interest related to specific objects, not categories. The "where" is given some geometric meaning. It relates to a coordinate system that is attached to a rigid object that has to be found in the scene. A robot that has to manipulate the object has to "measure" the relative position and orientation of this system in order to derive further information such as grasping point positions. In figure 1.2 this setting, including important coordinate frames, is shown. Note that for description purposes, in this document a general frame A is notated with a large letter associated with a four-tuple of origin and axes vectors, i.e. $A = (\boldsymbol{o}_A, \boldsymbol{i}_A, \boldsymbol{j}_A, \boldsymbol{k}_A) \in \mathbb{F} \subset \mathbb{R}^{3\times4}$ where the real numbers $\mathbb{R}^3$ represent the Euclidean space in millimeter units. Furthermore, the notation $C = {}^AB$ is used to describe the frame C that expresses how B is located and oriented relative to A. The computations necessary to operate on coordinate frames are given in [14]. Here a simplified notation is used. The transformation matrix is not written explicitly. In the appendix of this document the notation conventions are given.

The geometric setting of object and robot includes a world frame W that is the neutral frame in which all other frames are expressed. Furthermore, a robot frame R, the end effectors frame E, and the sensor frame S belong to the robot and can be moved independently according to the kinematic equations involved. The object frame O is assumed to be attached to the object. An assumption in this work is that the object is rigid, i.e. it does not possess any internal *degrees-of-freedom* (DOF) nor is it plastic. To estimate the conversion frame from O to S is the basic problem of object localization. This is notated with D, i.e. the frame $D = {}^SO$ describes how O is located and oriented in S. Therefore, if D is known then also O is known since R is known by the localization system of the robot and ${}^RS$ is based on the known kinematic relationships inside the robot. Finally, the object position and orientation can be expressed in robot or world coordinates as a basis for further actions.

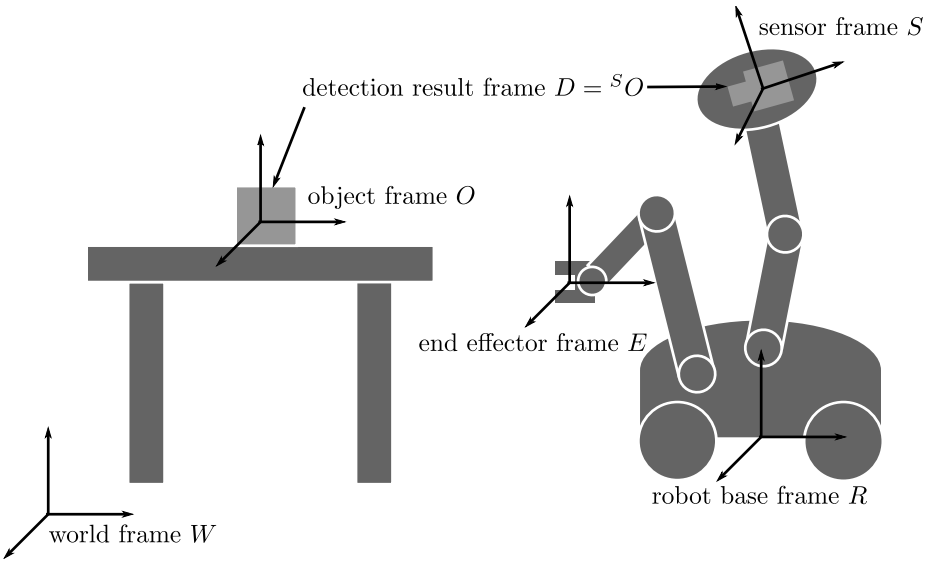

Figure 1.2: The geometric setting of object placed on a table in front of a robot. It includes a world frame W, a robot frame R, an end effector frame E, an object frame O and the sensor frame S. The conversion from O to S is the basic problem of object localization. This is notated with the frame $D = {}^{S}O$ that describes how O lies in S.

1.2.2 Learning Object Properties

A second criterion not treated so far, is whether recognition can improve over time based on experiences made with the object. The learning ability of humans is enormous; we can learn object appearances, behavior and functions through interaction with the object. The more learning capabilities are provided by a robot object recognition system, the more open-ended is the system. If an object cannot be recognized the first time, it can be learned and recognized subsequently. Ideally, this would be possible for single object instances of a certain category and also for category properties (e.g. when the robot knows what many "cups" look like, then it does not have to be taught each new instance of the category "cup"). Only a system capable of learning is useful in real-world application scenarios. Hence, in addition to the localization problem, a second interest in this thesis is the learning ability of the object detector.

1.3 Outline of the Approach

1.3.1 Basic Design Decisions

Some basic design decisions that are seen as working assumptions or hypotheses in this work have been made. Therefore, not all of them are exhaustively defended against alternatives in this document.

- The first assumption is that a visual approach is chosen. There can be various technologies used to detect objects in the robot's working environment (e.g. the use of RFID transponders attached to the object or intelligent environments). Typically these methods require the preparation of objects or environment, resulting in a need to organize different technology contributing parties. Therefore the visual approach seems reasonable. Another issue is that other, e.g. tactile, sensors can be used for grasping objects. However, these systems cannot estimate the object position in a larger region of space.
- The second working assumption and approach is the use of both RGB and 3D camera as separate devices. The need for 3D information in the context of object recognition is obvious since the size/distance of unknown objects or precise object shapes cannot be sufficiently detected with 2D color cameras alone. 3D data can be reconstructed from multiple 2D images but only reliably at matching locations. The use of two separate cameras for 3D and color in this work is because only very recently combined sensors have been developed. However, the use of two different imaging devices is still interesting in practical applications, as the two sources can be chosen depending on the application needs. Advances in resolution and speed related to both technologies can be combined easily.
- The third working hypothesis is that learning with on-board sensors is a good way to approach the extensibility of the robot's perceptual capacities. Object models could also be downloaded from the Internet by the robot. This requires manufacturers to publish models or for models to be scanned with some standard sensor devices. The main objection here is the organizational burden of the standardization, collection and distribution of this data as well as its adaption to specific robot hardware. An additional advantage using the on-board sensors is that error effects specific to a particular robot hardware operating in a specific environment occur at learning as well as during perceptional operation. Hence, some error effects can be nicely by passed.

1.3.2 Overview of Prior Approaches

As noted in the previous section, the focus in this work is on the localization of specific objects and how this can be done efficiently with models that can be learned on-line, i.e. during the robot's operation. In the following, some aspects of earlier works that gave rise to the approach introduced in this document are described. Here only a brief overview is given. More details and references to important publications are discussed in chapter 2.

Trainable classifiers such as artificial neural networks or *support vector machines* (SVMs) can be used for tasks like handwritten character recognition as described in [73]. The basic idea has been applied successfully to images that contain objects seen from different view angles e.g. in [58]. There are improvements to achieve robustness against partial occlusions and other disturbances in [64]. The problem with these "pure" learning approaches is that many effects that typically make object recognition in arbitrary scenes difficult must be communicated through the training set. This leads to a combinatorial explosion. Furthermore, in some works complete image patches are subject to classification. This means that the image position of the patch of interest must be pre-selected somehow. It follows that complete object detection as would be desired is not to be solved based only on these methods. However, the advantage of learning approaches is that object properties do not have to be modeled explicitly, only the example views must be given. In real-world scenarios it is easier to provide sample views than correct and complete geometric models.

Modern feature points such as SIFT or SURF ([47] and [4]) are made invariant against typical effects in vision, e.g. changes in position, size, brightness or even affine changes. This means that these invariant attributes need not be communicated with the training setm, as would be the case for a pure classifier-based approach. Generally, the system components that follow the feature point extractor (e.g. matching or classification components) may be less complex due to the properties of powerful feature points. Also partial object coverage can be tolerated if smaller subsets of feature points are sufficient for detection purposes. Therefore, approaches to (single) object detection are quite successful using feature points with matching descriptor level together with some global voting on pose consistency method. Using 2D feature points without any depth estimation leads to the problem that 3D models cannot be constructed. The feature point's size in pixels cannot be related to the real size in meters.

A third category of approaches works with range data directly. Geometric shape descriptions can be used (see [69] or [1] combined with some iterative model fitting strategy. Also so-called spin-images [35] and local shape context [6] can be used for modeling and recognition based in range data. These approaches do not use the texture information obtainable with color cameras. However, intensity, color and

contrast information can provide very distinctive features. Objects may have the same 3D shape, but different surface texture.

More recent and current work focuses on the use of color and depth information for recognition. The presented work can be seen as part of this new direction. On the one hand there are algorithms e.g. as part of the *point cloud library* (PCL) (see [44]) that work on point cloud data that is further augmented with attributes such as color information. The problem with approaches working on dense point clouds is the high data volume that has to be processed. Therefore, current work concentrates on how such algorithms can be implemented in parallel, to be executed on GPUs or on multiple processor cores. A second direction that uses 3D and color data are approaches that work on multiple 2D images (e.g. stereo-vision) to recover the depth information of feature points. The advantage is that there are typically fever features with augmented 3D positions than 3D points in the dense point clouds. The problem is the missing shape information other than the 3D point locations of the features. Furthermore, the existence of the features used is a necessity and the depth estimation step requires an extra feature matching process. Examples for these approaches are [22], [43], and [26]. With the use of both 3D and color information it becomes possible to use distinct features. Both modalities can be integrated in one approach either feature type can be enriched. These opportunities lead to interesting new research directions.

1.3.3 Basic Ideas of this Approach

The idea behind the approach presented here is to use powerful feature points augmented with 3D information. Here the 3D information stems from a 3D range image that contains a depth value for each image location. Therefore the normal orientation can be included as a further attribute in addition to the 3D position. Together with the 2D feature point gradient, full (6D) poses can be defined for each feature point [38]. Also, the general availability of range and color information is explored in the context of robot object learning. Construction of a range/RGB sensor can be done by using two separate cameras, one for range and one for color images with software components that provide the necessary transformations. Using this information some simplifications are possible:

- The first possible simplification enables sample data acquisition. Most of the approaches that are reported in the overview of the previous section do not incorporate any means of automated sample data production. Usually they rely on a "clean" object training set given a priori. In the work here, the combined 3D and color data provide means to extract such training sets during the operation of the robot. The segmentation method effectively "cuts" out the

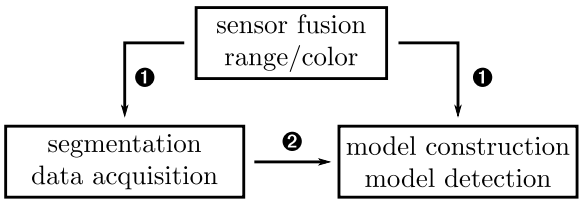

Figure 1.3: Illustration of the main areas of work in this thesis. The fusion of range and color information feeds to the segmentation and sample acquisition part ❶. The segmentation and sample acquisition produces training sets for the model construction ❷. Model detection reads in the sensor data ❶ and the constructed model from the training sets in order to detect it.

object appearance in the image using 3D context information. In this thesis solutions are provided on a conceptual level of how to retrieve this context knowledge in real-world learning scenarios.

- A second opportunity that comes with the combined data is a simplification and improvement of object model construction and detection. Complete coordinate systems are built on object surfaces at feature point locations. If these systems are stable, then a one-one correspondence pair consisting of a object model feature point and a scene feature point can reveal an approximate pose already. Many such hypotheses can be generated very quickly and stored in an accumulator data structure. Dense regions in the accumulator lead to the final object pose. Based on some experiments, it is shown that detection is possible even if large regions of the object are covered and even if the fraction of pixels that belong to the object is very small compared to the full scene.

The whole approach of this thesis is split into three main work areas as shown in figure 1.3. An important point that separates this work from others is the integral treatment of these areas. Furthermore, novel ideas within these topics are presented.

1.4 Organization of this Document

The thesis is structured as follows. First, chapter 2 introduces related work that is seen as background for this thesis displaying the state of the art. Then, chapter 3 gives a more detailed problem formulation of what kind of questions are tackled in this work, followed by an overview of the components that are selected, integrated, and further developed. There are three individual chapters devoted to the technical areas shown in figure 1.3. Chapter 4 treats the transformations and calibrations needed for sensor fusion. How the combined information can be used to acquire training data sets in real-world learning scenarios is the subject of chapter 5. Then in chapter 6 the extraction of feature points and the construction of local coordinate systems at feature point locations is described. Model construction and detection are also treated in chapter 6. Chapter 7 contains results including outcomes of experiments and tests. Finally, chapter 8 summarizes the work presented, draws some conclusions and discusses improvements and future work.

2 Related Work

In this chapter a selection of some foundations, methods and algorithms that are seen as background for this work are presented.

2.1 Classical Vision and Recognition Paradigms

In [28] general *vision stages* of typical object recognition systems are described. These are shown in figure 2.1. They comprise a generic framework for object recognition. *Conditioning* is the process that prepares the image e.g. by noise removal. *Labeling* assigns a discrete label to each pixel in the image. If there are only two possible labels this is called *binarization*. The goal of *grouping* is to find connected regions of pixels that belong to the same label. *Extracting* is the process of computing some geometrical description for each region. *Matching* is the final process of identification. It is verified whether or not the extracted shape matches the object being searched. This scheme structures typical robot vision systems. It stresses the top-down processing of data during which the important information is extracted from the images while the volume of data is reduced. These stages are very basic and can still be found in modern vision systems.

Another view of vision systems is that of a general pattern classification system that has a sensor input and outputs a discrete label. The input could be an image and the output a decision on the result, e.g. "this is object X." Generic stages of a pattern classification system are listed in [16] and shown in figure 2.2. Here, the data acquisition process is termed *sensing*. Similar to the labeling and grouping operations, introduced above, the next stage is *segmentation*. However, segmentation is more general. It means that pixels can be grouped according to whether or not they belong to an object. Segmentation is followed by *feature extraction*. A segmented region is processed with a feature extractor that converts the region into feature vector. This is a vector that consists of a selected set of task dependent attributes that were chosen during the system design. During the *classification* step the classifier processes the feature vector and outputs a class label. The *post-processing* stage can be used to include additional information such as context knowledge to improve the confidence of the result. The difference between the classifier approach and the

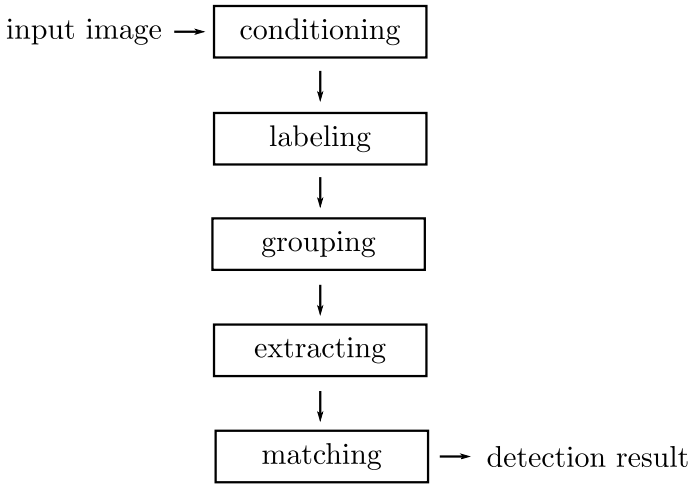

Figure 2.1: Classical stage model of vision systems.

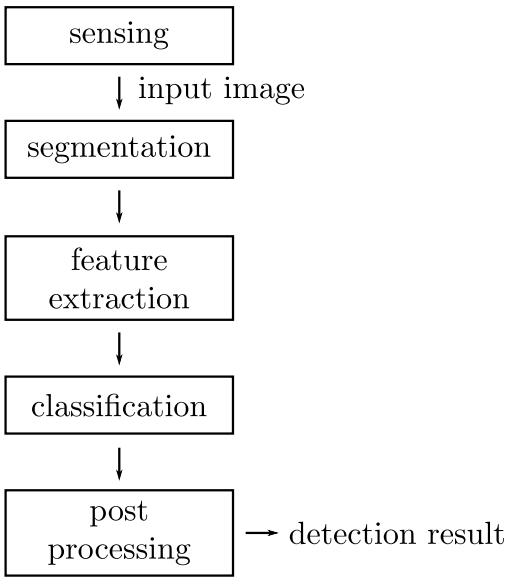

Figure 2.2: Generic stage model of pattern classifier-based systems.

stage model shown in figure 2.1 is that (typical) classifier systems are not tuned to geometrical perception tasks such as object localization or measuring orientation, but to rather qualitative outputs such as object categorization or detection of "faulty" cases in a quality control system.

2.2 Newer Approaches to Robot Vision

Top-down vision systems (see figure 2.1) fit to the classical *sense-plan-act* control scheme. New considerations on robot control architecture such as goal-directed *behavior-based control* have come up with new views related to robot control. Here, parallel behavior modules run simultaneously, each representing a certain goal of the system. The new approaches to intelligent robot control also have an effect on how vision can work in such a system. Instead of processing a complex image analysis sequence, special information is passed directly to the control modules (see [2]). The close coupling of sensors and actuators allows for behavior learning. Experiments reported e.g. in [52] and [76] are early examples of visual-motor learning.

There is a direct closed-loop approach that includes both vision and action in a goal-directed scheme. In the context of *visual servoing* [37] a closed-loop motor controller measures the feedback with a visual system. For instance, this allows the robot to grasp moving objects or compensates for errors of the initial pose estimation. Visual servoing can be applied to grasping tasks by a so-called in-hand mount of the vision sensor where the camera sits on the gripper. The discipline concentrates on the question of what vision pre-processing strategies are suitable and which control systems have to be chosen. Simplifications of both sides (control and vision) are also possible due to this integrative view.

The link between vision and action has also been reconsidered on the vision side. In the field of *active vision* [8] the assumption is that the camera can be moved by actuators which are controlled *by* the active vision system. Active vision addresses questions like how to acquire structure from controlled motion, general tracking, active focusing of attention, and sensing strategies to acquire data efficiently.

In *manipulation-driven vision* ([18]) the objective is to use manipulation to segment objects if a visual approach fails (e.g. object and background colors match). Closely related to this field is another topic that emphasizes the close interaction between vision (perception) and action, called *embodied vision* [3]. Here, the need for human-robot interaction is addressed in addition to the use of actuators. Further works focus on how vision can be combined with additional perceptional sources. For instance, force measurement sensors (see e.g. [75]) can be used.

2.3 Algorithms for Object Recognition

2.3.1 Pure Learning Approaches

As introduced in the previous section, object recognition can also be seen as a general pattern recognition problem. The typical assumption made is that objects are shown in a database at similar scales, cut out from the background at various rotational positions (e.g. generated with a turn table). Examples of reference databases of this sort are the COIL database [53] containing 100 objects each seen from 72 views (i.e. $+5^\circ$ rotation steps) or the ALOI database [21] that contains 1000 objects (also 72 views per object). In this setting the actual problem to be solved is to label the image with a distinct label associated with the object. The fields of *computational learning theory* and *statistical learning theory*) focus on algorithms that learn from samples given. There are three general problems investigated that can all be related to object recognition:

- Pattern recognition: the sample set is given together with output labels that have to be estimated after the algorithm is trained. In the simplest case, binary classification, there are only two values possible. Multiple class problems can either be solved by a network of binary classifiers or in some cases the algorithm itself can handle multiple labels. The relation to object recognition is that the input vectors can be directly sub-sampled from image data or that there is some preprocessing that provides for feature vectors of fixed dimensionality. An output label might denote a specific object or some general category. One early example of object recognition using classifiers on raw images is [58].
- Regression estimation: the goal is to derive numerical functions from high dimensional input data. The output space is typically a value in $\mathbb{R}$ but can be increased in the number of dimensions by using a collection of regression estimators. This could be related to compute orientations of an object based on the pure appearance.
- One-class learning: only the input values are given. This is also called *unsupervised learning*. The advantages of one-class machines are their suitability if the number of classes is unknown initially and the possibility of novelty detection [67].

There is a large set of algorithms available today. The SVM is a modern type of learning machine that has outperformed most other classical systems in a variety of applications since it was introduced [12]. A network of binary (two-class) SVMs was applied successfully to object recognition by classifying complete object images [58] of the COIL database (see [53]). Further experiments are reported with good results even if noise is added, the object is shifted in 2D, or the object is partially covered.

A continuation of this work with further improvements is presented in [64]. This system improves translation invariance and adds scale and background invariance. The latter is realized through adding artificial object views with white backgrounds to the original samples of the data set that have black backgrounds. Also, an incremental *hierarchical discriminant regression* tree can be used as an on-line- earning machine to classify complete images [77]. In the paper successful experiments to achieve invariance against 2D orientation of images are reported.

2.3.2 Modern Invariant Feature Point Detectors

Complete images can be indexed with an object label by using powerful learning algorithms as described in the previous section. There are, however, problems if these methods are used alone to build an object detector that has to detect objects under a large range of transformations (3D position, 3D orientation, camera parameters, etc) and other disturbances (e.g. shape deformations, occlusions, coverage, shadowing, etc). In detail the following problems occur with systems that are purely based on learning direct object views:

- Many object views must be presented in order to capture the full object information.
- There is a need to produce "artificial" training samples as e.g. done in [15] to account for many possible transformations.
- If the object has internal degrees of freedom, then also many state settings must be produced.
- In detection mode it is difficult to determine which region to select out of the scene data.
- Object-background separation is difficult if there is no knowledge about the object and background available.

So-called *feature points* help to overcome some of the difficulties mentioned above. They can establish a connection between the visual appearance of an object, its parts and the real underlying geometric shape. The idea is based on the fact that the image contains some salient regions or points that are detectable under many variations like 2D translation, 2D scaling, 2D rotation, affine changes, intensity variations, or noise. The second idea of feature points is that they also contain regional information collected in a high dimensional vector, called *descriptor*. The descriptors can be used to discriminate between feature points. Feature point detection and descriptor computation need to be stable against the same image transformations. There are different research groups actively investigating alternatives methods related to both

the detection of stable points and descriptor computation. In [66] a general concept for feature point comparison is introduced based on the concepts of *repeatability* of a feature point and *information content* of the descriptor. A first such feature point was introduced in [50]. These are based on a local discrete window that is shifted in several image directions. The differences to the origin (central) window are summed up. Therefore, features are selected that are defined stably in their 2D position. The features were improved in [29] using the autocorrelation matrix which is related to the Hessian matrix. These so-called *Harris features* were further improved in [66] by a more precise and faster localization method.

A fundamental extension to feature point detection is to use a pyramid representation of an image called *scale space*. A multi-scale version of the Harris-features was introduced in [49]. *Scale invariant feature transform* (SIFT) feature key-points ([47]) perform well in database tests and there are several applications using these features. The (binary) software can be downloaded at [45]. There are further implementations e.g. at [30] and others. There are also activities that port SIFT either to specialized hardware [10] or to compute feature points based on the GPU [79]. A common interest is to improve feature detection by:

- Reduction of the descriptor dimensionality
- Speed up of the computation time

A significant speed up in computing the scale space is the use of an *integral image*. A well-known method for fast object detection using features based on integral images is [74]. An integral image-based variant of the SIFT features is published in [48], the *FastSIFT* detector. It is a fast algorithm that is based on box filters realized with integral images and on some simplifications compared to the original SIFT approach. The improvement in computation time in comparison to the original SIFT is about a factor of eight. The feature point quality is comparable. Another new type of feature points is the so-called *speeded-up robust features* (SURF) detector [4]. This method is based on a fast Hessian computation that is performed using the scale space. The feature point descriptors are histograms of Haar-like wavelet responses that are summed up to four different measures: sum of dx, sum of $|dx|$, sum of dy, and sum of $|dy|$ over different frequencies. These values were selected based on experiments including different possible regional attributes. The authors show that these features are faster and more accurate than other state-of-the-art detectors. Other promising attempts related to feature points are the *star features* that can be used together with the *fern* method for matching ([56] and [55]) and the more recently appeared *BRIEF* features [9].

2.3.3 Feature Points and Classifiers

So far, the classifier methods and approaches using feature points were introduced separately. However, they can also be combined in order to recognize a single object or to detect general object classes. In the following, some examples of such approaches are briefly introduced.

In [40] SIFT feature point descriptors are classified based on one-class SVMs. A popular approach uses a cascade of weak threshold-based feature classifiers to detect objects [74]. An AdaBoost [20] learning strategy is used to select for those features that minimize the overall error. The result is a trainable single-object detector that works on gray-level images. However, this method takes a long time for training since all features of all sizes and width/height ratios need to be considered. However, detection is very fast since only a subset of features is used for matching. In [80] local (PCA-SIFT) and global contour features are fed into an AdaBoost classifier network. The novelty is that also the *types* of important features are selected using the AdaBoost procedure. Experimental results demonstrate a high performance of this method in detecting unknown objects of a trained category.

2.3.4 Geometric Models and Matching

The detection of the object's presence in the scene can be carried out virtually moving the object to some part of the scene where it fits. In this case detection is tightly interlinked with pose estimation. The data representations and matching algorithms that allow for robust and fast geometric matching were and are subject to research. In industrial applications, the *computer aided design* (CAD) models of objects are sometimes available. They can be used for detection and pose estimation in gray-level images or based on depth data acquired by some range sensor. Mathematical functions can be used to model and detect objects as single or compound basic geometrical bodies [1]. Also deformable objects such as Superquadrics can be used for object modeling. In [34] several types of Superquadrics are introduced together with additional operations for global bending and tapering. In [40] multiple Superquadrics are combined into an object model. Another popular shape representation is the triangular mesh-grid. Mesh-grids model the surface shape as neighboring triangles. These models have also been used for detection and pose estimation in robotics (see e.g. [70]). Furthermore, *oct-trees* can be used to represent 3D shape. This is especially interesting due to the possibility to inspect coarse structures fast (e.g. real-time obstacle avoidance).

A classical and very rudimentary method is to represent shape on the basis of so-called *point clouds* (sets of 3D points). The "correct" shape could only be modeled

with point clouds of infinitely many points. Therefore, point clouds used in practice are sparsely quantized approximations of the underlying surface structure. Two types of point clouds can be distinguished:

- *Raw* point clouds are directly derived from some range image sensor.
- *Pre-processed* (sparse) point clouds are reduced point clouds containing stable 3D feature locations.

There have been proposed different approaches in order to match point clouds, among them:

- For some distributions (e.g. L-shaped objects) the PCA method can be used to align both clouds.
- The transformation between two point clouds can be estimated on the basis of a closed-form solution based on *quaternions* if the correspondence point pairs are known [32].
- The *iterative closest points algorithm* (ICP) ([7] and [65]) iteratively fits two point clouds by estimating a transformation in each step and applying it to the argument point cloud that finally matches the source point cloud. Each iteration only uses a number of closest points. These are passed on to the frame estimation algorithm. The process is also called *registration*. The method is robust to some degree of noise. However, a pre-estimate of the transformation is required.

Some methods can be derived from 2D matching strategies. In the following list some examples are introduced:

- The *random sample consensus algorithm* (RANSAC) algorithm ([17]) can be used to estimate the transformation between two point clouds based on point triples. These can e.g. be defined as a point together with its first and second nearest neighbors. The algorithm would have to guess correctly two corresponding point triples. There is only a limited chance for success if the number of points is large and if the points are indistinguishable. Improvements of the RANSAC such as PRONSAC [13] try to speed-up the matching process by keeping track of good matching candidates.
- The *geometric hashing* method can also be used to match 3D point clouds (an overview is given in [78]). The algorithm builds a map with discrete buckets in which possible poses are entered. Each point triple defines a local coordinate system in which neighbors are expressed by discrete coordinate keys. Thus, the point triple becomes distinctive based on the point neighborhood configuration. During detection all possible matches based on the key entries are checked. Each match votes for a transformation between the two point sets. All votes

are stored in an accumulator. The result is based on the accumulator cell with highest count. One problem is the discrete nature on the voting space and another is the search complexity related to the point triples.

- Finally, the *Hough transformation* [33] could also be used to match two point clouds based on two matching 3D point triples. The triples are used to generate votes on transformations. Finally, the generated votes have to be processed in order to find a maximum density in the voting space. The problem here is the search complexity related to finding the correct point triples. However, there is no loss of information as the correct votes are conserved in the voting space for further analysis.

Further approaches that work on range data are based on local geometric shape descriptions. Spin-images [35] and local shape context [6] can be used for object modeling based on range data.

2.3.5 Combined 3D and 2D-based Approaches

Recent and current research work focuses on approaches that use 3D data combined with gray-level or color information. One motivation for this is that the gain in information that allows to cover a larger object class. The shape of two different objects can be equivalent while the surface texture differs. Vice versa, the color appearance can be equivalent but the shape differs. Two research areas can be identified that address the use of 3D information:

- 3D features based on multiple (e.g. stereo-vision) images
- Point cloud approaches with augmented color information

In [22] a first method is presented that is based on gray-level images. The following is a quotation from the abstract of the paper [22]:

> [...] we describe a system for constructing 3D metric models from multiple images taken with an uncalibrated handheld camera, recognizing these models in new images, and precisely solving for object pose. This approach not only provides for accurate pose, but also allows for integration of features from multiple training images into a single model that provides for more reliable recognition.

The approach is based on SIFT features that are computed from 3D image pairs. Learning and model detection stages are included. The poses are recovered quickly and accurately.

In [42] *random trees*[1] are used to classify feature points in stereo-vision images. A RANSAC algorithm is used to estimate a possible transformation between the model and scene feature sets. As a result the full object pose is obtained very quickly. The method can cope with limited background structure.

Though these approaches are promising, they still model the objects on the basis of 3D points that are only determined locally. Further shape information such as local normals are not used. However, using such information could lead to a reduction in matching complexity and to object models containing more information. Another problem is that typically such systems have to compare descriptor distances between the scene and several object models. The labeling is done based on relative distance measures. It would be an advantage if descriptor models were only depending on the single objects and not on objects sets.

Point cloud approaches such as e.g. ICP can be enriched with color information (see e.g. the web page of the Point Cloud library [44]). Point clouds are usually very data intense. Finding a relatively small object part in a large scene is seen to be difficult given the large number of possible correspondences.

2.4 Integration 3D Imaging Sensors

Robotic vision systems are sometimes augmented with panned or rotated laser range finders (see e.g. [23]) in order to obtain color and range information about a region in the environment. The disadvantage is the large amount of time needed for data retrieval due to the necessary motions. A way out of this is the use of 3D *time-of-flight* (TOF) cameras which are based on specialized camera ICs which measure the time of flight of an artificially produced light signal to an object and back. The measurement is performed for each pixel separately, yielding a complete 2D range image. There are also attempts to combine range and color information about the corresponding locations in the environment into one coherent data representation. In order to combine both range and color information there are several methods possible, among them:

1. Stereo-vision or multiple-vision systems realized with color imaging sensors (see e.g. [62])
2. Panned or rotated laser range finder integrated with a color imaging sensor (see e.g. [23])
3. Color imaging sensor and a time-of-flight range imaging sensor (see e.g. [41])

[1] This term is related to the random trees learning algorithm and not to be confused with the random trees method for path planning.

4. Stereo-vision and time-of-flight sensors (see e.g. [27] and [82])
5. Other depth imaging techniques (e.g. the "light coding" approach [60]) combined with color sensors.

   A comparison of general 3D technologies is given in [71] and [11]. Different time-of-flight cameras are discussed in [36].

2.5 Summary

The following itemization lists a short summary of important points:

- There are generic vision and recognition paradigms that describe the derivation of information from images to the desired result in a step-wise succession of abstract operations.
- Reconsidering the interrelation between vision and action can considerably simplify vision problems. For instance, the presence of a body that surrounds the vision system can simplify and help to solve vision problems.
- Learning algorithms can be used to work directly on image data even without intermediate (feature) representations.
- Robust and fast computable feature points can be used to detect objects even in the case of occlusions and other disturbing effects.
- Geometric models are a classical way to match objects in the 3D scene.
- Object shapes can also be naturally represented as, raw or pre-processed, 3D point clouds. There are a number of algorithms that can be used to match two 3D point clouds.
- A difficulty of point cloud matching is the large search complexity. At least two matching point triples are needed to estimate a single pose hypothesis.
- New imaging sensors, especially range imaging sensors can significantly simplify general vision problems since the 3D distance is given directly for each pixel.
- Recent and current work focuses on combining 3D and color information in order to detect object poses fast, precisely, and in complex scenes.

  The shortcomings of the works presented in this chapter and the rationale and motivation for the approach in the presented work are discussed in the next chapter (3).

3 Problem and Approach

The approach presented in this document has been developed in a number of research projects related to vision for mobile robots. All considerations that influenced the final approach were guided by some requirements one faces in real-world scenarios containing real objects and real robots. In this chapter the aim is to give a structured overview about the motivation and the realization of the integrated new approach.

3.1 General Motivation

In the previous chapter a collection of state-of-the-art methods is given. In this chapter the aim is to derive the approach presented here. Classes of object recognition systems are presented in section 3.2. Then in section 3.3 observations and inspirations that gave rise to the new approach are listed. A system overview follows in section 3.5 and section 3.6 summarizes this chapter.

3.2 Classes of Object Recognition

The following aspects help to distinguish between different kinds of object recognition systems:

- Object properties
- Scene conditions
- Input/Output types
- Learning ability

These aspects are treated in the following subsections.

3.2.1 Object Properties

A service robot may face a nearly unlimited set of objects during long term operation. It is not easily possible to give an exact definition in a rigorous way of what an object is. For instance, a typical set of objects could be those things we have in our household. It is clear that such a large set contains objects with very different attributes. Some of these properties are given in the following itemization:

- Shape attributes: concavity/convexity, compactness, topological degree, whole/part relationships, entropy measures, etc.
- Surface and material properties: arbitrary texture, glittering and metallic effects, transparency (e.g. glass), mirroring surfaces, light absorption (holes), etc.
- Changes of state: mechanical state (internal DOFs or plasticity), biological/chemical state, electrical/electronic state (e.g. displays change their appearance), etc.
- Other effects: e.g. long term effects (a candle shrinks over time, plants grow).

The requirement for a sophisticated robot would be to capture all of these effects to a degree that recognition and manipulation would be possible. Many of these object attributes still pose challenges. The problem is the large number of variations in appearance that can emerge. A term important here is *scalability*, meaning that the number and diversity of objects should not be limited by the type of representation selected. A further requirement is that the presence of many object models should not degrade recognition capacity nor aggravate the addition of new models.

3.2.2 Scene Conditions

There are cases in which a perceptual task is *easier* to complete with success than in other cases. In [64] different stages of object recognition systems are discussed by stepwise weakening assumptions about the image content. Inspired by this approach, in this thesis a number of *levels of complexity* of the scene setting are defined. These are shown in table 3.1 on page 27.

Each level introduces a new effect that permutes with all previous variations. In the first case there are no occlusions, uniform background and the fraction of pixels that belong to the objects is relatively high. Then in the next levels (levels two and three) background structure is added and the object is placed arbitrarily. At level three objects are placed at an arbitrary position and orientation and they occupy only a small fraction of the complete image. Thus also the background is arbitrary. If the

Level	Assumption
1	One of the (known) objects is shown always at approximately the same scale and 2D position but all three rotation angles my vary, the object occupies most of the image area and is presented against uniform known background.
2	The background can be arbitrary.
3	The object occupies only a small fraction of the image area and appears at different scales (z-coordinates) and alternative 3D positions (x- and y-coordinates).
4	The object view is further altered through scene setting related (global and local) factors.
5	The object's appearance changes due to factors related to object attributes.

Table 3.1: Complexity levels for object recognition systems.

object appears somewhere in the scene as in levels three and four, then the current view is changed due to *global* scene related factors such as the intensity of ambient lighting, or pollution. Also, *local* scene related factors can change the appearance of an object (level four). Neighborhood relationships of objects are a source for partial object coverage and additional lighting variations due to occurrences of shadows. Finally, level five introduces further changes in appearance due to object specific effects such as those described previously in subsection 3.2.1.

3.2.3 Input/Output Types

Many of the approaches that use classifiers are able to assign an object label to an image that contains an object. Approaches such as the SIFT detector find a 2D image position of an object. An important difference between the two is that they answer different questions:

- Classification: Given a number of trained object models and an input image derive the correct object label.
- Position estimation: Given the trained models, an image and an object label, estimate if the object is present in the image.

While the classification step is important, preference is given to the position estimation. The classification step can be derived from position estimation by evaluating a matching score. It follows a list of possible recognition system outputs:

- 3D position (translation)

- 3D orientation (rotation)
- State information (e.g. mechanical state)
- Object instance (individual object, e.g. *this* cup)
- Object identity (class e.g. cup)
- Object category (e.g. dishes, tools, function, etc)
- Multiple objects
- Others: ownership, object sub-components (part-of relationships), grasping points, etc

A difference between methods related to object recognition is the input other than the image itself. For instance, a label can be used as a selection of which object has to be found. Or, as noted before, a position can be given as initial guess and the question is *what* object there is. A further difference is whether or not the method can judge the presence of the object. As described in the introduction (see chapter 1) the goal of this work is an object pose output assuming the object label s given.

Another main distinction between types of recognition is passive "one-shot" recognition or permanent (active/attentive) recognition. In this work the "one-shot" case is followed since other schemes can be built upon it. Here, the processing type is blocking, i.e. once called, the recipient has to wait until termination. The goal is to make the single-object detection efficient such that it could be used as a component of a more sophisticated attentional vision system. The goal is to provide a method that can augment/enhance active and embodied vision systems.

3.2.4 Learning Ability

The recognition system has to embed information about the recognition targets in some way. In the following this information is referred to as *object model*. There are different ways to add models. Table 3.2 gives an overview.

Level	Type	Description
0	Hard coded	Explicit models or related knowledge is included in the development process by the engineers.
1	Learning based (development)	Only implicit knowledge is given in the development phase as "samples" of object appearances.
2	Learning based (customer, batch/one-time)	The robot needs to be introduced to a new object at the user's place. It gathers all information about the object at once, based on interaction with the user.
3	Learning based (customer, continuous)	As the above but the robot constantly updates the model (e.g. in SLAM[1]).
4	Autonomous experimental learning	The robot can acquire relevant object knowledge autonomously through interaction or experimentation with the object.
6	Cognitive learning	The robot can acquire relevant object knowledge autonomously through "imagining" interactions with the object.

Table 3.2: Levels of model acquisition.

3.3 Observations and Inspirations

In the following some goals and guidelines are discussed that are to be seen as background for this work:

- The first goal is to cope with a number of object properties. While it would be desirable to include all object properties given in the itemization on page 26, the presented work focuses on coping with arbitrary shape and texture. Internal DOF, plasticity and surface properties such as glittering or transparency are excluded in this thesis.
- Emphasis is put on difficulties related to scene conditions. This work focuses on the fact that the object to be detected may be appearing at arbitrary 3D positions and orientations possibly with some parts covered. This includes cases where the overall observable are of the object is very small compared to the size of the field-of-view. This relates to the levels 3 to 4 of table 3.1.
- The output of interest is the full pose (or frame) of the object. The problem of detection is simplified in the sense that an input label tells the algorithm which object it has to search. The judgment of whether the object is present or not

is the primary interest.

- A learning ability is seen as a requirement in the work presented. Not only that the algorithms are able to construct models, but also how sample data can be acquired in the real world are basic interests. The aims towards level 2 of table 3.2.

The motivation of the approach introduced in this document is to make a step forward in these respects. The main relations to other existing approaches (compare with the itemization on page 23) are the following:

- The general vision stages in section 2.5 do not incorporate learning. Though the stage model of pattern classifiers includes learning, it does not provide any means of how the sample data can be obtained. Learning, including sample data acquisition, is explicitly addressed in this work.
- From the field of embodied vision one can learn that the presence of a body can simplify recognition. One approach to learning in this work focuses on how the body can be used for sample data acquisition, too.
- A hypothesis of this work is that sophisticated object detection cannot be solved with a clever learning algorithm alone. However, machine learning techniques do play a role here as a clustering method is used to index feature points.
- Feature points that have been shown to be stable against typical effects (shift, scaling, noise, etc) are incorporated in the approach presented.
- Geometric models in the form of functional shape descriptions are not of interest in this work.
- The concept of 3D point clouds is extended in this work by adding texture information and other attributes.
- With the extensions of point clouds the complexity of detection decreases significantly.
- The main ideas and methods worked out during the research related to this document are based on a combination of two camera devices, a time-of-flight range imaging sensor and a color imaging sensor.

3.4 Rationale on Algorithm Level

Apart from the items above, some considerations on algorithm level motivate the approach of this thesis. A 3D pose of an object can be estimated purely on 2D model-images correspondences. However, in order to do this, a 3D model of the feature points has to be available. This is difficult if not impossible to learn on the basis of 2D information only.

3D information can be used in the form of 2D feature points found in multiple images and reconstruction of the depth information based on triangulation (e.g. as done in stereo-vision). The 3D information is only available sparsely where features occur in the image. Since the 3D information is only determined at the feature's location it may be difficult to construct a surface normal or to retrieve other shape properties. However, the feature point-based approach in general is very promising as many solved applications, e.g. 2D object recognition and images stitching are based on feature points.

In 3D this means that when using feature points, less data has to be taken into account computationally than e.g. would be the case when working with full point clouds. Therefore, one rationale here is to show how the use of features points augmented with 3D information leads to a very sparse representation which is suitable for model construction and detection.

Another difference is that in this work an attempt is made to isolate descriptor models. The motivation is to come up with a true single-object detector. This means that all the parts of an object model should depend on the object alone and not on sets of objects.

3.5 System Overview

The complete approach developed can be divided into three parts:

- Sensor fusion and calibration
- Sample data acquisition
- Model construction and detection

A description of the approach is given in the following paragraphs. In figure 3.1 the sequences of computations for single-object learning and detection are shown that are presented in this thesis. There are three main areas. SENSOR FUSION provides combined images (here called *shared images*) that are fed to the DETECTION part and to the LEARNING part. The system has two modes. In the *model construction* mode

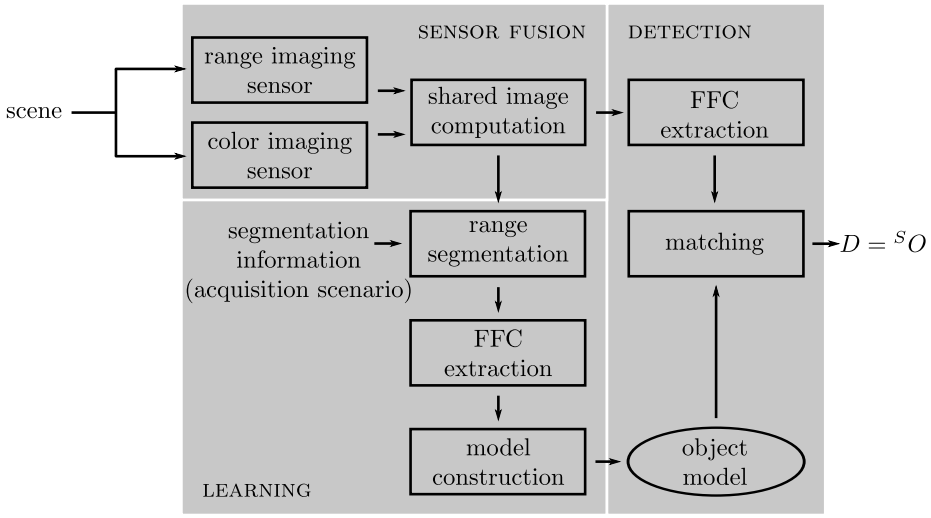

Figure 3.1: Proposed processing sequences for the model construction (learning) and model detection. The details of this figure are described in the text. Note that SENSOR FUSION parts provides single images (used in the DETECTION part) as well as image sequences (used in the LEARNING part).

the components depicted in the LEARNING area are active. During detection the components in the *detection* area are processed. Note that the two "FFC extraction" components in the DETECTION area and in the LEARNING area are equivalent.

At the heart of this approach there is the early fusion of both modalities at the very beginning of the recognition chain. An intermediate representation called *shared image* is an image with six overlaying channels for RGB color data and XYZ coordinate data. The discrete 2D pixel plane of these layers is congruent, implying that whenever some algorithm finds a result in 2D image coordinates either color information or coordinate information can be added.

With the shared image available, a second idea in this thesis is that *first-time segmentation* of learning views of an object becomes possible based on *range segmentation*. This means that *segmentation masks* defined in real 3D space can be used to cut out color regions that relate to the object's appearance. In a so-called *acquisition scenario*, components are used that are external to the vision system in order to get the information needed to position the segmentation masks.

The next idea of the approach suggested here is to use the shared image also for detection based on feature points. A new representation is the so-called *feature frame cloud* (FFC). Feature frame clouds incorporate feature points that can be accessed quickly and that are augmented with local coordinate systems. Thus, shape is also modeled, based on the parameters of the local coordinate frame. Detection and model

construction can be based on a single algorithm that estimates the relative frame between two FFCs. The suggested algorithm is based on voting on pose consistency and is similar to a Hough-transformation.

The complete approach realizes a single-object detector which constructs object models during learning and which outputs the detected frame $D = {}^{S}O$ (see figure 1.2 on page 7) during detection. The object frame O can be converted to to the robot's end effector frame to initiate a grasping action. A special attribute of this system is the explicit inclusion of a segmentation method that enables real-world object learning.

3.6 Summary

The main motivation that led to the work presented in this document is described in this section. The main goals are to detect the pose of an object in difficult scene conditions and with models that can be learned from sample data. This data can be acquired in the real world using a robot equipped with the selected sensors. Furthermore, an overview of the approach is given. Important terms are introduced and explained.

4 Sensor Fusion

One key idea in this thesis is the utilization of a so-called shared image that gives access to both 3D and color information about a region in the environment that is projected onto a virtual common image plane. This chapter introduces methods of how a shared image can be efficiently computed given the two selected sensory sources: a range imaging sensor and a color imaging sensor. Furthermore, the calibration method for the sensor set-up is described. The methods are derived from conventional camera transformations and calibration and extended by the specifics of the range imaging sensor. The methods introduced have been tested during the project work related to this thesis. There might be some overlapping of the methods described here with calibration schemes used elsewhere (e.g. published in the Internet). This topic has received much public interest recently due to the wide availability of low-cost range cameras used for video game control.

4.1 Motivation

The motivation is to obtain an image, called *shared image*, providing coordinate and color information at each pixel's location. The proposition is that such an image can significantly simplify existing vision algorithms or can lead to new, more efficient and powerful methods. The approach suggested here provides an approximation to an ideal shared image based on the following parts:

- a hardware configuration consisting of a conventional *charge-coupled device* (CCD) color imaging sensor and a TOF range imaging sensor,
- a method for reassembling a color image in the pixel coordinate system of the range imaging sensor by forward and back projection, and
- a self-contained calibration method for the selected hardware setting that provides for all parameters needed based on one calibration point list.

These methods are described from section 4.2 to section 4.4 in the same order as in the itemization given above. Section 4.5 gives a summary of the content of this chapter.

4.2 Hardware Setting

A combination of a color imaging sensors with a TOF range imaging sensor is selected for the computation of the shared image. Panned or rotated laser range finders are not chosen due to the fact that only very low frame rates are possible since extra movements are required to assemble a 2D image from single line scans. Furthermore, the main advantages of combining color imaging sensors with time-of-flight range imaging sensor over stereo-vision are:

- Uniform colored regions can still deliver 3D information if they are reflective in the infrared spectrum; this is difficult when using stereo-vision. In the optimal case, 3D and color information are defined for the complete image.
- There is no computational cost to pay for the correspondence search which is required for stereo-vision. For depth estimation based on stereo-vision, feature points or image patches have to be matched first in order to make depth estimation possible.

There are also disadvantages compared to stereo-vision systems:

- Current state-of-the-art time-of-flight sensors have rather low resolution, typically less than 200 pixels maximum in image width or height.
- The volume of the field-of-view is smaller since the emitted light energy rapidly decreases in intensity with the distance.
- The measured distances are not completely invariant against color variations in the reflected material.

In this thesis these disadvantages of using TOF range imaging sensors compared to stereo-vision are accepted in order to profit from the features of TOF imaging sensors. Usually, grasping tasks take place in the close vicinity of the robot. It is assumed to be more important that range values are defined for the complete image than that limitations in the field-of-view exist or that a lower range accuracy is obtained.

The purpose of the sensor set-up is to obtain range images and color images of the same area in the environment in a timely synchronized way at relatively high frame rates. Following this goal, a setting is selected of two sensors, a TOF range imaging sensor and a color imaging sensor, that are mounted next to each other and working in conjunction. During the work related to this document several such set-ups were used. These are shown in figure 4.2. A first set-up (see figure 4.1, ❶) was used early in this work for first studies. It included the Swiss Ranger 2 with 144x124 pixels in width and height and a color camera DFK42F02. The second set-up (see figure 4.1, ❷) was used later, including a range imaging sensor Swiss Ranger 3000 and the color camera DBK31F01. The third set-up (see figure 4.1, ❸) was the first such set-up

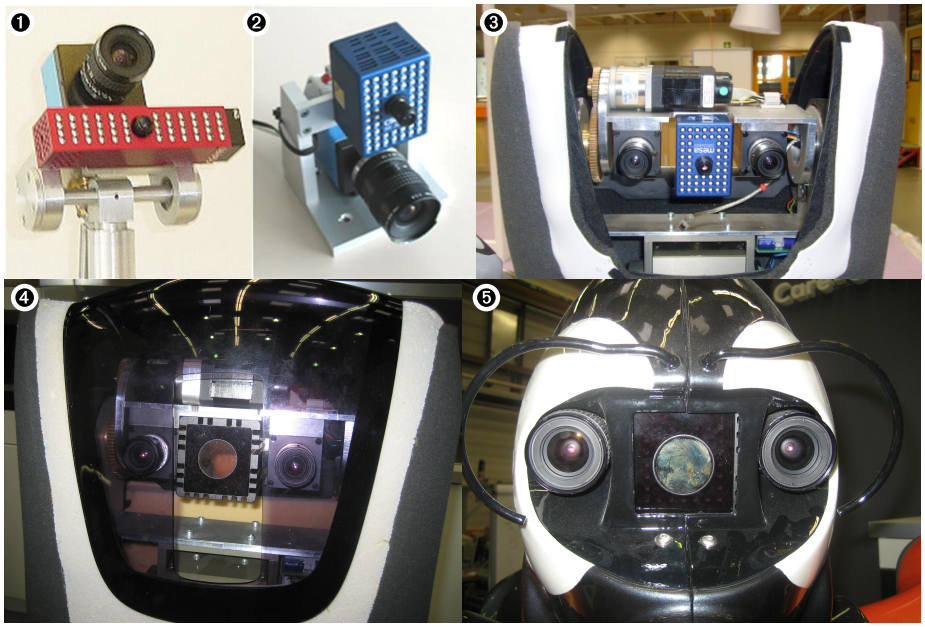

Figure 4.1: Sensor set-ups used. ❶: TOF sensor SR2 (red device, bottom) combined with the color camera DFK42F02 (black/blue device, top). ❷: TOF sensor SR3000 (blue device, top) in conjunction with the color camera DBK31F01 (bottom device). ❸: first set-up integrated on the Care-O-bot® 3 consisting of a TOF sensor SR3100 (center device) and two color cameras PIKE AVT (left, right); only the right color camera is used. ❹: current set-up used on the Care-O-bot® 3 with a range imaging sensor SR4000 (center). ❺: the set-up on the robot platform used in the DESIRE project, see figure 1.1 (right) on page 3.

integrated in the Care-O-bot® 3. It contained a Swiss Ranger 3100 and two color cameras of type PIKE AVT. The more recent set-ups used, shown in figure 4.1 (❹ and ❺), are mounted on the robots shown in figure 1.1 on page 3. Here the newer range imaging sensor SR4000 is used.

4.3 Shared Image Computation

4.3.1 Shared Image Definition

In order to combine the information of a time-of-flight range imaging sensor and a color imaging sensor one can think of a combined sensor setting as described in the previous section and shown in figure 4.1 as a single imaging sensor that directly delivers geometrical coordinates and color intensities for each pixel. In this work the combined image is termed shared image since color and coordinate information *share* the same pixel system. The shared image is notated as a pair of two triples (coordinate and color value matrices):

$$\mathcal{S} = (\mathcal{P}, \mathcal{C}) = ((\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}), (\boldsymbol{R}, \boldsymbol{G}, \boldsymbol{B})) \in \mathbb{R}^{w \times h \times 3 \times 3} \tag{4.1}$$

where $\mathcal{P} = (\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}) \in \mathbb{R}^{w \times h \times 3}$ is used for the coordinate part storing the 3D points (this image is called *coordinate image*). The other part $\mathcal{C} = (\boldsymbol{R}, \boldsymbol{G}, \boldsymbol{B}) \in \mathbb{C}^{w \times h \times 3}$ is a color image that stores the three color components. The symbol $\mathbb{C} \subset \mathbb{R}^3$ is used to denote the color space containing red, green and blue components. Note that in this document large calligraphic letters are used for tuples that consist of more complex types than scalars (otherwise vectors are used). Coordinate and color components are arranged in rectangular matrices of size $w \times h$, where $w \in \mathbb{N}$ and $h \in \mathbb{N}$ are the shared image width and height. It is convenient to define an access to the coordinate parts as vector functions over the image indexes:

$$\boldsymbol{p}_{u,v} = (x_{u,v}, y_{u,v}, z_{u,v}) \in \mathbb{R}^3 \tag{4.2}$$

where $\mathbb{R}^3$ represents the Euclidean or physical space. In analogy, the color part of the shared image can be described with:

$$\boldsymbol{c}_{u,v} = (r_{u,v}, g_{u,v}, b_{u,v}) \in \mathbb{C}. \tag{4.3}$$

Since the optical axes of the sensors are not aligned, the computed shared image can only be an approximation to the reality observed. Errors occur since the two sensors "see" from different perspectives into the scene. In the following subsections, the steps that are needed for the shared image computation are introduced in detail.

4.3.2 Transformations

The goal is to compute shared image $\mathcal{S}$ from the raw images of a range imaging sensor and the color imaging sensor, denoted as $\boldsymbol{T}$ and $\mathcal{C}'$. The *range image* is written as a matrix $\boldsymbol{T} \in \mathbb{R}^{w \times h}$ of size $w \times h$ (i.e. it is set to the shared image size) and contains the

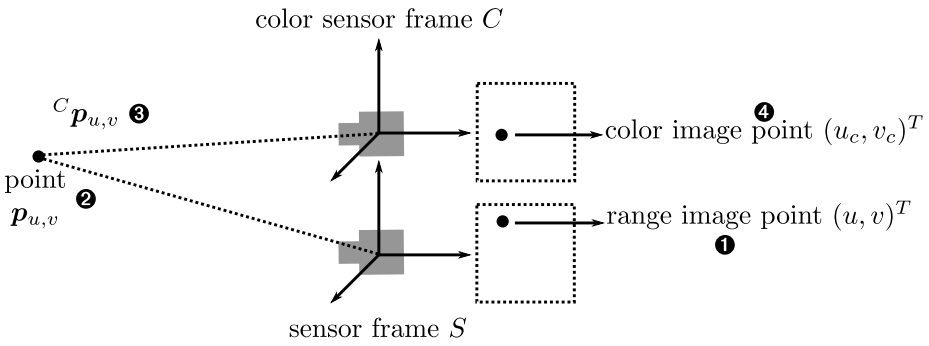

Figure 4.2: The sequence of computations to obtain a shared image. The time-of-flight values $t_{u,v}$ at $(u, v)^T$ (❶) are used to estimate the 3D coordinates $\boldsymbol{p}$ (❷) which are then converted into C (❸) by computing ${}^{C}\boldsymbol{p}_{u,v}$ and mapped onto the color image's pixel system in order to get $(u_c, v_c)^T$ (❹) which then leads to the color value at $(u, v)^T$.

time-of-flight measures $t_{u,v} \in \mathbb{R}$ that are related to the distance of a light reflective object. The raw color image $\mathcal{C}'$ is augmented with the following access function:

$$\boldsymbol{c}'_{u_c,v_c} = (r'_{u_c,v_c}, g'_{u_c,v_c}, b'_{u_c,v_c}) \in \mathbb{C} \tag{4.4}$$

where the primes are used to distinguish this color from the shared image color $\boldsymbol{c}_{u,v}$. In analogy to the shared image, $w_c \in \mathbb{N}$ and $h_c \in \mathbb{N}$ are the width and height of $\mathcal{C}'$.

The sequence of computation is shown in figure 4.2. First, the range imaging sensor reading at the image position $(u, v)^T$ (see figure 4.2, ❶) is used to compute the real coordinates $\boldsymbol{p}_{u,v}$ (figure 4.2, ❷). Note that the sensor system S is used as base system ($S = W$). Therefore, leading superscripts are omitted ${}^{S}\boldsymbol{p}_{u,v} = \boldsymbol{p}_{u,v}$ (the leading superscripts are used to denote in which frame a point is measured, otherwise). Next, the coordinates in the color camera system C, ${}^{C}\boldsymbol{p}_{u,v}$ are computed by coordinate system conversion (figure 4.2, ❸). The result is then projected onto the image plane of the color image to get $(u_c, v_c)^T$ (figure 4.2, ❹). Finally, color information is attached to the color layers of the shared image, $\boldsymbol{c}_{u,v} = \boldsymbol{c}'_{u_c,v_c}$.

In the following, the necessary equations needed to perform these computations are described. The equations used are those from conventional camera projection models as described in e.g. [19] or in the OpenCV manual. Also, the dependency of the time-of-flight measures and the real distance is treated in some other works. In the following the computation sequence is described.

First, an iterative algorithm loops over the image $\boldsymbol{T}$ and reads out the time-of-flight values $t_{u,v}$. Here the term *forward transformation* is used to describe the transformation that maps the internal readings of the range imaging sensor $t_{u,v}$ onto the real coordinates $\boldsymbol{p}_{u,v}$ in the sensor frame S.

In order to perform this transformation, first the relationship between the real Euclidean radius distance $d_{u,v} \in \mathbb{R}$ and the time-of-flight measure $t_{u,v}$ has to be known. The point distance $d \in \mathbb{R}$ of a general point $\boldsymbol{p} \in \mathbb{R}^3$ is given by:

$$d = |\boldsymbol{p}| = \sqrt{x^2 + y^2 + z^2} \tag{4.5}$$

where $|\boldsymbol{p}|$ denotes the absolute value or magnitude of $\boldsymbol{p}$. Only points in the field of view are considered here, i.e. points that lead to valid pixel coordinates after the projection. The point $\boldsymbol{p}$ is projected onto the image sphere (at unit distance) yielding the new point $\boldsymbol{x'} = (x', y')^T \in \mathbb{R}^2$, a projection on the ideal image plane:

$$\boldsymbol{x'} = \begin{pmatrix} x/z \\ y/z \end{pmatrix}. \tag{4.6}$$

Radial distortion that results from the refraction of light beams at the lens surface is usually corrected using a radial polynomial giving a new projection $\boldsymbol{x''} = (x'', y'')^T \in \mathbb{R}^2$:

$$\boldsymbol{x''} = \begin{pmatrix} x' \left(k_3 \left(y'^2 + x'^2 \right)^3 + k_2 \left(y'^2 + x'^2 \right)^2 + k_1 \left(y'^2 + x'^2 \right) + 1 \right) + p_2 \left(y'^2 + 3\,x'^2 \right) + 2\,p_1\,x'\,y' \\ y' \left(k_3 \left(y'^2 + x'^2 \right)^3 + k_2 \left(y'^2 + x'^2 \right)^2 + k_1 \left(y'^2 + x'^2 \right) + 1 \right) + p_1 \left(3\,y'^2 + x'^2 \right) + 2\,p_2\,x'\,y' \end{pmatrix} \tag{4.7}$$

where k_1, k_2, k_3, p_1, and p_2 are the radial and tangential coefficients used for *undistorting* the image. Note that there are more refined models available using 8 distortion parameters. Both options were used in the experiments related to this thesis.

Using the internal parameters of the camera, the image location $(u, v)^T$ can be evaluated:

$$\boldsymbol{u} = \begin{pmatrix} rnd(x''\,f_x + c_x) \\ rnd(y''\,f_y + c_y) \end{pmatrix} \tag{4.8}$$

where f_x and f_y are the focal distances measured in pixels and c_x and c_y the offsets in pixels. Furthermore, $rnd : \mathbb{R} \to \mathbb{N}$, $rnd(x) = \lfloor x + 0.5 \rfloor$ is used as a rounding function since $\boldsymbol{u} \in \mathbb{N}^2$.

To obtain $\boldsymbol{x}_{u,v}$ from $t_{u,v}$, the direction has to be inverted to a forward transformation from the internal range sensor reading to the sensor frame S. Therefore, first ${\boldsymbol{x''}}_{u,v}$ needs to be computed from $\boldsymbol{u}$. This can be done by inverting equation 4.12 which is non-trivial. Another method that can be used is to work with a corrected image position directly that is here denoted by $(u', v')^T \in \mathbb{N}^2$ such that

$$\boldsymbol{p}_{u,v} = \begin{pmatrix} z\,(u' - c_x)/f_x \\ z\,(v' - c_y)/f_y \\ z \end{pmatrix}. \tag{4.9}$$

A corrected image can be computed by a pre-processing step that uses the forward transformation to compute remap matrices. These maps contain for each $(u', v')^T$ in the undistorted image the $(u, v)^T$ in the distorted image (see e.g. [54]).

At a certain pixel location $(u, v)^T$ the distance $d_{u,v}$ is used to compute $z_{u,v}$ with

$$\frac{z_{u,v}}{1} = \frac{d_{u,v}}{{d'}_{u,v}} \tag{4.10}$$

for a normalized image plane with unit focal length. The intrinsic divisor ${d'}_{u,v}$ can be computed for each pixel based on

$${d'}_{u,v} = \sqrt{1 + \left(\frac{u' - c_x}{f_x}\right)^2 + \left(\frac{v' - c_y}{f_y}\right)^2}. \tag{4.11}$$

These values are constant for all $(u, v)^T$. They only need to be computed once.

Finally, there is a relationship between d (real distance) and t (time-of-flight value). During the work here, this was initially modeled as a quadratic function:

$$d = k\,t^2 + l\,t + m \tag{4.12}$$

where k, l and m ($k, l, m \in \mathbb{R}$) are constant parameters that are computed during calibration. Other works suggest a higher order polynomial or a spline approximation in order to achieve a higher precision. The real relationship is a periodic error function. In the SR4000 this function is already calibrated over the full working range. How to use this correction is described in chapter 7.

The aforementioned equations comprise all the ingredients relevant to computing $\boldsymbol{p}_{u,v}$ from $t_{u,v}$.

4.3.3 Color Assignment

Now, the color values $\boldsymbol{c}_{u,v}$ have to be computed for all pixel locations in the shared image. The color camera is assumed to be fully calibrated intrinsically and extrinsically. The extrinsic coordinate system C is measured in S and can be used for the extrinsic transformation. In table 4.1 the relevant parameters are shown. A point ${}^C\boldsymbol{p}$ in the coordinate system of the color imaging sensor can be computed by

$${}^C\boldsymbol{p} = \boldsymbol{R}_C^T\,(\boldsymbol{p} - \boldsymbol{o}_C) \tag{4.13}$$

where $\boldsymbol{R}_C^T$ is the transpose of $\boldsymbol{R}_C$ and $\boldsymbol{o}_C$ the base point of C in S (see table 4.1). The equations to obtain the pixel coordinates $\boldsymbol{u}_c$ are the coordinate transformation and

Constant	Description
$\boldsymbol{R}_C$	Rotation matrix from S (set to zero) to C
$\boldsymbol{o}_C$	Translation vector, base point of C in S
$f_{x\,c}$ and $f_{y\,c}$	Scaling into the image plane
$c_{x\,c}$ and $c_{y\,c}$	Image plane center
$k_{1\,c}$, $k_{2\,c}$, and $k_{3\,c}$	Radial distortion coefficients
$p_{1\,c}$ and $p_{2\,c}$	Tangential distortion coefficients

Table 4.1: Extrinsic and Intrinsic parameters of the color imaging sensor.

equations 4.6, 4.7, and 4.8 with the new point ${}^{C}\boldsymbol{p}$, as argument and the parameters listed in table 4.1.

Typically, in practical settings like those shown in figure 4.1 on page 37, the color images are of much higher resolution (here up to a factor of about eight), than the range images. In order to get a more stable shared image, the initial color images can be down-sampled first. The range image can also be up-sampled in order to keep a good color image resolution. For this thesis a compromise in resolution was chosen at a range resolution of three times the original image size leading to to $w = 528$ and $h = 432$. The color image was down-sampled to $w_c = 640$ and $h_c = 480$. The algorithm 1 pseudo code (see next page) describes the proceeding. The shared image $\mathcal{S}$ is computed based on the source images $\boldsymbol{T}$ (time-of-flight range image) and $\mathcal{C}'$ (original high-resolution color image). Note that the intrinsic and extrinsic parameters are assumed to be given. How these can be computed is the subject of the next section, section 4.4. At step 1 the algorithm starts with resizing the images. This is an optional step. If range imaging sensors of sufficient resolution were available, then this step could be discarded. Next, in step 2 the range image $\boldsymbol{T}$ is undistorted to improve image quality. This measurement also serves for a better alignment of u with $x_{u,v}$ and v with $y_{u,v}$ at a constant $z_{u,v}$. In step 3 the x-, y-, and z-layers of $\mathcal{P}$ are computed. Then, in step 4 two storages need to be initialized. The first, $\boldsymbol{Z}_{buf}$ is used as a z-buffer that keeps track of the closest point so far for which a certain $(u, c)^T$ led to $(u_c, v_c)^T$. There can be more than one pair of $(u, c)^T$ mapping to one and the same $(u_c, v_c)^T$. The z-buffer is used to select one of those pairs, in this case the closest one in z-distance. The z-buffer is initialized with a large z-value, z_{max}, that cannot be exceeded by any time-of-flight value t. The second storage $\mathcal{U}$ stores the pixel positions of $\mathcal{P}$ at the current (closest so far) z-value in the pixel system of the color image.

Algorithm 1 $\mathcal{S} \leftarrow computeSharedImage(\boldsymbol{T}, \mathcal{C}_c)$

1: Optional: resize input images first (enlarge $\boldsymbol{T}$ and downsize $\mathcal{C}'$).
2: Undistort using mapping matrices computed from k_1, k_2, k_3, p_1, p_2, and the focal parameters f_x, f_x, c_x, and c_y.
3: Compute $\mathcal{P} = (\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ from $\boldsymbol{T}$ using the equations 4.6, 4.7, and 4.8 and related parameters.
4: Initialize a z-buffer $\boldsymbol{Z}_{buf}$ with size $w_c \times h_c$ and set all elements to an upper bound z_{max}. Initialize a two-channel position store $\mathcal{U} = (\boldsymbol{U}, \boldsymbol{V})$ with size $w_c \times h_c$ containing the image positions u_{u_c,v_c} and v_{u_c,v_c}. Set all pixels in the color image of the shared image $\mathcal{C}$ to zero (= not valid).
5: **for all** rows $1 \leq v \leq w$ **do**
6: **for all** columns $1 \leq u \leq h$ **do**
7: Read $\boldsymbol{p}_{u,v}$ from $\mathcal{P}$
8: Compute $\boldsymbol{u}_c$ based on the extrinsic and intrinsic transformations in analogy to equations 4.6, 4.7, and 4.8 with the parameters of the color imaging sensor (table 4.1).
9: **if** $\boldsymbol{Z}_{buf\,u_c,v_c} \leq z_{u,v}$ **then**
10: **continue** with step 5
11: **end if**
12: Set $\boldsymbol{c}_{u,v} \leftarrow \boldsymbol{c}_{u_c,v_c}$ (the basic color copy step)
13: **if** $\boldsymbol{Z}_{buf\,u_c,v_c} > z_{u,v}$ **then**
14: $\boldsymbol{c}_{u_{u_c,v_c},v_{u_c,v_c}} \leftarrow 0$ (0=black marks non-valid pixels)
15: **end if**
16: Store current z-value and pixel position $\boldsymbol{Z}_{buf,u_c,v_c} \leftarrow z_{u,v}$, $u_{l\,u_c,v_c} \leftarrow u$, and $v_{l\,u_c,v_c} \leftarrow v$
17: **end for**
18: **end for**
19: Optionally fix the masked regions with some "inpainting" method (see e.g. [72])

The basic loop which performs the computation of $\mathcal{C}$, the color part of the shared image begins at step 5. After reading out $\boldsymbol{p}_{u,v}$ (step 7) and computing $\boldsymbol{u}_c = (u_c, v_c)^T$ (step 8) the algorithm checks whether a certain location $(u_c, v_c)^T$ was visited before and if the corresponding z-value was closer than the previous one (step 9). Only if this is the case, then the color copy operation in step 12 is executed. Otherwise the algorithm continues at step 5. After the color copying to the current location $(u, v)^T$, the last location in the shared image $(u_{u_c,v_c}, v_{u_c,v_c})^T$ that was projected on the same $(u_c, v_c)^T$ is set to zero (non-valid). This solves the problem of multiple projections.

4.3.4 Shared Image Example

In figure 4.3 an image is shown that was computed by the methods introduced. Figure 4.4 shows a 3D rendering from two perspectives. There are errors in the resulting images due to the offset of the optical centers of both sensors. As can be seen in figure 4.4 viewing from the top (right image) large errors can occur at the object's borders. This is natural because the fact that range measures are averaged over the area that is associated with a single pixel.

4.4 Calibration

In the following subsections, two calibration methods that were used during the research work related to this document are described. The first method recovers all parameters of the sensor setting. The second method makes use of the factory pre-calibration of the range imaging sensor. In chapter 7 some results of the achieved precision are reported.

4.4.1 Complete Calibration

To obtain all parameters that are needed the calibration procedure can be based on calibration images recorded from both sensors synchronously. There are different methods implemented in software packages that provide the necessary functionality to detect the cross points of these patterns and to estimate the extrinsic and intrinsic parameters. A precondition for the calibration procedure suggested here is that the range imaging sensor can also deliver a gray-level intensity image called $\boldsymbol{I} \in \mathbb{R}^{w \times h}$. Furthermore, the intensity image of the color image $\boldsymbol{I}_c \in \mathbb{R}^{w_c \times h_c}$ has to be computed.

The complete calibration procedure suggested here is self-contained in the sense that after initial recording of the calibration images, all parameters are computed automatically without the need for additional manual activities. Figure 4.5 shows the five calibration steps required. These are described in the following.

The image sets $\{\boldsymbol{I}_1, \boldsymbol{I}_2, ...\boldsymbol{I}_k\}$ and $\{\boldsymbol{I}_{c1}, \boldsymbol{I}_{c2}, ...\boldsymbol{I}_{ck}\}$ with $\boldsymbol{I}_n$ and $\boldsymbol{I}_{cn}$, $1 \leq n \leq k$ represent the gray-level calibration images. The set $\{\boldsymbol{T}_1, \boldsymbol{T}_2, ..., \boldsymbol{T}_k\}$ contains the range images.

In step ❶ of the procedure shown in figure 4.5, the corners of a chess board pattern are detected. Together with the real 3D coordinates of the points on the pattern that are known *a priori*, three sets are available. The first set:

$$\mathrm{O} = \{\{\boldsymbol{o}_{1,1}, \boldsymbol{o}_{2,1}, ..., \boldsymbol{o}_{l,1}\}_1, \{\boldsymbol{o}_{1,2}, \boldsymbol{o}_{2,2}, ..., \boldsymbol{o}_{l,2}\}_2, ..., \{\boldsymbol{o}_{1,k}, \boldsymbol{o}_{2,k}, ..., \boldsymbol{o}_{l,k}\}_k\} \tag{4.14}$$

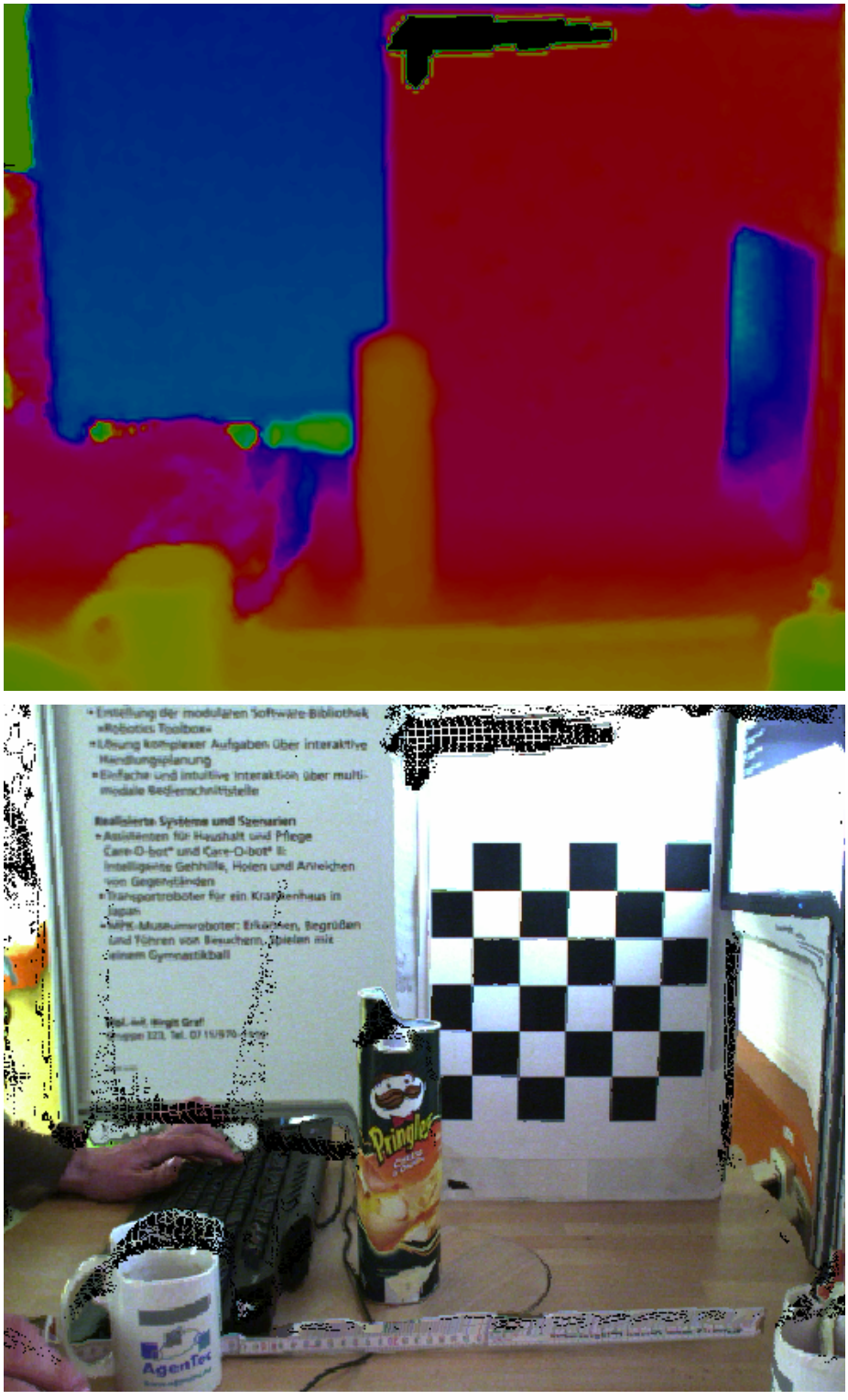

Figure 4.3: Some results of the shared image computation as suggested in this document. Top: the z-channel coded in spectral color. Bottom: the color part of the shared image (doubled areas are marked black).

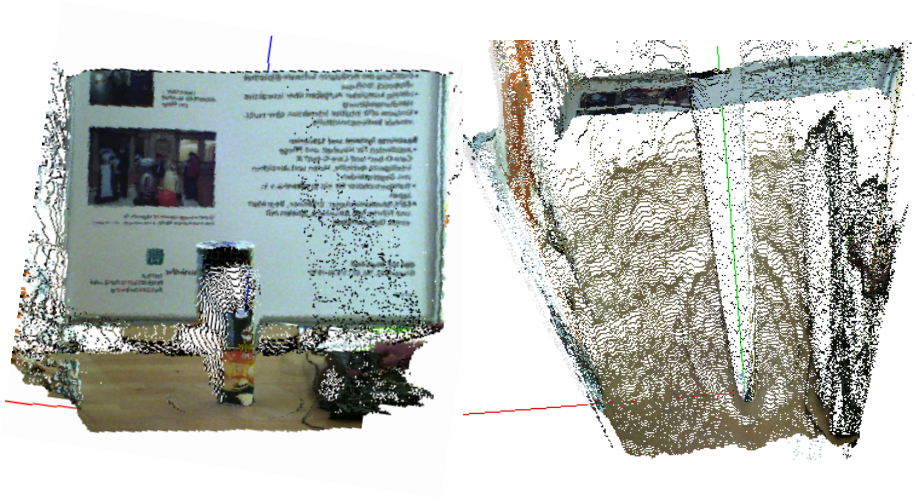

Figure 4.4: 3D views of a Shared Image. Left: front view of a scene containing the cylindrical crisps box. Right: the same scene from a bird's perspective. The red, green, and blue lines in the background are the (x, y, and z) axes of the reference coordinate system.

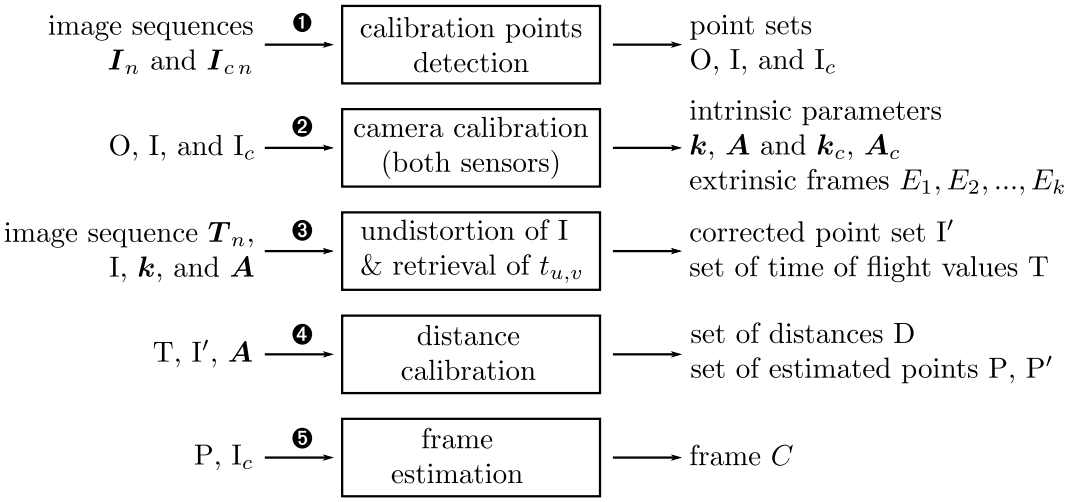

Figure 4.5: Calibration steps. First, the calibration points are detected in the calibration images (❶). Then the cameras are calibrated with a conventional calibration function (❷). After the calibration points are undistorted the time-of-flight values are retrieved (❸), the distance calibration is computed together with a new 3D point set (❹). Finally, the extrinsic frame can be computed (❺).

with $\boldsymbol{o}_{i,j} \in \mathbb{R}^3$ contains the object (rig) coordinates. The second set:

$$\mathrm{I} = \{\{\boldsymbol{i}_{1,1}, \boldsymbol{i}_{2,1}, ..., \boldsymbol{i}_{l,1}\}_1, \{\boldsymbol{i}_{1,2}, \boldsymbol{i}_{2,2}, ..., \boldsymbol{i}_{l,2}\}_2, ..., \{\boldsymbol{i}_{1,k}, \boldsymbol{i}_{2,k}, ..., \boldsymbol{i}_{l,k}\}_k\} \tag{4.15}$$

with $\boldsymbol{i}_{i,j} \in \mathbb{R}^2$ consists of the (interpolated) 2D coordinates of the points in the intensity image of the range imaging sensor. The third set:

$$\mathrm{I}_c = \{\{\boldsymbol{i}_{c\,1,1}, \boldsymbol{i}_{c\,2,1}, ..., \boldsymbol{i}_{c\,l,1}\}_1, \{\boldsymbol{i}_{c\,1,2}, \boldsymbol{i}_{c\,2,2}, ..., \boldsymbol{i}_{c\,l,2}\}_2, ..., \{\boldsymbol{i}_{c\,1,k}, \boldsymbol{i}_{c\,2,k}, ..., \boldsymbol{i}_{c\,l,k}\}_k\} \tag{4.16}$$

contains the 2D image positions in the gray-level image of the color imaging sensor.

After the point sets are available, both cameras can be calibrated using the method of [81] that is e.g. implemented in the OpenCV (see step ❷ in figure 4.5). The following results of this are needed in the further proceeding:

- the intrinsic parameters of the range imaging sensor: $\boldsymbol{k}$ (distortion coefficients) and $\boldsymbol{A}$ (intrinsic matrix containing f_x, f_y, c_x, and c_y)
- the intrinsic parameters of the color imaging sensor: $\boldsymbol{k}_c$ (distortion coefficients) and $\boldsymbol{A}_c$ (intrinsic matrix containing $f_{x\,c}$, $f_{y\,c}$, $c_{x\,c}$, and $c_{y\,c}$)
- the extrinsic frames $\{E_1, E_2, ..., E_k\}$ of each view

Using the distortion coefficients $\boldsymbol{k}$ the points in I are undistorted and the time-of-flight values are read out from $\boldsymbol{T}_n$ at the corrected edge points stored in I′. These and the set of time-of-flight values T are used for the further proceeding. In order to perform these computations the intrinsic parameters are also needed. This is done in step ❸.

Now, an important step is the distance calibration of the range imaging sensor (step ❹ in figure 4.5). The translation and rotation estimates per view can be used to compute a set of real positions, called P, of the points O. For each point in P the distance radius d is given by $d = \sqrt{x^2 + y^2 + z^2}$. The list D of the distance radii can be used to estimate k, l, and m of equation 4.12 by regression. Then, the complete forward transformation can be applied to I′ yielding a point list P′ consisting of the real values the range sensor would deliver with the computed calibration parameters during operation. The quadratic function used here can be replaced by a more complex approximation function in order to model the periodicity of phase angle/distance relationship more precisely.

The last step, step ❺ of the proceeding shown in figure 4.5 is related to the relative frame estimation, i.e. to the computation of an approximation for C. For this the newly computed list P′ can be projected onto the color imaging sensor's plane since this is already calibrated intrinsically. Errors occur due to the (so far) unknown frame C. Based on this data known algorithms can be used to estimate the transformation (see e.g. the implementation of the OpenCV function `solvePnP()`).

4.4.2 Calibration Based on Default Parameters

The range imaging sensor used here (SR4000) is pre-calibrated. The x-, y-, and z-data are computed with calibrated parameters over the complete measurement range. The calibrated model includes all intrinsic parameters and the non-linear relationship of distances and phase measures.

Based on 3D-2D point correspondences, the internal parameters can be reconstructed using known calibration algorithms (see e.g. the implementation of the OpenCV function `calibrateCamera()`). The pairs are directly available from the 3D output images of the range imaging sensor and the pixel locations. In order to calibrate with non-planar data some algorithms need a pre-estimate of the camera matrix. This can be obtained by using two one-dimensional linear regressions that input normalized x- and y-positions and the pixel locations. More details and experimental outcomes of this method are described in chapter 7.

4.5 Summary

In this chapter, a set of tools to obtain combined 3D and color information is discussed. Preference is given to a set-up including a time-of-flight range imaging sensor and a conventional CCD color imaging sensor. This set-up can be constructed easily from two cameras without any changes in hardware. Methods are introduced to calibrate the sensor setting and to compute a so-called shared image $\mathcal{S}$. The calibration sequence is self-contained. It only requires point sets from a calibration rig as used in conventional camera calibration. In the subsequent chapters the shared image will be used as a basis for the realization of the object detector. In chapter 7 precision estimates of the sensor fusion methods are presented.

5 Sample Data Acquisition

In order for robots to learn object models of any kind, segmented object data has to be collected. This chapter introduces *acquisition scenarios* that make full utility of other components "external" to the object detector and of context knowledge in order to provide learning data. Segmentation of object views is done using 3D masking shapes in the coordinate image.

5.1 Motivation and Approach

The construction of a model from sample views seen in the real world is a fundamental necessity for robots intended to serve in environments that contain objects not known to the developer *a priori*. There are two topics that have to be addressed in order to provide such ability: sample data acquisition and model construction. While the latter is addressed through many approaches in machine learning and computer vision, the former is less often subject to research. It is quite hard to perform a first segmentation of an object view if the object is assumed to be unknown. The availability of 3D and color information in one image provides a solution based on 3D masking shapes that can be parametrized. The learning process itself is not completely autonomous at this stage. However, much less effort is required than in a case where in which the operator provides the segmented training views by manual masking. There are different methods that can be implemented on a real robot platform to control the segmentation. In this chapter some of these are presented and discussed. In section 5.2 the basic masking algorithm is described. Then, in section 5.3, so-called acquisition scenarios are introduced, procedures for obtaining some of the parameters that are needed for the segmentation. Section 5.4 contains some illustrative results.

5.2 Range Segmentation

The process of range segmentation can be used generically for different modes of acquisition. Initially, an interesting 3D region is defined in the shared image. After that, the color information of all points within this region are extracted. In order to determine the position and shape of this region, further context information on the object's location and size is needed. The problem is how to separate the data that belong to the object from those that belong to the background. This has to be done without the object model yet established. Here, this process is termed first-time segmentation, describing the need for a segmentation that is performed on object views prior to model construction.

Using the shared image, which contains color and coordinates of image locations, such a first-time segmentation can be based on a range segmentation process, i.e. using 3D information for "cutting out" the color and 3D parts of an object. The specific acquisition scenario needs to provide two types of information per view: a learning frame called L and a segmentation shape defined in this frame, called *mask*. This shape can vary depending on the scenario. In the work related to this thesis mostly spheres and cylinders were used. The 3D measures of the range imaging sensor are unstable at locations where the angle from the measured point vector and the surface normal is large. Therefore, it was decided to incorporate filtering based on this angle. The following algorithm (see pseudo code below) describes the first-time range segmentation. The inputs to the algorithm are the shared image $\mathcal{S}$, the learning table frame L, and the segmentation mask. The output is a segmented shared image called $\mathcal{S}_{seg}$ in which all color and coordinate values outside the mask are set to zero.

Algorithm 2 $\mathcal{S}_{seg} \leftarrow singleViewSegmentation(\mathcal{S}, L, \text{mask})$

1: **for all** rows in $\mathcal{S}$, $1 < v < w$ **do**
2: **for all** columns in $\mathcal{S}$, $1 < u < h$ **do**
3: Only consider points for which $\boldsymbol{p}_{u,v}$ lies within the mask
4: Compute the normal $\boldsymbol{n}_{u,v}$ and reject all points for which $\angle(\boldsymbol{p}_{u,v}, \boldsymbol{n}_{u,v}) > \theta$
5: Set the rejected image location to zero in all image layers
6: **end for**
7: **end for**

The surface normal can be computed with:

$$\boldsymbol{n}_{u,v} = (\boldsymbol{p}_{u+1,v} - \boldsymbol{p}_{u-1,v}) \times (\boldsymbol{p}_{u,v+1} - \boldsymbol{p}_{u,v-1}) \tag{5.1}$$

for all pixels $1 < v < w$ and $1 < u < h$. However, this is not a very stable solution since it depends on a small number of pixels. It can be improved by using larger pixel areas and fitting of a least-squares orthogonal regression plane on the basis of

singular value decomposition (SVD). The normal can be corrected such that it points to the sensor by conditional inversion. This also prevents toggling if the normal is not well defined. Note that the mask can be realized easily by using an in/out equation (e.g. sphere equation). In the experiments related to this document, the shape description chosen is as a cylinder limited in its height. The learning frame is located on a learning table with its x- and y-axes aligned to the table surface. The z-axis is equivalent to the upright normal. The cylinder can be described in terms of radius and maximal and minimal height values. This simple strategy has been used during the work described in this document. For the angle limit, a value of $\theta = 0.6\,\pi$ can be used safely.

The segmentation can also be done on feature point level. The advantage is that features caused by the segmentation edges are suppressed. The segmentation on pixel level is more generic. Such images can be passed to different subsequent processing stages.

5.3 Possible Acquisition Scenarios

In this document, the term acquisition scenario is used to describe the fact that the robot possesses some means of interacting with the environment in order to gather sample data. The following items display the situation:

- A service robot situated in the real-world environment needs to collect the views of a new object in order to be able to detect it during future operations.
- After a kind of initialization (triggering), the robot starts to enter a pre-programmed mode or sub-routine that allows it to acquire the needed information.
- In order to achieve its learning goal, the robot may use functions of the robot control system and/or some context knowledge given by the human operator.
- The collection of object information may be supported by a tutor (human operator) or may be fully automated.
- The successfully collected data are stored and passed on to the model construction part.

Thus, a specific acquisition strategy contains system components and process sequences that describe how the information about the object can be acquired. At this point the description is rather abstract and the itemization above leaves details unanswered. In the next subsections examples of such acquisition scenarios give more detail.

5.3.1 Actuated Robot Motion

Typically, mobile robots are equipped with a localization and mapping system in order to be able to move around in the environment. This means that the robot can estimate its position and orientation in the environment relative to a fixed world coordinate system. The transformation between both is constantly estimated by the localization system. For this, the localization system maintains a map of the environment to estimate the current position of the platform. This map also serves as a basis for goal directed navigation. This external component can be used for the realization of acquisition scenarios. To vary the views of a static object located in the environment the robot may circumscribe the object. Another possibility is to move a robot arm with an in-hand mounting of the sensors. Also, an actuated robot head and/or torso can produce the necessary motion. The complete strategy can be sketched as follows:

1. The robot's user has a new object that the robot will have to detect during future operation.
2. The object is placed on a dedicated location ("learning table").
3. A 3D masking shape (e.g. box or cylinder) is defined in world coordinates surrounding the object.
4. The robot moves around the table ensuring the complete masking shape is contained in the field-of-view.
5. The images are segmented by range segmentation and stored for later model construction.

In this scenario the object itself remains static. This implies that some parts of the object may be covered during the whole process and cannot be learned. However, if the robot moves safely around equipped with some obstacle avoidance then this is a very harmless scenario that can be implemented easily.

5.3.2 Robot-Internal Motion

A simple but important support for sample acquisition is the possibility that the robot can read out its proprioceptive joint sensors (joint angle readings) of the manipulator arm. Therefore, the robot can compute the position of its gripper within the sensor frame S based on kinematic forward transformations. This implies that the position of the appearance of the gripper and a possibly attached object in the shared image is known. Once the object is grasped the robot can move (e.g. rotate or shift) and capture object views. A more sophisticated method could include the opportunity

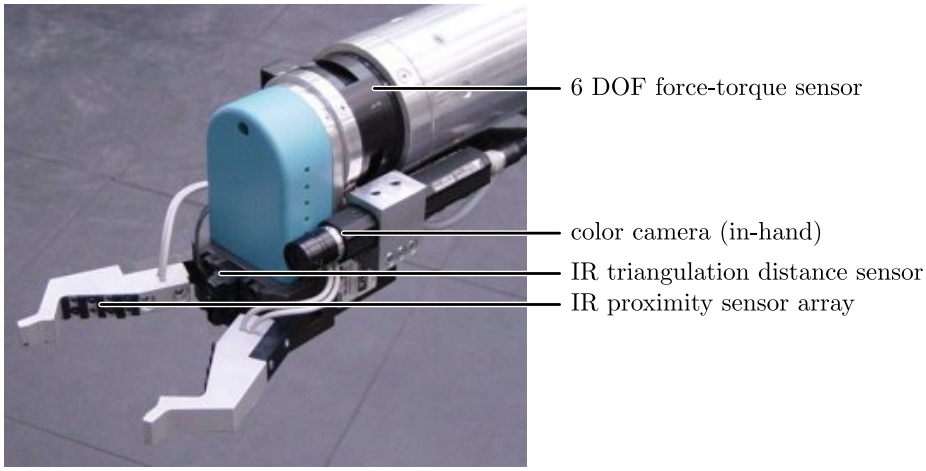

Figure 5.1: Gripper sensors example (Fraunhofer IPA): IR detectors, a camera and and IR distance sensors are integrated in the gripper for triggered object passing.

of releasing the object and grasping it again in another configuration. This could compensate for limitations of the kinematics. There are different possibilities for the robot to retrieve the object at the beginning of the acquisition process:

- Grasping of unknown objects by on-line 3D shape approximation. This method attempts to fit a first approximate shape description into the object. However, the risk of failure is high in general scenarios.
- Object passing. This method is based on the idea that the human operator passes the object to the robot. There can be simple solutions to this based on sensors within the gripper (see figure 5.1 for an example). It is hard to provide a general solution since at this stage the robot does not know valid grasping locations on the object's surface.
- Indirect object passing and shape approximation. The object is first placed on a dedicated place and then taken by the robot. This can be implemented using a tray such as is integrated in the Care-O-bot® 3.

There are further components needed for these modes. Flexible shapes have to be fit into the region in the environment where the object is located and/or object passing has to be realized. The latter is in its simplest version just a trigger to close a force-controlled gripper. Such a system was realized for an older research robot using the in-finger sensors shown in figure 5.1. An example of such an acquisition scenario could be as follows:

1. The user wants the robot to learn a new object.

2. The object is passed to the robot. The human operator ensures that the grasp is stable by putting the object into the gripper correctly.
3. The robot holds the object in a stable manner and moves it within its field-of-view.
4. A suitable masking shape is located around the gripper whose position is known through the kinematics.
5. Different views are segmented by range segmentation and stored for later model construction.

This strategy is desirable since the human is not involved during the generation of sample views. Hence, the robot may take its time to collect many views, possibly stopped by a program part that measures the completeness of the model on-line. A difficulty that cannot be solved easily is that the robot may not know possible grasping points of an unknown object. This can be resolved partly in the scenario where the human places the object into the gripper. Then at least one set of grasping points could be known and more grasps could be learned by exploratory trials. Another problem in this scenario is that the appearances of finger structures of the robot remain in the segmented views. The simple range segmentation strategy is not suitable for removing these. Sophisticated methods could use a 3D model of the robot in order to remove these appearances.

5.3.3 Human Object Demonstration

An interesting possibility is that the human moves the object in order to produce the images needed for training. This can be realized on the basis of human detection and human motion capture algorithms that are available.

Another possibility is to choose the segmented 3D region fix. Then the human operator is responsible to move the object into the masked region. This was implemented in some set-ups related to this work. In figure 5.2 some visualizations of this scenario are shown.

An acquisition strategy example involving human object demonstration may proceed as follows:

1. The user wants the robot to learn a new object.
2. She or he holds the object in the field-of-view of the robot.
3. The masking shape position is computed by an external component that is able to detect and locate the human and especially the human hand that holds the object.

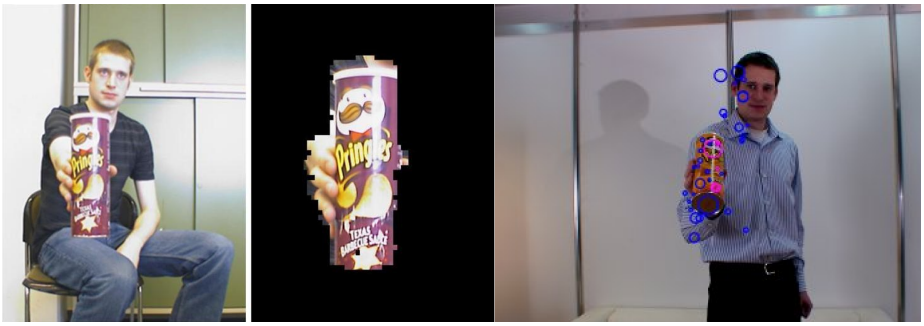

Figure 5.2: Object appearance segmentation based on demonstration. In both cases the segmentation mask is defined by a threshold in z. Left: the pixel values are masked. Right: masking is done on feature point level; purple rings annotate accepted feature points and blue rings rejected ones.

4. Different views are segmented by range segmentation and stored for later model construction.

A drawback of this approach is that the human selects the training views. Valuable object information might be ignored. With additional feedback (e.g. speech) the robot could direct the human operator to show valuable views not seen so far. Parts of the human fingers remain in the segmented images when the simple masking operation is used. These could be removed on the basis of a geometric model of the human.

5.4 Range Segmentation Examples

The range segmentation has been used by the author in two research projects and during further works related to this thesis. This includes both segmenting on pixel level and segmentation based on feature level. Some illustrations are given in figure 5.2. The algorithm for the range segmentation of color appearances has been used for different versions of the object detection system. In order to be able to produce object views with known transformation frames a learning table set-up is used (see figure 5.3). An implemented software enables the user to place the initial learning frame in the center of the rotating table. Since rotation can be controlled precisely, all other frames are also known. A database including five objects was recorded for the purpose of this work. The raw shared images and the segmented versions of the first views are shown in figure 5.4. The time to perform such a segmentation ranges from 20 to 50 ms on the test computer, sufficient for capturing training views fast enough. The examples given here illustrate that the method is capable of producing training images of similar quality as those contained in computer vision databases (e.g. [53] or [21]).

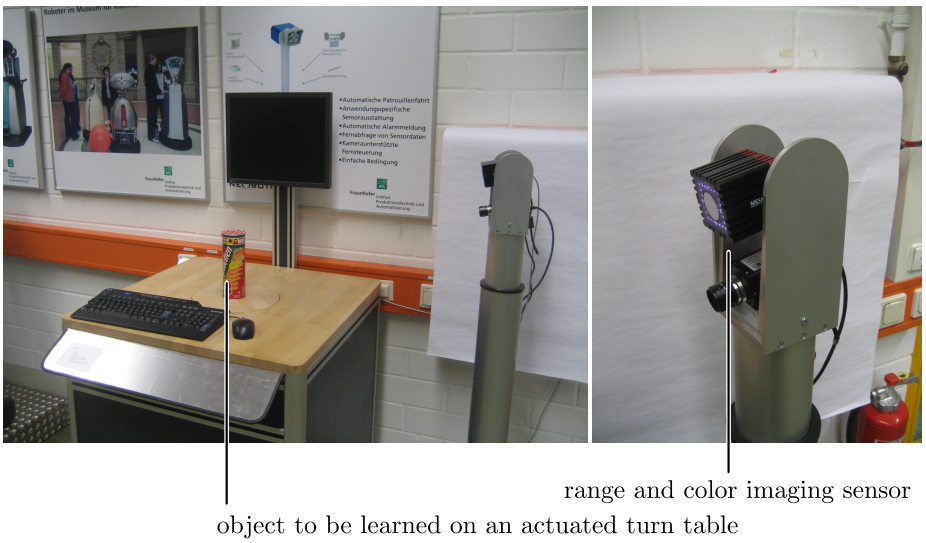

Figure 5.3: Object learning cell used in the experiments.

Figure 5.4: Segmented objects of the test database.

5.5 Summary

This chapter has presented several solutions to sample data acquisition through the introduction of

- range segmentation and
- a number of feasible acquisition scenarios.

The concepts introduced provide realistic opportunities for sample view acquisition in the real world. This has been verified by many real-live tests during the research work related to this thesis.

6 Feature Frame Cloud Extraction and Matching

2D feature point detectors can be applied to the color part of the shared image. The features can be augmented with local coordinate systems based on the coordinate part of the shared image. As a result, a new data representation is available that combines invariant feature attributes with geometric information about position and orientation. This new representation, called *feature frame cloud* can be used for efficient model construction and detection. The computation and matching of feature frame clouds is described in this chapter.

6.1 Motivation and Approach

The aim is to further exploit the properties of the shared image for object detection and object learning. This is done based on an intermediate representation called *feature frame cloud*, a term chosen in analogy to the more common term *point cloud*. The inclusion of *feature* and *frame* account for the fact that the basic elements are not only defined as 3D points. Rather, complete frames (3D position and orientation) are associated with each element, together with discrete feature labels that can be used to distinguish between features.

FFCs can be used to model a large range of objects based on coordinate and color information. They allow for the construction of object models and efficient object detection by simplifying correspondence search. The computation and use of FFCs is based on the following principles:

- Application of the feature point detector to the different image layers (red, green, and blue) of the color part $\mathcal{C}$ of the shared image.
- Discretization of the descriptor vector based on pre-partitioning and clustering descriptor sets based on image identifier, a discrete feature point size measure and the Laplacian sign of the feature point. This information, together with the cluster label, forms a low-dimensional *discrete description key* that is used for fast access.

- Computation of a local coordinate frame associated with each feature point based on the 2D gradient direction and the surface normal in the coordinate part of the shared image.
- Efficient storage of feature points by an associative memory that is accessed by the discrete description key. Each key points to a small-sized list of feature frames that are mapped to that key.
- Once the FFCs are computed they can be used for model construction and detection by an algorithm that estimates the relative frame between two FFCs based on voting on pose consistency.

FFCs are a sparse representation since only feature point locations are included. They contain partial shape and texture information since both influence the construction of the local frame. In the following sections these principles are further described. In section 6.2 the FFC computation is treated. Then follows the object model construction and detection in section 6.3. In chapter 7 experimental results related to these algorithms are presented.

6.2 Feature Frame Cloud Computation

A feature point can be defined as a quintuple:

$$\mathcal{F} = (u, v, s, \phi, \boldsymbol{d}) \in \mathbb{N} \times \mathbb{N} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}^l \tag{6.1}$$

where $1 \leq u \leq w$, $1 \leq v \leq h$, $s \in \mathbb{R}$, $\phi \in [0, .., 2\,\pi)$, and $\boldsymbol{d} \in \mathbb{R}^l$ are the image coordinates, scale, orientation, and descriptor vector of a feature point. The descriptor vector has the dimensionality $l \in \mathbb{N}$. Such features are computed e.g. by applying a SIFT or SURF detector (see also chapter 2). They are defined for single intensity channels (e.g. gray-level images).

Lists of feature points from different color layers of the shared image together with the coordinate information can be converted to a feature frame cloud consisting of access keys to frames lists. One way of how this can be done is presented in this section. Hence, a feature frame cloud is defined as a set of pairs each consisting of a discrete description key vector and a list of frames:

$$\begin{aligned} \mathrm{F} = \{&(\boldsymbol{k}_1, \{F_{1,1}, F_{1,2}..., F_{1,n_1}\}), (\boldsymbol{k}_2, \{F_{2,1}, F_{2,2}..., F_{2,n_2}\}), ... \\ &..., (\boldsymbol{k}_m, \{F_{m,1}, F_{m,2}, ..., F_{m,n_m}\})\}. \end{aligned} \tag{6.2}$$

In this document, sets other than the number spaces are denoted with roman (upright) capitals and lists are written as enumerated sets. F is used to symbolize

the complete feature frame cloud as a list of key/frame list pairs. The other symbols $\boldsymbol{k}_i$ and $F_{i,k}$ with $1 \leq i \leq m$ and $1 \leq k \leq n_i$ (m, $n_i \in \mathbb{N}$) are used for the key vectors and frame list elements respectively.

Such a feature frame cloud F can be implemented and used efficiently on the basis of associative containers that map from discrete vectors to frame lists with low access complexity (e.g. trees or hash maps). If the maximum and minimum values for each key dimension are known *a priori* then also a linear arrangement is possible leading to constant access complexity. For matching purposes, one has to consider only frame pairs of the same key in the two feature frame clouds. Since two correspondences already define a full conversion frame, each pair gives rise to a hypothetical relative frame. For these reasons FFCs can be matched efficiently.

In the following, the basic feature frame cloud extraction algorithm will be described. The algorithm converts the shared image $\mathcal{S}$ into the FFC representation F. The function is termed $getFFC$ and consists of two major parts: feature point extraction and local frame estimation. These steps are further described in the following subsections. The pseudo-code of the proceeding is given below.

Algorithm 3 F $\leftarrow$ $getFFC(\mathcal{S})$

1: Compute the feature point list for the three image layers (RGB) of the color part of the shared image
2: **for all** features in the list **do**
3: Compute the discrete description key $\boldsymbol{k}$ (from image label, size interval, Laplacian sign, and cluster label)
4: Compute the local frame F on the basis of the 2D feature point gradient and 3D surface normal
5: **end for**
6: Append the new feature frame to F

In step 1 the feature points are computed from the color part of the shared image $\mathcal{S}$. The SURF blob detector is used in the software related to this document. However, it can be replaced by any feature point detector that provides the interface described with equation 6.1. In steps 3 and 4 each feature point is mapped on a feature frame. These transformation steps are described in the following subsections. The final step 6 in algorithm 3 is the entry of the new key/frame list pair. If the key is already contained in the FFC then the frame can be appended to the existing frame list directly. Otherwise, a new key/frame list pair has to added to the associative store.

6.2.1 Discrete Descriptor Key Computation

The discrete description key $\boldsymbol{k}$ is an integer list that serves as access address for the frame lists. At this point the concept is rather general. There could be many attributes included as key components (e.g. different feature point types). Ideally, the components are already discrete naturally. Continuous measures need to be quantized first introducing errors at interval borders due to noise. In the approach presented here the key vector composition is chosen to be

$$\boldsymbol{k} = (i, r, l, c) \in \mathbb{N}^4. \tag{6.3}$$

The components (i, r, l, c) are described in the following.

The first entry $i \in \{0, 1, 2\}$ denotes the image channel in which a feature point is detected. The values $0, 1, 2$ are used to enumerate the RGB layers of the color part of the shared image $\mathcal{C}$. The second entry $r \in \mathbb{N}$ is a discrete feature point size derived from the feature points size s using the following formula:

$$r = \left\lfloor \frac{|\boldsymbol{p}_{u,v} - \boldsymbol{p}_{u+rnd(0.5\,s\,\cos\phi),v+rnd(0.5\,s\,\sin\phi)}|}{r_0} \right\rfloor \tag{6.4}$$

where u and v are the feature point pixel coordinates, ϕ the feature point angle, and r_0 a constant that defines the discrete interval size (in this work it is set $r_0 = 10\,mm$). The function rnd is used as a rounding function. The term $0.5\,s$ is the feature point's radius (half the size). Note, that the real 3D size is used as attribute. This can be computed by using the z-distance and the feature point's scale s, which is delivered by the feature point extraction. The third element of $\boldsymbol{k}$ is derived from the Laplacian *sign* of the feature point $l \in \{0, 1\}$. If the sign is -1 then l is set $l = 0$ to describe a local minimum. If the sign is 1 then l is set to $l = 1$ which refers to a local maximum. The fourth and last discrete entry is a discrete *cluster identifier* $c \in \mathbb{N}$, $1 \leq c \leq n_c$ where n_c is the maximum cluster number. This is obtained from a clustering algorithm that is applied to pre-partitioned descriptor sets of the object specific training set F_{train} during learning. This computation is described in the following subsection.

6.2.2 Clustering Feature Point Descriptors

Commonly, feature point detectors such as SIFT, SURF and related approaches deliver a descriptor $\boldsymbol{d} \in \mathbb{R}^l$, where $l \in \mathbb{N}$ is the number of dimensions. These are made invariant against changes in perspective and brightness as well as robust against noise. Usually, gradient statistics (SIFT) or responses to different frequencies (SURF) are used as descriptor vector components. Since the feature frame clouds contain the full geometric relations of the feature points, less discrimination power is required for

a single feature point. The geometric relations in the object model can be used to discriminate feature point groups. The goal is to simplify labeling which is commonly realized by nearest-neighbor searches in the high-dimensional descriptor space. Instead, the descriptor sets are clustered and each training descriptor is associated with one of the clusters. For recognition the descriptor under question is either contained in one of the trained clusters or resides outside all clusters. Then the feature point is rejected.

In the current version of the implementation of the approach presented in this document, *QT clustering* (see [31]) is used since it is very simple, can be fast and it is easy to implement. Also, it does not require a number of clusters to be defined *a priori*. Maybe *EM clustering* could also be used for computing better adapted cluster shapes. Additionally, sub-space reduction methods such as PCA or unsupervised learning (e.g. self-organizing feature maps) could be used as pre-processing strategies. Another possibility is to use one-class SVMs [39] or any other one-class learning machine that is suited. QT clustering is chosen in the work presented for the reasons given above.

For the retrieval of the cluster identifier the prefix $\boldsymbol{k}' = (i, r, l)$ of $\boldsymbol{k}$ is used as a shorter key that partitions all descriptors of all features in F_{train}. This means that the descriptors are pre-partitioned into smaller sets for each different $\boldsymbol{k}'$. Also the cluster identifier retrieval functions are indexed with $\boldsymbol{k}'$. Let $\mathrm{D}_{\boldsymbol{k}'}$ denote a descriptor partition associated with $\boldsymbol{k}'$ and

$$c_{\boldsymbol{k}'}(\boldsymbol{d}) = \begin{cases} i \in \{1, 2, ..., n_c\} \text{ if } \boldsymbol{d} \text{ maps to cluster } i \\ 0, \text{ otherwise.} \end{cases} \tag{6.5}$$

This function retrieves a cluster identifier for any descriptor $\boldsymbol{d}$ given $\boldsymbol{k}'$ ranging from 1 to n_c, the largest possible cluster identifier. The function $c_{\boldsymbol{k}'}$ is only available after the basic clustering process is executed. In order to provide $c_{\boldsymbol{k}'}$ based on the QT clustering the conventional QT algorithm can be reformulated (see algorithm 4 below).

Algorithm 4 $\mathrm{C}_{\boldsymbol{k}'} \leftarrow$ *getClusters*$(\mathrm{D}_{\boldsymbol{k}'})$

1: Initialize first cluster prototype $\boldsymbol{c}_1 \leftarrow \boldsymbol{d}_1$ and a counter $k \leftarrow 0$
2: **while** $\mathrm{D}_{\boldsymbol{k}'}$ is not empty **do**
3: Compute all distances between the elements in $\mathrm{D}_{\boldsymbol{k}'}$
4: Choose the sample $\boldsymbol{d}_{max}$, that contains the most members in a sphere given through t_{cls}
5: Remove $\boldsymbol{d}_{max}$ and all members of the cluster defined by $\boldsymbol{d}_{max}$ and t_{cls} from $\mathrm{D}_{\boldsymbol{k}'}$
6: Set $\boldsymbol{c}_k \leftarrow \boldsymbol{d}_{max}$ and $k \leftarrow k + 1$
7: **end while**

The input of algorithm 4 is a set of descriptors $\mathrm{D}_{\boldsymbol{k}'} = \{d_1, ..., d_n\}$ and the output a reduced set of cluster prototypes $\mathrm{C}_{\boldsymbol{k}'} = \{c_1, ..., c_m\}, m \leq n$. The parameter t_{cls} controls the reduction. If the threshold is small then the cluster prototype set equals the original descriptor set. If in turn t_{cls} is chosen very large then only one prototype remains.

The size of the set $\mathrm{D}_{\boldsymbol{k}'}$ is successively decreased by removing clusters that are given by samples that have many neighbors within a sphere defined by t_{cls}. Note that the pseudo code is only printed for description of the algorithm's output. An efficient implementation does not need to compute all the distances more than once. When using QT clustering, retrieval of cluster labels can be quickly by only evaluating the Euclidean distance up to a dimension where the sum of squared component differences exceeds t_{cls}^2. The cluster labels can be of different scopes. In the approach followed here they are object specific, i.e. the sets $\mathrm{D}_{\boldsymbol{k}'}$ are derived from an individual object image collection. Also a global cluster set could be learned incrementally. If a new object is introduced then some descriptors are covered by clusters learned already and others add new clusters. In the following, the cluster label is called c with $c = c_{\boldsymbol{k}'}(\boldsymbol{d})$, where $\boldsymbol{d}$ is the descriptor in question. Experiments that were performed during this work showed that the chosen components of $\boldsymbol{k}'$ are stable enough for object detection in typical scenes.

6.2.3 Local Frame Construction

Estimating a local frame at the 3D location of the feature point becomes possible by using the surface normal $\boldsymbol{n}_{u,v}$ (see e.g. equation 5.1 on page 50) at the feature point location $(u, v)^T$ as the first axis. The next axis can be computed from the gradient direction ϕ on the basis of a normal vector describing the gradient plane:

$$\boldsymbol{n}_g = (cos(\phi + \frac{\pi}{4}), sin(\phi + \frac{\pi}{4}), 0)^T. \tag{6.6}$$

The intersection of the gradient plane and the surface normal describes how the feature point is oriented on the surface of the object. Here, the surface is approximated using a local plane defined by $\boldsymbol{n}_{u,v}$. This is the first axis of the local frame. The second axis that is called $\boldsymbol{g}$ (gradient axis) is chosen to align with direction of the gradient. This can be computed by:

$$\boldsymbol{g} = \boldsymbol{n}_{u,v} \times \boldsymbol{n}_g. \tag{6.7}$$

The third axis called h is the one orthogonal to $\boldsymbol{n}_{u,v}$ and $\boldsymbol{g}$ given by $\boldsymbol{h} = \boldsymbol{g} \times \boldsymbol{n}_{u,v}$. Finally, the local frame F can be constructed with $\boldsymbol{i}_F = \boldsymbol{n}_{u,v}$, $\boldsymbol{j}_F = \boldsymbol{g}$, and $\boldsymbol{k}_F = \boldsymbol{h}$. The construction is depicted in figure 6.1.

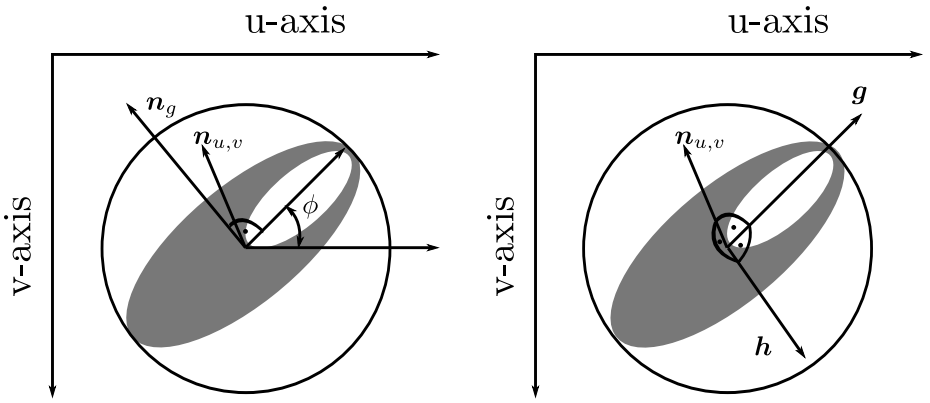

Figure 6.1: The computation of a feature frame from a feature point. The gray ellipse shows an artificially created circular blob feature on an inclined plane. The white disc within the feature area serves for a strongly defined intensity gradient. With the intensity gradient and the surface normal available a local frame can be constructed. The black ring shows the detected SURF feature with a gradient angle ϕ. A normal vector $\boldsymbol{n}_g$ is used together with the surface normal $\boldsymbol{n}_{u,v}$ to construct the gradient vector $\boldsymbol{g}$. Finally, a third vector $\boldsymbol{h}$ is constructed which is orthogonal to $\boldsymbol{g}$ and $\boldsymbol{h}$.

6.2.4 Example

A real-world example of feature frame cloud computation is presented in the following. In figure 6.2 a scene view is shown as shared image (top/left) and as FFC (bottom). Also a zoom-in (top/right) shows the feature frames. The colors of the balls visualize the discrete key vectors as separate colors. The local frames are drawn using colored (red, green, and blue) lines for the frame axes (x, y, and z). The computation of this FFC was performed without the cluster label (since this is object dependent).

6.3 Model Construction and Detection

The computation of the FFCs can be applied to the segmented images in S_{train} for model construction or to scene images during detection. The resulting training set of FFCs is denoted as F_{train}. In this section, the goal is to combine the partial object views into an object model and to use that model for detection. Both processes, object model construction and detection are based on the same method of matching feature frame clouds as described in the following subsection, subsection 6.3.1. Then follow subsections 6.3.2 and 6.3.3, which explain how model construction and detection can be realized based on the frame estimation algorithm.

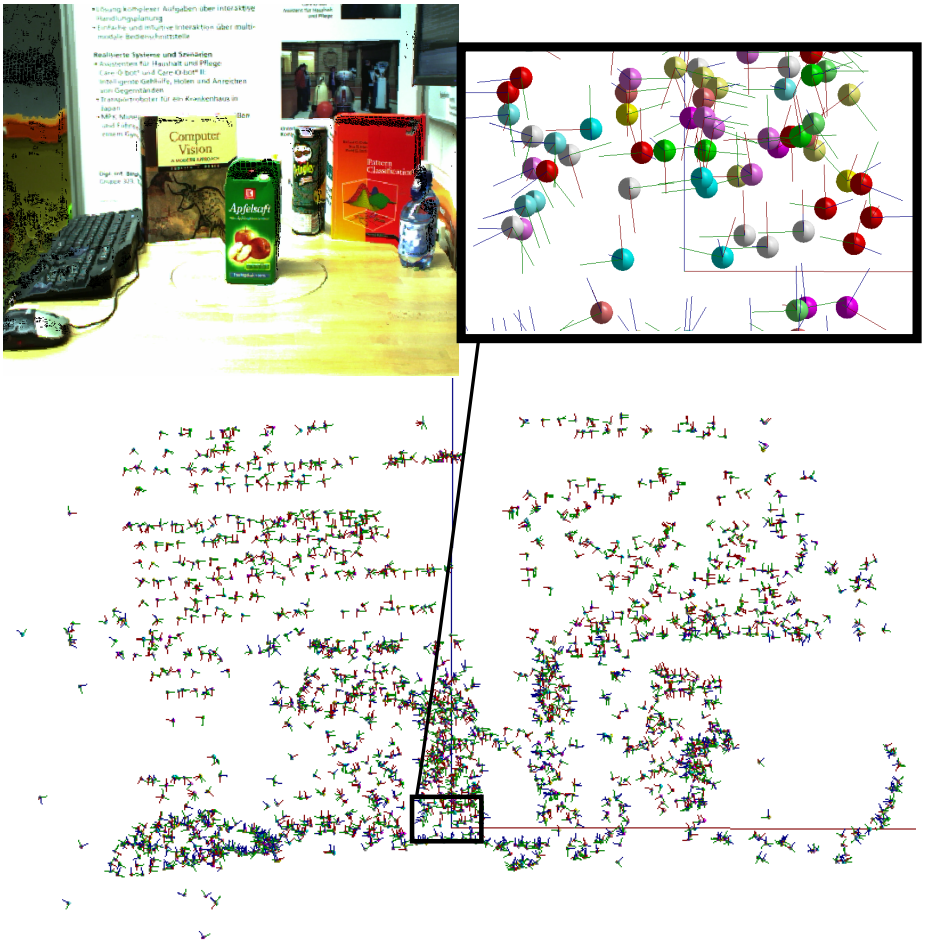

Figure 6.2: An example of FFC computation. Top/left: the original scene shared image. Bottom: the scene as feature frame cloud. Top/right: a zoom into the feature frame cloud. The ball's color denotes the discrete key vector and the three colored lines are the frame's axes (red = x, green = y, and blue = z).

6.3.1 Feature Frame Cloud Matching

Feature frame cloud matching is the process of estimating the relative frame between two FFCs (source and argument FFC) such that if the frame is applied as coordinate transformation to the argument cloud many local feature frames with equivalent keys have the same position and orientation in both clouds. The solution given here does this estimation based on density evaluation in a voting space that contains possible origin frames G_h (the subscript $h \in \mathbb{N}$ is used as a hypothesis enumerator) of an argument FFC called G measured in the base FFC F.

Let F be the source feature frame cloud as defined by equation 6.2 and G a second "argument" feature frame cloud that is contained in F and G is the unknown base frame of G in F. An assumption is that there is some overlap, i.e. both clouds contain some common feature frames. For the description here G is composed as follows:

$$\begin{aligned} \mathrm{G} = \{(\boldsymbol{l}_1, \{G_{1,1}, G_{1,2}..., G_{1,l_1}\}), (\boldsymbol{l}_2, \{G_{2,1}, G_{2,2}..., G_{2,l_2}\}), ... \\ ..., (\boldsymbol{l}_k, \{G_{k,1}, G_{k,2}, ..., G_{k,l_k}\})\}. \end{aligned} \tag{6.8}$$

Assume that the origin system of F is chosen as world frame (zero offset and neutral rotation). To estimate G it is possible to compute and store all possible hypothesis frames H that result if the correspondence assumption is asserted to be true for each pair. All hypotheses can be enumerated by listing all possible tuples $(F_{i,k}, G_{j,l})_h$, which have matching keys $\boldsymbol{k}_i = \boldsymbol{l}_j$. Assume there are n such hypotheses such that $1 \leq h \leq n$.

In the following the computation of a single hypothesis H is described by using transform equations (see [14]). For simplicity the symbols $F_{i,k}$ and $G_{j,l}$ are replaced by F and G, respectively. In figure 6.3 the construction is shown. The first FFC, F, contains a frame F that is known relative to the world frame W by the transformation ${}^{W}_{F}\boldsymbol{T}$. In the second FFC, G, the frame G is assumed to be a correspondence to F. Therefore,

$${}^{W}_{F}\boldsymbol{T} = {}^{H}_{G}\boldsymbol{T} \tag{6.9}$$

holds. Furthermore, there are two ways to obtain G: through the direct measure ${}^{W}_{G}\boldsymbol{T}$ or indirectly over the unknown H. This can be expressed as follows:

$${}^{W}_{G}\boldsymbol{T} = {}^{W}_{H}\boldsymbol{T}\,{}^{H}_{G}\boldsymbol{T}. \tag{6.10}$$

Equations 6.9 and 6.10 can be combined and solved for the frame H, yielding

$${}^{W}_{H}\boldsymbol{T} = {}^{W}_{G}\boldsymbol{T}\,{}^{W}_{F}\boldsymbol{T}^{-1} \tag{6.11}$$

which computes the hypothesis. Now let the set H be a hypotheses set that is a collection of all G_h

$$\mathrm{H} = \{H_1, H_2, ..., H_n\}. \tag{6.12}$$

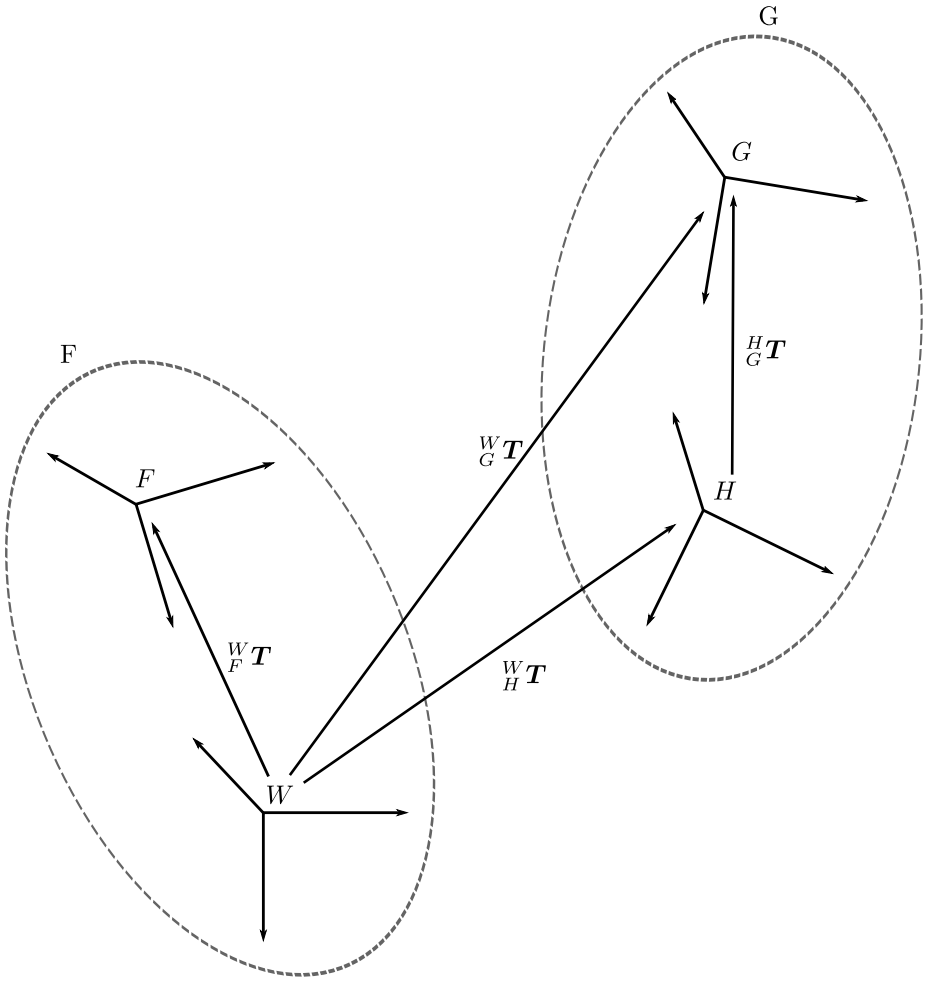

Figure 6.3: Dependencies of coordinate frames in F and G (see text for the descriptions of the symbols).

An example of such a set for a real scene and a matched object model is shown in figure 6.4. It is possible to retrieve an estimate of the frame H by evaluating the empirical density around single frames in H. In figure 6.4 this is shown by the (manually annotated) frames in the left and right image.

A density for each frame in H can be computed by iterating over all possible $n\,(n-1)$ pairs and computing a *frame distance* $d(H_i, H_j)$. Such a frame distance can e.g. be computed by the suggestion in [25] that is based on the addition of translation and rotation distances. The distance measure used in this work is described in the appendix. For each H_i a counting function $c(H_i)$ can be computed denoting the sum counts in a vicinity defined by a threshold $t_F \in \mathbb{R}$. This count could also be weighted by a general weight based on the key occurrences in both clouds. The maximum

Figure 6.4: This figure shows the set of hypotheses H for an object model of a book. Left: the scene (color part of the shared image) containing the book. Right: the dots represent the x- and y-coordinates of the base point of each frame hypothesis. The red, green, and blue lines in the left and right image represent the object reference frame (left image) and the result frame (right) that are manually annotated in this figure.

frame found is called H^*. To find this frame more efficiently, the voting space is arranged as a set of cubes defined by an interval size for the translation components and a fix angle for the rotation components. The storage of all votes is commonly called *accumulator*. During the voting phase the address of the maximum cube is continuously updated and the subset contained in the cube with maximum density is called H*. These considerations are summarized in algorithm 5 (see below). It estimates the frame H^* on the basis of two feature frame clouds P and Q.

In the software used for the tests in this document the translation bin sizes are set to 50 mm and the rotation sizes to $\pi/4$. The components are the three (x, y, and z) parts of the translation vector $\boldsymbol{o}_F$ of a frame F and three angles. The first angle is the rotation angle α_F and the remaining are the sphere angles θ_F and ϕ_F of the rotation axis $\boldsymbol{u}_F$ of the frame F. Per vote there are actually two cells occupied, one with the bin coordinates resulting from a $floor$ operation and one from rounding per dimension. The frames are stored collectively in their corresponding cubes. It follows that the precision of the result depends on the error of one hypothesis only. The threshold of the distance function is chosen at $t_F = 25$ and would correspond to millimeters if there was no rotational difference.

The complete algorithm is described by the pseudo code of algorithm 5 (see next page). The worst case complexity of the first part of the algorithm (the voting phase) is at most quadratic, i.e. if there was only one matching key in both clouds. In the best case the complexity is linear (if there is only one frame per matching key). Since the density evaluation in the second part is based on all pairs of hypotheses, it is quadratic on the numbers of votes. The discretization of the accumulator helps overcome this problem in practice. An assumption here is that the best voting cube

also contains the densest hypothesis. This is not generally the case and thus it is a weakness of this approach.

Algorithm 5 $H^* \leftarrow getFrame(\mathrm{F}, \mathrm{G})$

1: **for all** local frame pairs $(F_{i,k}, G_{j,l})_h$ with $\boldsymbol{k}_i = \boldsymbol{l}_j$ **do**
2: Compute a single hypothesis H based on equation 6.11.
3: Store the pair into a discretely sub-divided accumulator data structure containing H (e.g. based on an associative memory structure).
4: Update a cube specific count by adding w_h
5: Update the maximum cube's address
6: **end for**
7: Select a sub-set H^* of the cube in the discrete voting space with the highest density. Assume there are n elements in H'.
8: **for all** possible pairs (H_i, H_j) in H^* with $1 \leq i \leq n$ and $1 \leq j \leq n$ and $i < j$ **do**
9: Compute a frame distance $d_F(H_i, H_j)$.
10: Update the density count $c_(H_i)$ if $c_(H_i) < t_F$).
11: **end for**
12: Choose the kth frame H^* with maximum $c_w(H_k)$ as result.

An advantage is that the algorithm is very robust to noise and can be shown empirically to detect very small subsets of object model feature points in large scene FFCs. Another advantage is that the concept is open to include other helping knowledge such as weights based on the feature key distributions. Such weights could also be based on general feature point type statistics or context knowledge. Another advantage of this algorithm is that the time for estimation could be computed on the key counts before actually running the algorithm. An upper bound of the computation time could be sent to the client. This could be desired for time critical applications.

6.3.2 Object Model Construction

An object model is a FFC called M that has to be constructed from the segmented training views S_{train}. Object model construction consists of three distinct parts:

1. The descriptor database has to be constructed in order to be able to compute the discrete descriptor key completely with the cluster identifier c. The input to this are all descriptors of the FFCs in F_{train}. Note that F_{train} has constant cluster labels set to 0 since the clusters are not established at this time. Hence, effectively only the prefix $\boldsymbol{k}'$ is used as key.

2. If the acquisition strategy provides the relative frame from each view to the next (e.g. through internal robot motion) then these frames are used to write all FFCs together in one remaining FFC, called model M. Otherwise, if the relative transformations are not known then an appropriate technique to combine all views (one possible method will be described in this subsection) is applied in order to get an approximate model M.

3. *Cluster* and *centralize* the remaining object model (see appendix). Clustering can be done to remove copies of one and the same feature frame. The term *centralizing* is used here to describe the process of aligning the remaining model FFC around the neutral frame in order to get the reference frame O "into" the objects and minimize rotation errors. A method that can compute a centralized set of frames is described in the appendix of this document.

These steps comprise the following algorithm called $getModel$. The input to this algorithm is the set of segmented shared images S_{train}. The output is a model FFC called M. The pseudo code is given below.

Algorithm 6 $M = getModel(S_{train})$

Initialize a list of feature frame clouds F_{train}
for all shared images in S_{train}, $1 \leq i \leq n$ **do**
  Compute $F \leftarrow getFFC(\mathcal{S}_i)$ with the cluster identifier set to zero $c = 0$.
  Append F to F_{train}
  Append the descriptors to the corresponding descriptor set $D_{\boldsymbol{k}'}$
end for
Train the object dependent cluster prototype set C using $getClusters$ based on all clouds in F_{train}
for all feature points $1 \leq j \leq n$ in all FFCs of F_{train} **do**
  Compute the cluster identifier c and update the key
end for
Combine all views into one model FFC M
Cluster and centralize M

If the acquisition scenario provides precise frames then it is strongly recommended to make use of them. Then the fusion step (step 11) is only a copying and coordinate conversion process. If not then the fusion of all partial view FFCs into one model is equivalent to solving a global optimization problem based on an appropriately chosen fitness function. The output is a list of frames that describes each view relative to a common reference frame. However, such a global solution is very expensive.

During the work here several simpler fusion methods have been tried. Some were only based on frame estimation between FFCs. It was found that such methods are not adequate due to the problem of sequential error propagation. It was observed that it is necessary to also match views that are not direct neighbors in the sequence. However, it is difficult to decide algorithmically which pairs have to be matched. The best simple method found during the work presented here computes some approximate object models hierarchically. This is shown in figure 6.5. The shapes resulting from this strategy are not accurate but compact enough to achieve similar recognition rates as with the known transformations. Finally, (in step 12) the model FFC is clustered to remove multiple occurrences of one and the same feature point and centralized. Here also this clustering is done by QT clustering based on the frame difference function (see appendix). Those frames that lie centrally in dense regions are kept while their neighbors are suppressed. The centralization is based on computing the centroid of frame base points and average frame axes (see also the description in the appendix).

6.3.3 Object Model Detection

The object detection and pose estimation algorithm has to find the object reference frame O in the sensor system S based on local frames that lie on the object's surface: this is shown in figure 6.6. The resulting detection frame is denoted with $D = {}^{S}O$ and its approximation $\hat{D}$. Detection and pose estimation is based on the application of the $getFrame$ function. Let S be the scene FFC filtered by the descriptor clusters of the object model O. Then $\hat{D}$ can be computed using:

$$\hat{D} \leftarrow getFrame(\mathrm{S}, \mathrm{M}). \tag{6.13}$$

6.4 Summary

This chapter has introduced the notion of feature frame clouds, a representation that leads to efficient matching. Feature frame clouds can model the current scene view, partial object views and complete object models. They allow for efficient access due to low-dimensional keys and contain local object shape information. A possible

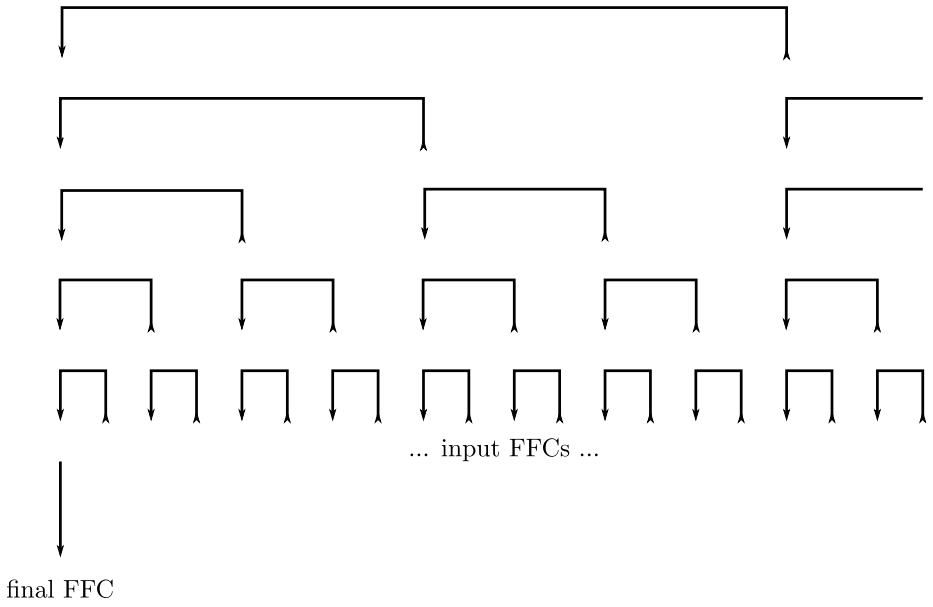

Figure 6.5: Hierarchical registration of single view FFCs into one object model. Each arrow represents a frame estimation and importing operation of two views into one. The backend of the arrows (right side) are the source views and the arrow heads (left side) point to the destination views which contain the feature frames of both views after termination.

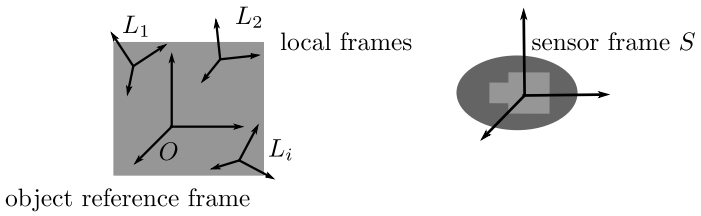

Figure 6.6: The detection of an object model based on detections of local frames. In the object model each local frame is known relative to the world system. Therefore, a correct correspondence votes on O in S.

registration technique to compute approximate object models from partial views if the conversion frames from one sample view to the next are unknown is presented, too. In chapter 7 some experiments and results on parameter settings and matching are given.

7 Evaluation

This chapter contains some experimental results and discussions related to the approach introduced in this document. These experiments, tests and evaluations concentrate on the algorithmic components of the approach. This chapter is organized as follows.

- First, the combined sensor calibration and shared image computation is treated. It is shown that the range and color image can be aligned accurately.
- The second part is devoted to the FFC construction and matching algorithms. It is shown how the local frames can be constructed and they lead to a reduction in the search complexity. Different matching methods are discussed and reasons are given to support the approach that evaluates densities in the frame space.
- This is followed by the third part that treats descriptor matching based on the simple thresholding method and the additional components of the discrete access keys.
- The fourth part contains some tests of the complete system. Tests include learning with known learning frames, learning with the registration procedure, and detection in complex scenes.

These parts are presented in the following sections. Note that these measurements are based on software that is not optimized for speed. Large parts could be implemented based on parallel algorithms and further improvements could be based on pre-computation of reused parameters as well as avoidance of unnecessary copies of data involved.

7.1 Combined Sensor

The image fusion algorithm (algorithm 1 on page 43) for the shared image computation and the calibration procedures have been implemented and tested in C++. In the following, representative results are reported.

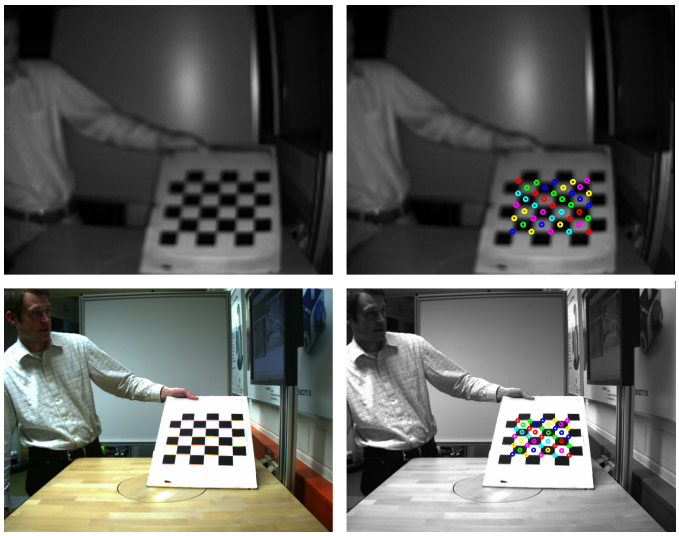

Figure 7.1: Examples of calibration images. Top: resized image of the range imaging sensor (left) with detected corners and center points (right). Bottom: the intensity image of the color imaging sensor (left) and the detected corners (right).

7.1.1 Calibration of the Full Sensor Setting

Calibration Image Recording and Selection

For the detection of calibration points the function `cvFindChessBoardCorners()` from OpenCV is used. Note that in addition to the corner points, the interpolated centers of the squares are also used in order to provide different brightness variants to the distance calibration of the range imaging sensor. In figure 7.1 examples of calibration images are shown. The original 176×144 sized intensity images of the range imaging sensor are enlarged by a factor of three using the OpenCV function `cvResize()`. The intensity image of the range imaging sensor is enhanced in contrast in order to improve the corner detection.

The original 1280×960 sized color images of the color imaging sensor are resized down to 640×480 pixels. The color images are converted into gray-level images by using the OpenCV function `cvCvtColor()`.

In a representative test 343 images were taken and 137 remained after exclusion of images in which not all corners could be detected or in which corners were detected at false locations (manually filtered). After the image recording and selection there were 5617 calibration points available. The recorded images and point sets are used in the following paragraphs.

Intrinsic Calibration of Both Imaging Sensors

First, the calibration images are used to calibrate the intrinsic parameters $\boldsymbol{A}$, $\boldsymbol{k}$, $\boldsymbol{A}_c$, and $\boldsymbol{d}_c$ of the two sensors. Here, the OpenCV function `cvCalibrateCamera2()`, which implements a method based on [81] is used. The function is called with an option to fix the pixel aspect ratio at 1. This is possible since the pixels of the range imaging sensor selected here are squares. The 5617 calibration points that were recorded result in a pixel error of 1.33 pixels (standard deviation). For the color imaging sensor the pixel error on the calibration points used is 0.63 pixels. The low pixel resolution of the range imaging sensor can explain the large error.

Corrected Calibration Points

The method used here undistorts the calibration range images $\boldsymbol{T}_n$ to re-detect corrected corner points during calibration. Range images are also undistorted during operation as a first step. For both purposes maps are are computed with the OpenCV function `cvGetUndistortMaps()` for more efficient remapping. The undistorting step for the range images recorded here takes $38\,ms$ in average measured over the original 343 calibration images with non-optimized code on the test computer (Pentium 4, 3 GHz with 2 GB memory). The set with 137 points used here contains only 124 images in which all corners are detected correctly in the undistorted images, resulting in 5084 calibration points.

Distance Calibration

A least-squares quadratic regression) was applied to the 5084 calibration points to obtain the parameters k, l, m. The regression maps the values in T to the real distances D. The overall metric error of the forward transformation is computed by measuring the difference between the 3D coordinates predicted with the intrinsic parameters contained in set P′ and those derived from the extrinsic frames E_n. The overall error on the point set used here is $17.5\,mm$. Also the x-, y-, and z-components errors were evaluated, as well as the error in the distance radius d. The outcome is that the distance error in d is $17.4\,mm$. The error in x is $2.2\,mm$ while the error in y is $2.3\,mm$. The relatively large error in d directly causes the large error in z. The error distribution in of $d - \hat{d}$ ($\hat{d}$ denotes the estimated values) in millimeters is shown in figure 7.2.

In summary, the final distance error of the overall sensor calibration resides at $17.4\,mm$ on the 5084 calibrations points set. Note that only a quadratic approximation is used for the relationship of phase angle and real distance. Better approximations could improve the error. Another weakness here is that the “ground truth”

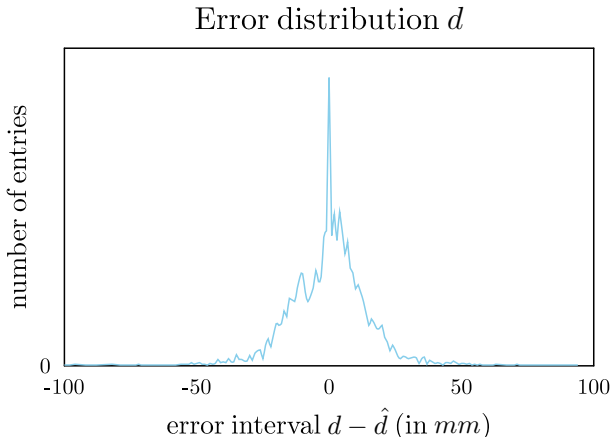

Figure 7.2: Remaining distance error distribution: the histogram shows the number of $d - \hat{d}$ entries per mm-sized bucket. The peak is at 0 with 363 entries.

distance data are provided by the chess pattern matching. These measurements can be erroneous, too.

Extrinsic Frame Estimation

The frame C of the color sensor has to be estimated as part of the calibration. The related part of the software used here takes the forward projected range imaging sensor measures P' and minimizes the error on the corner/center locations in the intensity images of the color imaging sensor. The implementation uses a simple greedy search. The search steps are decreased if no improvement in any dimension is possible. The resulting parameters on the calibration point set used here contain a translation offset in y of about $-94\,mm$. This is roughly the real difference in height between the cameras in the set-up used. The pixel error is, at 4.3 pixels, higher than those resulting from the single intrinsic camera calibrations. However, considering that this error is over the full chain of sensors it may be acceptable. The error distributions of $u_c - \hat{u}_c$ and $v_c - \hat{v}_c$ in the pixel system are shown in figure 7.3 and figure 7.4.

An example of the output of the procedures introduced is shown in figure 4.3 on page 45.

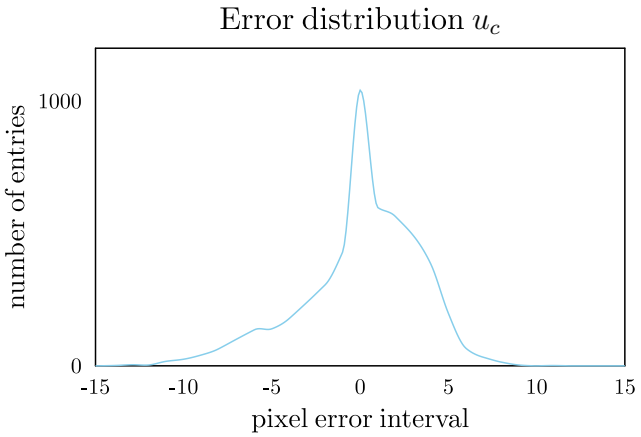

Figure 7.3: Pixel error distribution $u_c - \hat{u}_c$ of the full sensor calibration.

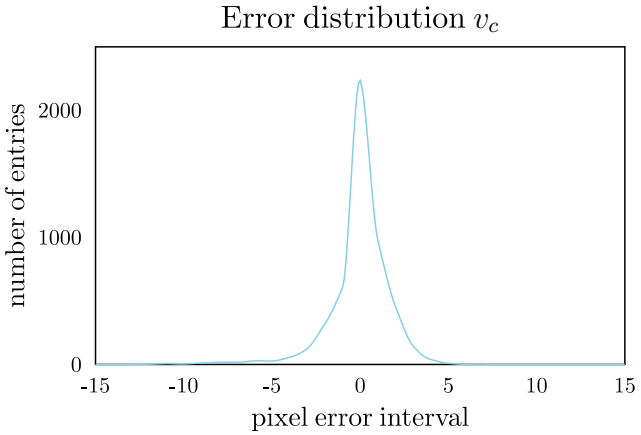

Figure 7.4: Pixel error distribution $v_c - \hat{v}_c$ of the full sensor calibration.

7.1.2 Calibration based on Default Parameters

In the following a further experiment on the sensor fusion is reported. It is based on the range camera's pre-calibration. The advantage compared to the full calibration approach that is that the camera already compensates for the non-linear relationship of the time-of-flight phase shift measure and the real distance.

First, the camera parameters of the time-of-flight camera are reconstructed in order to compute the map for the range image distortion correction and resizing. This can be done by using the measured 3D coordinates based on some artificial range values that can be passed to the sensor together with the 2D image positions. In order to run the calibration based on the OpenCV function `calibrateCamera()` with all

points in one view (i.e. as calibration grid containing non-planar points) some initial estimates of the intrinsic parameters must be given. This is done by using half the image width and half the image height as estimates for (c_x, c_y). As focal parameters (f_x, f_y) the scale parameters from two one-dimensional linear regression lines are used that are fit to the pairs $(x/z, u)$ and $(y/z, v)$. An experiment was conducted based on a raw range image filled with random range values. At a resolution of 176×144 this led to 24344 2D/3D point pairs. These could reconstruct the internal parameters with an error of 0.00075 pixels.

Using these intrinsic parameters the range image can be corrected for distortion. A test based on this correction shows the fitting accuracy of the range camera. A coordinate image and the gray-level image of the range imaging sensor are used. Both are corrected for distortion. A chessboard pattern is detected in the raw gray-level image and the 3D coordinates are picked at interpolated 2D image positions. The pattern consists of a planar grid 9×6 points surrounding squares of $0.04\,m$. The matching algorithm from [32] is used to estimate the pose. All point distances of the detected points and their real ground-truth values in the detected pose are used to compute a 3D error. In a representative test run at a distance of $0.9\,m$ between camera and pattern, the matching error result was $0.00187\,m$, which is in millimeter range.

The shared image calibration to be evaluated is based on the same software used in the previous tests. It is a realization of the method described in section 4.4.2. Again, the initial estimates of $(c_{x\,c}, c_{y\,c})$ are based on the image size of 640×480 and the focal values based on two linear regressions. Since in this setting the sensors are approximately mounted at the same z-coordinate and y-coordinate, the scaling parameters based on the pairs $((x/z, u_c), (y/z, v_c))$ can be used as initial estimates. These are provided by calibration pattern detection in the gray-level images of both cameras. A test was performed using about 4000 calibration point pairs of the form $(x, y, z) \mapsto (u_c, v, c)$ and the reprojected error was 0.7 pixels in the higher color camera resolution of 640×480. The error distributions of $u_c - \hat{u}_c$ and $v_c - \hat{v}_c$ (the hat denotes the computed values) in the pixel system are shown in figure 7.5 and figure 7.6.

An example image produced with this calibration is shown in figure 7.7. It includes fixing of the missing regions that are caused by the fact that the optical systems of both cameras are spatially separated. The computation of the shared image can be done fast. On the demo computer (single core 2.8 GHz) used here it takes about $30\,ms$ without the region fixing. However, it is not an optimized implementation. The resizing of the range image and the correction for distortion could be combined in one step. The fixing operation is computationally expensive and expands the computation time to about $200\,ms$. However, the advantage is that feature points that appear due to the strong edges around the masked region are avoided. The mask for the in-painting algorithm is computed after the shared image construction. Both

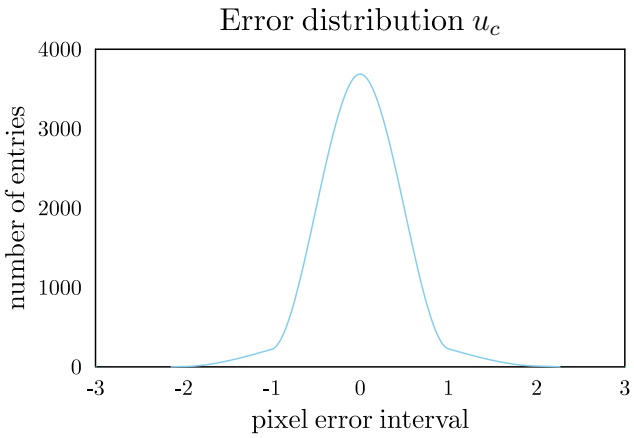

Figure 7.5: Pixel error distribution $u_c - \hat{u}_c$ (interpolated) of the calibration based on the default parameters on the camera.

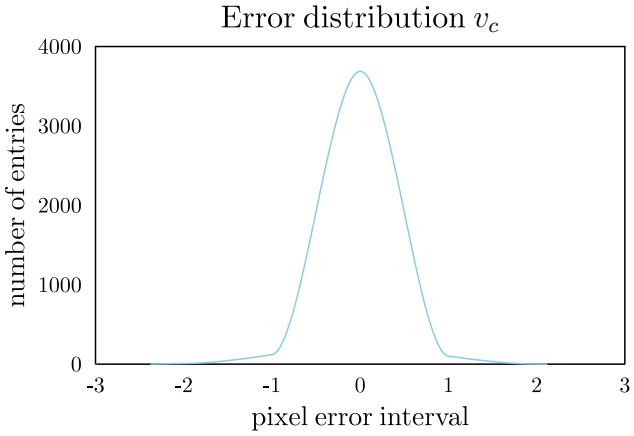

Figure 7.6: Pixel error distribution $v_c - \hat{v}_c$ (interpolated) of the calibration based on the default parameters on the camera.

Figure 7.7: Shared image with precise calibration and region fixing (see e.g. the bottom rows that are interpolated).

steps could be combined in order to increase computation speed.

The basic outcome of this experiment is that both images can be combined with a shared color image with sub-pixel accuracy in the larger 640×480 image size. The calibration based on the parameters set by the manufacturer is more precise than the full calibration as reported in subsection 7.1.1 and is expected to keep good precision over the full measurement range. This can be explained by the fact that the non-linear phase angle/distance error is corrected on the camera directly and by the use of a better extrinsic frame estimation algorithm.

7.2 FFC Computation and Matching

7.2.1 FFC Computation

An important question about the feature frame construction is whether there are frames that are stable from different views. Here, stable means that their relative coordinate system to the object center system is fixed among varying views. An example for such a frame construction is shown with a visualization in figure 7.8.

The frames in figure 7.8 are built based on the calibration used in section 7.1. The image region for normal construction based on a least square plane fitting is 21×21 pixels. The feature point threshold is chosen to be very high to isolate these features from surrounding features in order to make them visible. The top row in the image shows the construction with only the 2D gradient (i.e. the normal is set

Figure 7.8: Depiction of the frame construction. Top row: an object at 0°and 30°where the frames are constructed only using the 2D gradient and setting the normal to $\boldsymbol{n} = (0, 0, -1)$ (i.e. it does not change the other frame vectors). Bottom row: the same views as in the top row but including the normal direction in the frame construction. The frames are drawn with RGB encoding the x-, y-, and z-axes. The blue z-axes in the bottom row show the surface normals. The red circles denote the feature label.

constant) while the bottom row shows the full construction. It can be seen that the frame construction is possible – the gradient and normal computation can lead to frames that are attached locally. The blue colored normal stays orthogonal on the surface and becomes visible at higher degrees of rotation. However, due to errors in the normal computation and due to the fact that some features are radially invariant (e.g. round blobs), it is possible that a single frame becomes instable.

7.2.2 Pose Detection

A test was done based on the segmented unlabeled feature points of the object shown in figure 7.8 using 36 views at increments of 10°. The labeling was disabled to only concentrate on the geometric matching in this test. Each view was matched with its predecessor and the result frame was compared to the known frame between the two views. A threshold $t_{surf} = 300$ was chosen and the size of the voting cells was set to $10\,cm$ (translation) and 180° (rotation). The feature points were filtered based on the normals with an angle threshold $\theta = 0.55\,\pi$.

The difference between the true transformations and the computed ones were compared using difference metric described in [25] that is the sum of the translation

distance (distance of the base vectors of the two frames) and a number that measures the distance of rotations based on the trace of the rotation matrices product. In this experiment the final error in translation was 1.13 cm and 7.86°. This means that based on the settings described above and the feature frames constructed, a detection precision in centimeter range and in the range of 10 degrees is possible if the object contains enough features in each view.

There are different factors that influence the possible accuracy:

- First, the range data measured with the time-of-flight camera are affected by depth errors caused by so-called multi path reflections.
- Second, to detect stable normals, the computation must be based on large image regions. But this also induces a smoothing effect and the correct rotation of the object is damped.
- Third, in this experiment all feature-pairs in object and scene are considered and the feature count is high per view (about 1500 features in average). This leads to a large number of noisy frames in the voting cell containing the correct result. Due to the unstable rotational parameters of the constructed frame rather large voting cell sizes have to be chosen. Larger cell sizes decrease pose accuracy.

Therefore, this approach would immediately improve with better techniques to compute the normals in the range image. In the experiments on registration (section 7.4.1) of the different object views a rotation increment of 5°is used in order to improve the frame estimation between the training views. Another opportunity for improvements could be to compute object dependent or scene dependent (or both) voting cell sizes. However, this possibility is not explored here.

In a second experiment using the FFC-ICP algorithm the precision could be further improved up to 4.6° and 0.92 mm by taking into account 100 nearest frame neighbors. This means that the information content of the views is possibly limited and that the large voting cell sizes needed decrease the accuracy. However, in further experiments, described below, the ICP variant was not found to be able to match views with high differences in rotation and translation. This is due to the basic assumption that near frames are matches. Therefore, it is possible to combine the two matching algorithms in order to improve the pose accuracy. The feature point correspondences that led to a vote in the densest voting cell could be stored. All the feature points pairs that belong to this cell could be passed to the ICP algorithm. However, this opportunity is not further explored in this thesis.

7.2.3 Lowering the Search Complexity

The possibility of such a construction leads to a decrease in matching complexity. Although due to the unstable frame construction a relatively high feature count is necessary to compute more accurate results, the advantage is that the single pose votes can be computed based on only two correspondences.

Imagine that the computation of the frames would require 3 points. In the positive case in which 3 features in the scene setting match with 3 features on the object, this means there are $k = n(n-1)(n-2)$ (n is the number of features in the scene) possible variations to consider. The variations need to be used for this because the order is important during the frame construction (see e.g. [32]). It follows that the complexity to find the correct matching triple in the scene is $O(n^3)$ in big-O notation.

In the worse case it is not known that a triple in the model is visible in the scene. Therefore, all possible triples in the object model $l = m(m-1)(m-2)$ (m is the number of features in the object model) have to be considered, each $k = n(n-1)(n-2)$ many times. For simplicity let $m = n$ (i.e. object model and scene have equivalent feature count). Then the additional tests required lead to a complexity of $O(n^6)$.

Reductions of this complexity are possible by including further 3D information. For example, if two 3D points are used and one additional surface normal then the normal and the difference vector of the two 3D positions can be used to compute the frame axes. The complexity can be reduced to $O(n^2)$ in the positive case and to $O(n^4)$ in the case where it is not known if the features are in the scene.

The local frame construction proposed here leads to the linear $O(n)$ complexity in the positive case and to $O(n^2)$ in the worse case. This leads to an opportunity for algorithms that use many features, possibly of different types, still efficiently. This search reduction is seen as a clear achievement of this thesis.

Processing more features still efficiently also allows less precision in the feature labeling. This leads to the possibility of the simple thresholding approach for descriptor labeling. Now the "descriptor model" of the object only depends on the single object alone and not on models based on sets of objects. This is seen as an advancement compared to existing methods that use comparisons of nearest neighbor distances between descriptor sets of different objects. If the difference of the feature descriptor of the object and the nearest descriptor in the scene is compared to all the differences of that scene feature and the closest neighbor of all other objects in the object set, then this leads to a complexity $O(k)$, where k is the number of objects. Here, this is $O(1)$ since only the descriptors of one object need to be processed.

Please note that these theoretical aspects related to search reduction do not mean that the implementations used in this work readily outperform implementations of

existing approaches based on multiple feature matches in order to compute one frame hypothesis. Due to the currently unstable local frame construction and non-optimized software this is likely not to be the case. A large voting cell size has to be used which causes errors. However, it shows the potential of the approach related to scalability in the number of features and matching speed.

7.2.4 Matching Algorithms

The reduction in search complexity can be explored in different search schemes. The ICP and RANSAC methods converted to working with FFCs were implemented and compared to the density-based approach.

The ICP FFC version takes a number of nearest frame neighbors and computes a combined conversion frame between model and scene and applies it to the model until it aligns with the object in the scene. The algorithm is fast when there are enough feature labels such that the probability is high that the nearest neighbors in that label are real correspondences. When the number of features with equivalent labels is higher and the distance increases between model and scene FFC, then the algorithm fails. This was verified with a number of tests not printed here.

For each iteration the RANSAC FFC version guesses a (i.e. only *one*) correspondence pair of the same label, computes the relative frame and verifies the frame. A frame distance measure (see in the appendix, chapter 8.2) can be used to compute the number of inliers as a quality value of the pose hypothesis. A disadvantage of the approach is that either the iterations or the quality measure must be used to terminate. In both cases it is unknown whether the best possible pose has been found or if the algorithm needed many unnecessary iterations. Also, the verification of each hypothesis is an extra computational cost compared to the density-based approach. This complexity could be reduced further by organizing the feature frames into some fast accessible structure such as a tree. However, this does not eliminate the need for the confirmation step. It only makes it faster.

During the work related to this document tests were conducted based on randomly created FFCs. Source and target FFCs were precisely equivalent, but the target FFCs were transformed by a randomly chosen relative frame. Each test cycle consisted of 100 trials with different FFCs randomly created and at every 10 cycles the relative frame was varied.

In the test shown in figure 7.9 the feature count starts at 100 and proceeds to 500. There is only one key, i.e. the features are not discriminative. Though this might seem to be a rather extreme test, note that with 20000 features and 40 keys (all equal count) the complexity is equivalent. Such a scenario is realistic in the

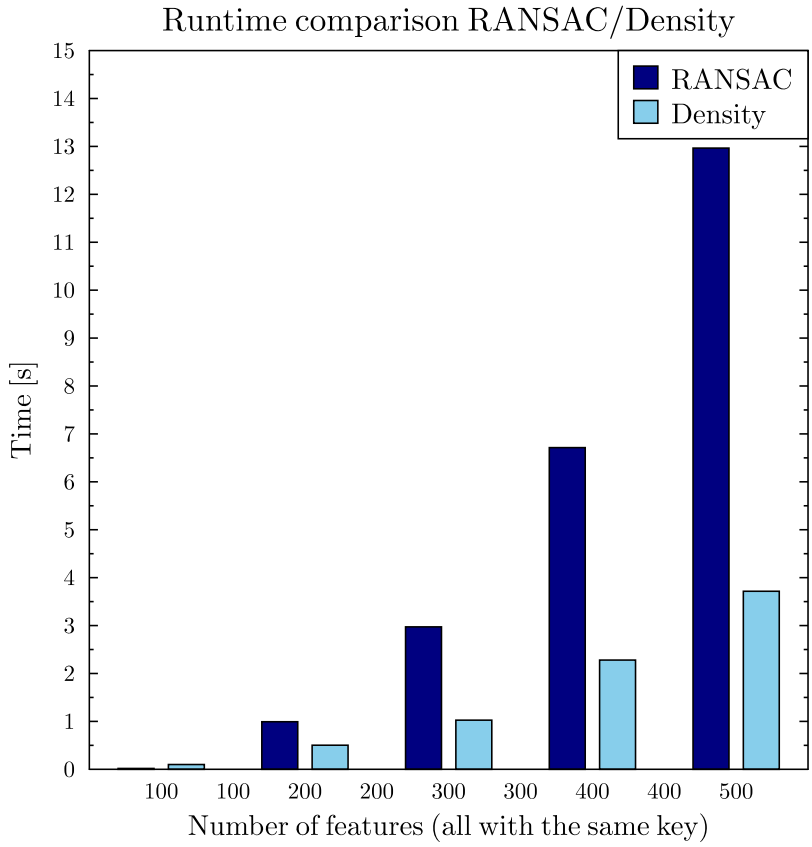

Figure 7.9: A comparison of the RANSAC algorithm with the with the density-based approach. All features have an equivalent key. In this extreme situation the voting algorithm outperforms the RANSAC approach.

case in which fewer discriminative features are used or in which the scene contains multiple instances of the same object. It is shown in figure 7.9 that the density-based method is faster in these tests. Other tests with 100 to 500 features and 10 keys were performed, too. Here the RANSAC was faster until a feature count of 400.

It is interesting that many state-of-the-art approaches build on the RANSAC algorithm even though it takes a long computation time if there are many features that can not be distinguished well. The success of RANSAC can only be explained by the robust but complex indexing procedures used on descriptor level. With the FFC representation and the density-based approach it is possible to cope with not as discriminative features (which may be computed faster due to simpler descriptors).

The fact that a small subset of features can be used to find approximate object poses quickly means that:

- Scenes may contain more features (e.g. due a wider field-of-view).
- The number of object features may be smaller (e.g. due to featureless objects or object coverage).
- More features can be used (e.g. lower thresholds or different types).

7.3 Descriptor Tests

The SURF features were selected in this work since the results on the tests proposed in [4] are superior to other feature points in terms of accepted performance measures. Additionally, since the integral image is used, SURF is very fast. SURF has the option to choose between a 64 and 128 dimensional descriptor. Here, the 64 dimensional version was chosen. An experiment was conducted to observe the stability of the SURF feature points in the color part of the shared images manually. This is shown in figure 7.10. The green points in figure 7.10 denote the correct matches (i.e. which belong to the object). Only the first (0°) view was used to teach the cluster database with the 64 dimensional descriptor vectors. The test parameters were adapted manually. It can be seen that up to 45° (last image at the bottom/right in figure 7.10) there are still many green points in the region of the apple juice box.

Systematic tests were done using ten of the objects shown in figure 7.7. The parameter $t_{surf} = 500$ was chosen. This value was found to be adequate in some tests not reported here. The features of all ten objects, placed on a rotation table, were computed in order to measure the following error, which is a combination of the false alarm rate and miss rate:

$$e_d = 0.5 \left(\frac{n_{fa}}{n_{neg}} + \frac{n_m}{n_{pos}} \right) \tag{7.1}$$

Figure 7.10: A visualization of the SURF descriptor stability. The first 9 images of the object database (object: juice, 5° rotation) are shown. Only the descriptors of the first image are taught and then matched with the subsequent images. Stable descriptors (green points) exist up to about 40° rotation

where $n_{fa} \in \mathbb{N}$ is the number of false alarms, $n_{neg} \in \mathbb{N}$ is the number of feature points outside the object region, $n_m \in \mathbb{N}$ is the number of misses, and $n_{pos} \in \mathbb{N}$ is the number of feature points inside the object region. The separation between whether a feature point is located inside or outside the object region is based on the segmentation masking cylinder described in section 5.2. In order to measure the generalization capacity over different poses only every second view (+20°) was taken for training and all views (+10°) were taken for testing.

The smallest error for gray-level images only based on the cluster label found here is $e_d = 0.182$ with $n_m = 0.269$ and $n_{fa} = 0.095$ at a threshold $t_{surf} = 0.24$. If separate RGB channels are used, the rates are slightly better leading to $n_m = 0.265$ and $n_{fa} = 0.090$. The addition of the Laplacian sign and the size labels led to $n_m = 0.268$ and $n_{fa} = 0.084$.

It follows that the augmentations of the discrete keys do not drastically improve these rates. However, the value of the separate RGB features is that generally more features are available and that objects of different color but same or similar surface

texture can be distinguished. Since the features are stored in separate lists, the increase of the number of features does not increase the search complexity quadratically if new keys are added. As noted before, the advantage of using FFCs is that higher feature counts can still be processed efficiently.

The remaining keys (Laplacian sign and size interval) do not increase the errors but allow for better splitting of the feature frames into shorter lists resulting in faster matching. The additional value related to error rates of the augmented components discussed here may possibly increase with larger number of objects.

Due to the efficient matching, the high remaining miss rates can be compensated with a larger number of features. This results in a higher probability that there are at least a few matches. Larger feature counts can be achieved by changing the feature point parameters or by using feature extractors of different type.

7.4 System Tests

In this section, results of complete object learning and matching based on FFCs are reported and discussed. The tests are based on software implementations that differ in some respects from the implementations used in the previous evaluations. Most importantly, the sensor fusion used during the time of database recording did not achieve the accuracy of the results in subsection 7.1.2 and image fixing was not applied. In these tests the normal is calculated from a small image region (3×3 windows) and the depth data is not precise due to multi path reflections. Furthermore, the implementations are not optimized for speed. Still, the tests show that object learning and object detection are possible in complex scenes with the methods introduced. The tests are split into three parts:

- First tests treat learning and detection with known learning frames. This means that the object model FFCs are simply constructed by applying the known transformations at learning time to each view. This technique relates to those scenarios in chapter 5 in which the transformations are known from the context (e.g. the robot holds and rotates the object). Detection results are established by counting matches based on the segmentation shape and evaluating sums of frame distances to the original transformations that are known. Three types of detection tests are reported:
  - Matching with the segmented training views is used to compare the re-estimated learning pose with the true conversion frame. The tests show the correctness of the implementation and some problems that occur with too low feature counts.

  - Matching with the non-segmented training views is performed in a similar way. However, the segmentation is disabled. The test shows what detection rates are achievable in a complex scene setting, in which most of the features belong to the background. These tests measure the impact of the background features on pose accuracy.
  - Matching in random scene images is tested by counting in the 2D projection of the result frame position is located within some annotated rectangle in the test image. The rectangles are annotated manually. These tests do not measure the performance in pose precision accurately. However, they show whether approximate pose detection is possible in a qualitative sense.

- Further tests focus on the more difficult case of learning and detection with unknown learning frames. The database used here is equivalent to the "known frames" case but the transformations between the training views are not fed to the model construction algorithm. Not all detection tests of the known frame case are repeated as the exact pose information is lost after registration. Therefore, these tests are only based on counting if the detected poses match with the correct image region.
- Finally, the third part describes some impressions gained during live tests with a real robot platform.

These tests are described in the following subsections.

7.4.1 Learning and Detection with Known Learning Frames

Learning

Here, some results of the model construction algorithms are reported. First tests were based on the known learning frames. All partial views were converted into the learning frames, thus constructing the real object model FFC by importing all transformed views into one FFC. Table 7.1 shows numbers are that characterize the model construction process.

The learning times for the objects are very low (about $47\,s$ for all of the five objects with 72 views per object). This can be explained by the fact that the SURF extractor is very fast and there are no cost intensive registration operations following. The resulting model FFCs are shown in figure 7.11.

It can be seen that there are relatively many keys for one object even after clustering. This means that relatively many features have to be extracted in order to raise the number of features with equivalent keys between separate object views.

Object	No Keys (1)	No Keys (2)	Learning time (s)
book Duda	10084	3502	9.42
book Forsyth	14398	6442	15.14
bottle	2047	958	4.47
juice	12209	4532	11.45
crisps	9494	4281	10.17

Table 7.1: Numbers for the learned object models with known frames for a representative parameter setting. "No Keys (1)" are the number of keys before clustering and "No Keys (2)" after clustering.

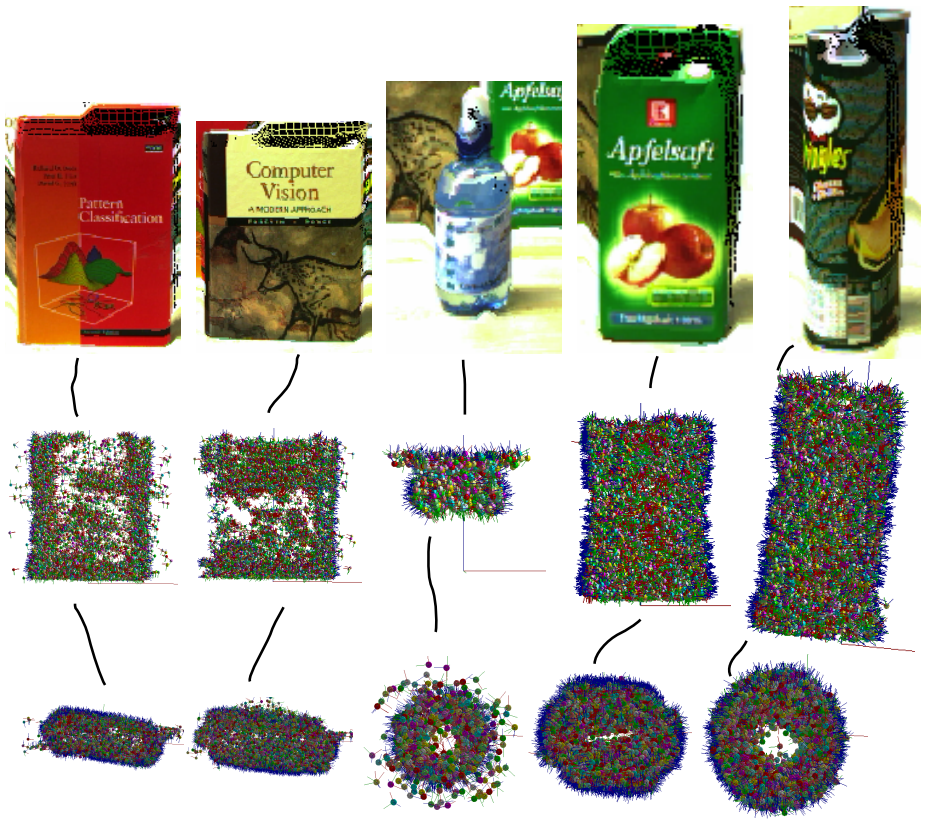

Figure 7.11: Learned models FFCs for the objects in the database. From left to right there is the same order as in table 7.1 from top to bottom. The upper row shows the front views and the lower row the top views.

Object models of the quality reported here approximately match with the real shape. This is only possible using the filtering mechanism based on the angle between pixel position and normal as described in chapter 5. Otherwise, the models tend to be much larger due to pixels that mix foreground and background. The models obtained are shown to be adequate for detection purposes in the next subsection.

Detection

There are three kinds of tests relevant in this section as described in the introduction to this chapter:

- Matching with the segmented training views.
- Matching with the unsegmented training views.
- Detection in random scene images.

The following paragraphs report on the different tests.

Matching with the Training Views Matching with the training views is important for checking the correctness of the mathematical operations involved and the overall implementation of the system. The error of these tests is measured with:

$$e_F = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_F(L_i, O_i)^2} \tag{7.2}$$

where L_i $(1 \leq i \leq n)$ is the ith learning frame and O_i is the ith estimated object frame. The distance measure d_F^2 used here is:

$$d_F = \sqrt{d_t^2 + (u\, d_r)^2} \tag{7.3}$$

where d_t is the translation difference and d_r is the rotation difference described in the appendix. Both are measured in millimeters. The value u is set to $u = 10\,mm$. A distance value of 20 either refers to a translation error of $20\,mm$ with no rotation error, a rotation error of 180° with no translation error, or a mixture of translation and rotation error.

In a first test the object models are matched to the training views with minimal voting cell sizes. This case corresponds to the exact situation in which the matching result and learning frame between two views are be identical. It turns out that with the implementation used here this is true since the resulting error is with $e_F = 0.0003...$ close enough to zero.

Figure 7.12: Error histogram of the known frames case (segmented images). There are five objects in the database and 72 rotated views per object. The histogram shows the distribution of the summands $d_F(L_i, O_i)^2$ of equation 7.3. The bucket size is equal to 10 units. This means that an error in the 150 bucket corresponds e.g. to a pure translation error of $\sqrt{150}\,mm \approx 12.25\,mm$. The tail of the distribution is cut at bin 250 and all remaining entries are included in the 250 bin.

In the second test voting cell sizes of $50\,mm$ and $180°$ are chosen. The error histogram is shown in figure 7.12, and in figure 7.13 some representative annotated result images are shown. The outcome is that detection in the centimeter range is possible, except for some complete mismatches that are collected in the bin 250. Most measurements are exact matches located in the first histogram bin. The main distribution expands up to bin 150. This is due to noise in the voting cells stemming from feature matches with some model features that do not belong to the observable object features. The entries larger than 250 refer to complete mismatches that can occur if too few features are available. An example of this case is the book edge shown in figure 7.13.

The average FFC matching time in these tests was about $0.07\,s$ on a 2.7 GHz test computer (single core). The results show that the computations are mathematically correct. One problem is that large voting cell sizes have to used. This is due to the instability of the local features frame's rotation components. A second problem is the fact that some views contain too less features. The error for the correct matches

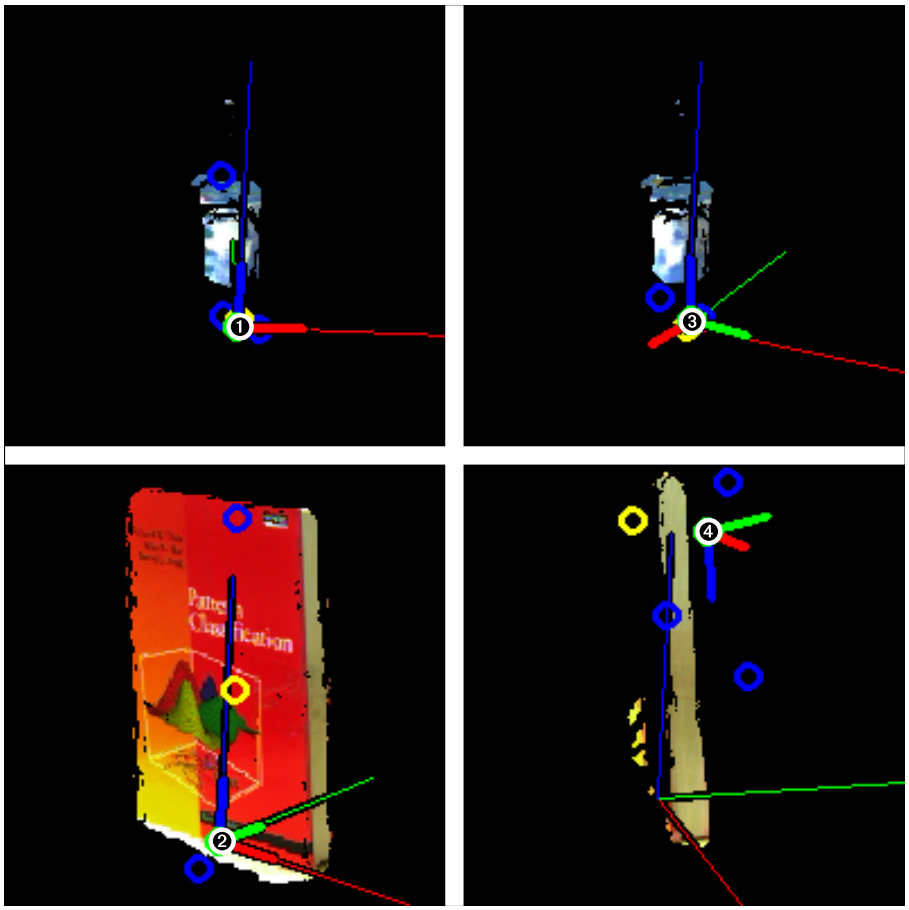

Figure 7.13: Detection results with known frames and segmented images. The bold frames with their x-, y-, and z-axes (red, green, and blue lines) are the estimated values O_i. The long thinner frames are the ground truth frames L_i. Left: examples for very precise matches (❶ and ❷). Right: mismatches due to too few features in an object view (❸ and ❹). The thin line long axes denote the ground truth frame, the short thicker axes the frame as is computed by the algorithm. The blue and yellow circles denote the next best matches (yellow: second best, blue: third to fifth best).

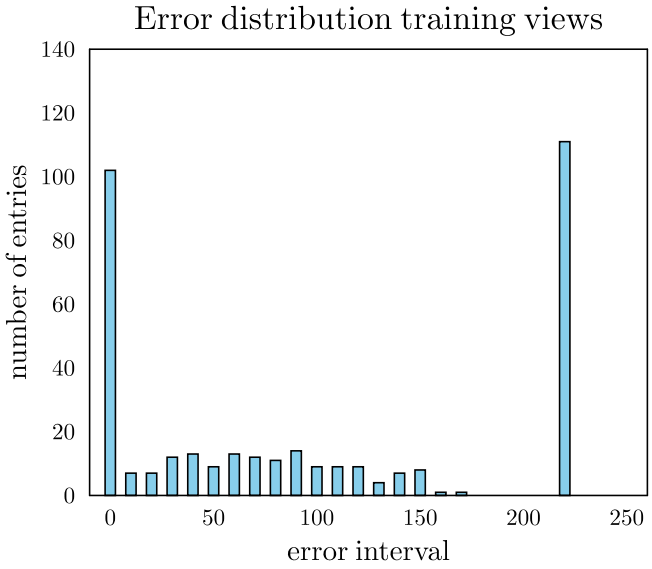

Figure 7.14: Error histogram known frames and unsegmented images. The tail of the distribution is cut at bin 220 and the remaining entries are added to this bin.

is roughly in the 1.5 cm range (translation).

Matching with Background A second series of tests test was run without background removal. The focus of these tests is the question whether other objects in the scene disturb correct detection. Dense regions in the voting space can occur elsewhere than at the object's real position. In the test performed the hit rate h and the error of the frames e_F were measured. The hit rate h counts the number of times the detection result is located within the cylinder used as segmentation shape. The error e_F is computed as described above.

The error histogram is shown in figure 7.14. The number of mismatches shown in bin 220 increases due to background disturbance. Note that bin 220 includes all results of the tail of the distribution. Still, a large amount of the results corresponds to exact matches. The hit rate in this test was $h = 0.73$, i.e. 262 views of 360 (5 objects, 72 views per object) were matched correctly. Here the full detection time was 0.5 s on average. The occurring errors are due to views having too few features compared to some regions in the background. This decreases the stability of the maximum voting cell. Especially, the text parts in the background interfere with the correct voting cell. Such dense regions may "survive" the basic loop in which

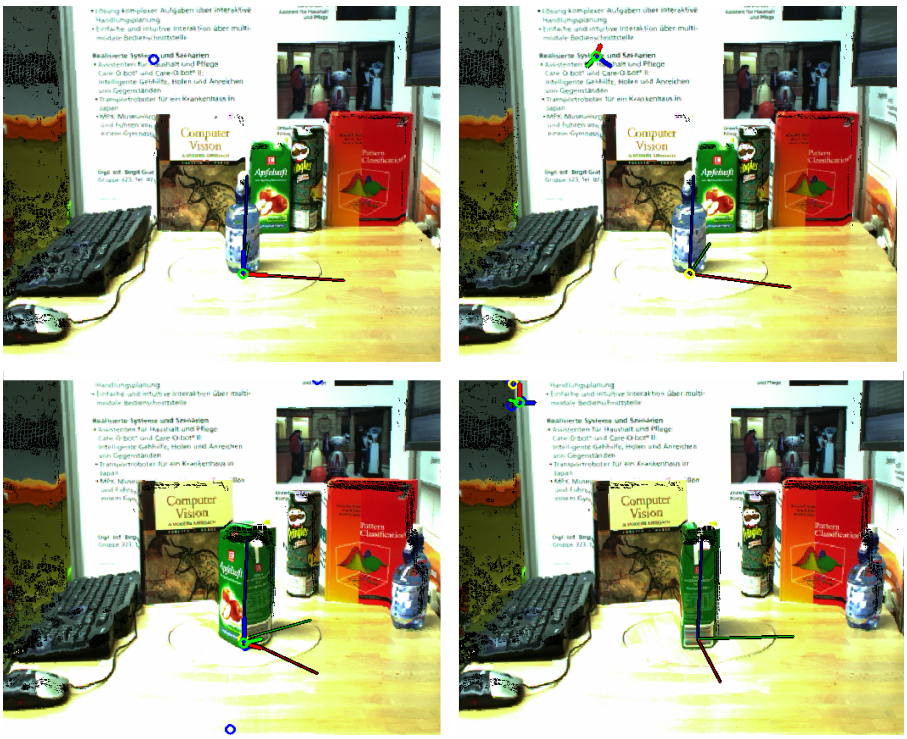

Figure 7.15: Detection results with known frames and unsegmented images. Left: examples of very precise matches. Right: inaccurate cases in which the background text on the poster interferes with the true correct voting cell. The frame annotation is equivalent to figure 7.13.

the hypotheses are generated. Better results would be possible using smaller cell sizes. However, this is only possible with a more stable computation of the rotation components of the local frame.

The resulting pose error remains in the range as before regarding the the correct matches. However, there is also a large number of mismatches due to the fact that large voting cell sizes have to be chosen. Densely distributed background features in the scene image also lead to dense feature distribution in voting cells that do not contain the target conversion frame. With a more stable computation of the rotation components of the local frames smaller cell sizes could be chosen to avoid this problem.

Matching in Random Scene Views In the final test series it was investigated how the system copes with random scene settings. A database consisting of 16 images was used for this purpose. The first five test images were drawn from the training

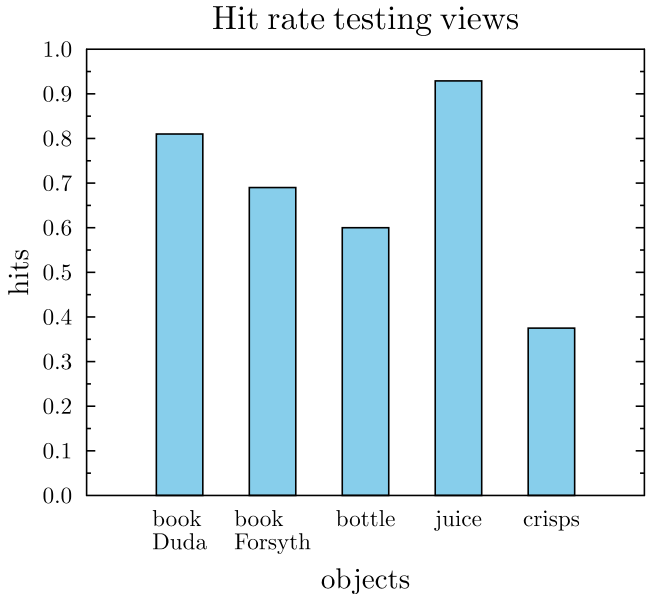

Figure 7.16: Single object rates for random scene images.

data; the rest are random scene set-ups. One scene image was created artificially as an exact copy of its predecessor and then darkened and blurred. The random scene images contain different object configurations.

Since here the correct object frames are unknown, the testing was based on annotated 2D rectangles. Approximate left, right, upper and lower borders in which the objects reside were manually defined. Not all objects can be seen in all images. The maximum number of hits possible in the perfect detection case is 67 in this database.

With the object models built from known conversion frames between the training views, 46 results are correct. In the successful cases the detected poses within the annotated rectangles seem to be correct. In figure 7.16 the hit counts of each object are shown. Figure 7.17 illustrates positive results. These include views with large object coverage. The average detection times for these images is $0.5\,s$ on the test computer (Pentium 5, 2.7 GHz, one core) with non-optimized software.

The results show that object poses not contained in the training set can be approximated using the registered model FFCs.

Figure 7.17: Illustration of selected results. Top three rows: results of the book "Duda", bottom row: results for the apple juice box. Note that in the bottom/left image the fraction of object related pixels is only about 2% (i.e. 98% of the image is background).

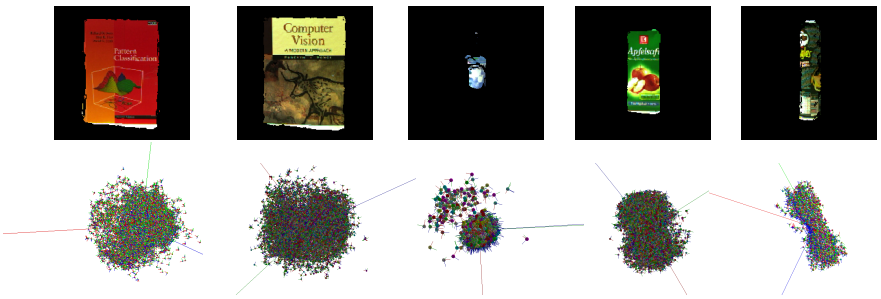

Figure 7.18: Approximated models constructed directly from the training views, not knowing the relative transformations between each view. The resulting models are rough approximations of the real shape.

7.4.2 Learning and Detection with Unknown Learning Frames

The final tests do not assume known frames. This is e.g. the case in the acquisition scenario in which the human shows the object. A difference to the real scenario is that here no fingers or other parts of the human body are contained in the training views. The approximated models were constructed using the hierarchical registration method. The time for training was only 10% higher than in the known frames case. This shows that the registration method is fast. The main problems are caused by views with a feature count too low. The registration method suggested can cope with this case. It produces compact but only approximate models.

The approximated models are shown in figure 7.17. The detection results are comparable to the known frames case with 46 hits on the test set. In the left/bottom image of figure 7.17 the fraction of pixels that belong to the object is only about 2% of the full image size. This shows that the methods are very robust to large amounts of background features in the image. The registration leads to an object reference frame central in the object. The correctness of this frame could only be validated visually during the tests reported here.

The outcome is that in the case using approximately registered object models the error is equivalent to the known frames case.

7.4.3 Live Tests

The software was ported on the robot platform shown in figure 7.19 (on the right). The software was shown live to project reviewers by demonstrating the learning and detection part on a randomly selected object out of an object set used in the project.

Figure 7.19: Live tests on the robot. Left: a segmented training image. Center: the object is detected even in the case of another object located on top of it. The green circle annotates the position of the detected frame. Right: the robot platform used in the live tests.

The learning and detection part were fully integrated and demonstrated to the project reviewer audience. As acquisition scenario, the human object demonstration was selected with a virtual wall and limitations in image width and height as segmentation shape. An example of a segmented training view is shown in figure 7.19 (left image). Several objects were taught on-line. The software was prepared to capture 15 object views per object. In most cases it took about 15 s to construct an object model out of these views. Therefore, it could be demonstrated live.

After training, the detection part was run iteratively and a green circle (see figure 7.19, right image) depicted the 3D coordinates of the detected frame's base point $\boldsymbol{o}_D$. It could be demonstrated that the detection method is fast enough to re-detect the object in small cycles (estimated at about 300 ms) including all necessary computations. The detection time was higher than during other tests reported in this chapter due to the use of a larger amount of features. Tests showed that large coverage can be coped with (as shown in figure 7.19, right image). An evident disadvantage of the system is that only objects that provide features can be trained and detected.

7.5 Summary

The experiments, tests and observations in this chapter can be summarized as follows:

- High precision range and color image fusion is possible based on standard calibration techniques only requiring a minimal manual effort. This is especially true for the calibration that uses the internal parameters of a calibrated range imaging sensor.
- Local coordinate frames at 3D feature point locations can be constructed that are stable over small object rotations. A set of one-to-one correspondences of local features frames can be used to recover an approximate pose.

- Since only one-to-one correspondences are needed in order to estimate a single vote on the object pose, the search complexity can be lowered from $O(n^6)$ to $O(n^2)$ compared to algorithms that need point triples for pose recovery.
- In the case of many possibly indistinguishable features, the approach of evaluation densities in the voting frame space outperforms a random (RANSAC) search.
- Due to the cheap computation of single votes on the object pose, descriptor matching can be simplified. An interesting solution is a simple threshold-based approach since only the features on the object of interest have to be processed and no relative measures among elements of an object set have to be taken into account.
- The tests on system level show that approximate object poses can be found in cluttered environments. However, since the rotation components of the single local frame are not very stable between similar views the voting cell sizes have need to be large. This induces errors such that mismatches can occur.
- Detection is possible in difficult setups where only a small fraction (2%) of the image belongs to the actual object or where parts of an object are covered.

8 Summary and Conclusions

In this chapter the content of this thesis is summarized and pointers to open issues are given.

8.1 Achievements

A new object detector is presented, which outputs an approximate object pose and which can learn object models. The models consist of shape and texture information. Its main application area is the service robot domain. However, applications in other areas are also possible.

First, the fields of service robots and object recognition are investigated. An assumption made in this thesis is that single object detection is an important capability needed. Some selected requirements led the work described here, among them:

- Full 3D pose output (3D position and orientation) is needed.
- Detection must be efficient (computationally cheap) and possible in "difficult" settings including object coverage, variations in brightness, different geometric settings and other effects. Also, the object it must be detectable if only a small fraction of observable space belongs to it.
- The system needs a learning ability that provides real robots with the capacity to learn object models in real-world learning scenarios since the objects are not known *a priori*.

The author believes that with the approach presented in this document, a step forward is achieved in these directions.

A main idea of this thesis is to combine range and color images and to explore the improvements possible with the new data. Two potential opportunities have been identified and further investigated:

1. Making real-world object learning feasible and implementable by using a simple range segmentation method to segment object views.
2. Reducing the search complexity by using a new object representation for object

model construction and object detection.

This thesis shows that both opportunities can actually be realized. The complete approach developed can be roughly divided into three parts:

- Sensor fusion and calibration
- Sample data acquisition
- Model construction and detection

Sensor fusion is done in analogy to stereo-vision by using conventional camera transformations. It is possible to reconstruct a shared image (a color image in range image format) such that for each pixel location, color and coordinate information are available. The reconstruction is an "inverse" process that assembles a color image seen from the range camera by fetching color values from the color camera based on 3D projections. In addition to the shared image computation algorithm the thesis also describes calibration methods that are self-contained and based on known calibration components. One method determines parameters for the complete sensor setting. A second one makes use of the default range camera parameters that are set by the manufacturer. The first method is used if the camera is not calibrated already. The second method is simpler and leads to more precise results. It is shown in the evaluation part of this document that image fusion is possible with a back-projection error in sub-pixel range related to the larger 640×480 color image size.

Sample data acquisition becomes possible for real robots in real-world environments by using the shared image coordinate part. Segmented images are obtained by pixel-wise masking, or by masking on feature point level. The parameters required to correctly position and layout masking shapes can be obtained by some acquisition strategies a service robot could follow. The robot could move around the object, it could move the object itself, or a human tutor could show the object. For all three cases advice is given on how the masking shape can be positioned during learning using components of the robot's control system. That this segmentation leads to data that can be used for object learning is shown in the evaluation part of this document. This fact is further supported by live trials not reported here. Problems occur if the object is transparent, has a metallic or glittering surface, or if object parts are too small in relation to the camera resolution. Of course, the approach is limited by the capabilities of the range imaging sensor used.

The capability of extracting relevant 3D and color data of the object in complex scenes is seen as a first achievement of this work. It is independent of the object representation and matching algorithm used in subsequent processing stages. The figure-background separation based on the 3D masking shapes is capable of delivering isolated object views required for learning.

Object model construction and detection are based on a new data representation called feature frame cloud (FFC). Two attributes of the FFC support efficient matching, the use of discrete keys that allow for fast accessing and the availability of local coordinate frames. The local coordinate frames are sparsely distributed since they are built only at feature point locations. They incorporate partial shape and texture information. The local coordinate frame is constructed by using the feature point's image gradient and the local surface normal. The FFC is a set of key/frame list pairs. Two FFCs can be matched efficiently by considering all possible frame correspondences, where one correspondence is from the base cloud and one from the argument cloud, that match with respect to the key. Each pair provides a full conversion frame hypothesis between object and scene.

This basic property of FFCs reduces the search complexity for a variety of matching algorithms. In the case where point triples are needed for full conversion frame reconstruction, (e.g. point cloud-based approaches or 3D feature points based on stereo-vision) the complexity of finding two matching triples can reach $O(n^6)$ (here n is the number of features in the object model *and* in the scene). A reduction to $O(n^4)$ is possible with approaches that would use e.g. two 3D positions and one surface normal. The use of FFCs reduces this further to $O(n^2)$. The features that are used to construct a FFC are not limited to the type used in this thesis. Full frames could be obtained based on other (e.g. purely shape-based) measures. The fusion of the 2D features and local 3D information is only one approach. However, this approach is seen as an elegant way to combine the 3D and 2D "worlds".

The reduction in search complexity related to full conversion frame estimation is seen as the second achievement of this work. Matching algorithms can be made faster and therefore higher feature counts are made feasible to process. This can lead to more robust recognition in large and complex scenes where only a tiny subset of the feature points belongs to the object of interest. A weakness of the current approach is that currently the poses obtained are imprecise. Especially the rotational components of the local frames need improvement. This is caused mainly by unstable normal computations, not strongly defined 2D gradient directions, and depth measuring errors (e.g. light reflections over multiple paths).

The matching algorithm suggested in this thesis collects all hypotheses resulting from all pairs and evaluates a density in the frame space by using a distance function suited for frames. The advantage of this approach is that a global optimum can be found. This makes it also particularly interesting for multi object detection. Comparisons to the more popularly used RANSAC technique (converted to FFC matching) shows that the density based approach outperforms RANSAC at high feature point counts of indistinguishable features. This in turn puts fewer requirements on feature descriptor matching and can lead to more robust recognition in complex scenes. The results of the density-based algorithm could be used for more precise pose estimation.

An ICP FFC variant could use the approximate pose as a basis for a more accurate fitting.

FFC matching is used to construct an object model by a hierarchical strategy if the relative frames between the learning views are not known. If they are known, then the views are just transformed to assemble the final object model FFC. The hierarchical strategy leads to approximate models that can be used for fast approximate detections. The registration technique can cope with situations in which two successive views have almost no matching features. An object can be detected by applying the object model FFC to the scene FFC. The methods are shown to work by some evaluations and live tests on a real robot platform. However, one major drawback is that the objects need to be structured such that they contain a sufficient number of the feature points chosen. Simple objects such as a uniform colored ball or box are impossible to model with this technique. Another disadvantage of the voting strategy is the voting cell size parameter which has to be selected manually. This could be made object dependent, scene dependent, or both.

8.2 Further Work

Some immediate improvements would enhance the approach presented here, e.g:

- Improvements related to the range measurement.
- Faster and more precise range and color sensor fusion.
- Pattern-free automatic calibration of the imaging sensors.
- Automatic on-line parameterization of the learning masks.
- More stable computation of the rotational components of the local frames.
- Object dependent feature parameters.
- Use of other alternative feature points (either color-based, or shape-based, or combinations).
- Automatic on-line adjustment of the voting cell size parameters.
- Improvement for registration in the case of unknown learning frames.

Furthermore, the following directions could be interesting building on this approach:

- Statistical knowledge about feature point distributions on the descriptor label level could be used to improve accuracy or to speed up matching.

- The approach presented here can also be generalized to a multiple object detector by using global descriptor clusters. during the introduction of new objects new clusters are only added if necessary (i.e. many clusters are shared by different objects). The matching algorithm would then iterate over all descriptor models and each voted hypothesis would be stored in conjunction with one or more possible object labels.
- Fast controlled detection is possible by using the masking shape that is used for segmenting the training data as a pre-selector during recognition. Tracking would thus be easily possible by continuously updating the masking shape position based on the previous detection result.
- Attention selection control could direct the masking shape to constrain the search space and to provide support for context-based recognition.
- FFCs could be used for category detection by the use of key or frame statistics or to model animated objects containing mechanical degrees-of-freedom. Then separate static FFCs (the components) would have to be "connected" by the kinematic equations.

These opportunities support the idea that research on the capabilities of feature frame clouds could be an interesting future direction in the field of robot vision.

Appendix

In this appendix the notation and and mathematical basics used in this document are described.

3 General Type Notations

A *scalar* $s \in \mathbb{R}$ is written as a small letter, as are single numbers of other numerical sets, e.g. $u \in \mathbb{N}$. Tuples of scalars are used to represent *vectors* of different dimensionality as column vectors, i.e. $\boldsymbol{a} = (a_1, a_2, .., a_d)^T$, where d is the dimensionality and ^{T} is used to represent the transpose. Typical vectors used here are 2D and 3D vectors, e.g. $\boldsymbol{a} = (a_x, a_y, a_z)^T \in \mathbb{R}^3$ describes a 3D column vector used for a position or direction with magnitude in space and $\boldsymbol{b} = (u, v) \in \mathbb{N}^2$ is a discrete point in an image plane. Generally, subscripts are used to further specify variables or constants. Numbers in the subscripts are used for enumerations.

Tuples of more complex types than scalars are described by large letters, e.g. a frame $F = (\boldsymbol{o}_F, \boldsymbol{i}_F, \boldsymbol{j}_F, \boldsymbol{k}_F) \in \mathbb{F}$ describes a coordinate frame with an origin point $\boldsymbol{o}_F \in \mathbb{R}^3$ and the directions of the coordinate axes $\boldsymbol{i}_F \in \mathbb{R}^3$, $\boldsymbol{j}_F \in \mathbb{R}^3$, and $\boldsymbol{k}_F \in \mathbb{R}^3$. Large bold letters denote matrices of scalars such as a rotation matrix $\boldsymbol{R} \in \mathbb{R}^3 \times \mathbb{R}^3$. For matrices of more complex types than scalars, large letters in calligraphic are used, e.g. $\mathcal{I}$ is used for a 2D image (arranged in rows and columns) that has tuples of scalars as elements. Sets are denoted with large roman letters such as $\mathrm{S} = \{a, b, c\}$ independent of the type of elements they contain. Lists are represented as enumerates sets, e.g. $\mathrm{L} = \{e_1, e_2, ...e_n\}, n \in \mathbb{N}_1$.

4 Point and Frame Operations

In this document, a frame F is defined by $F = (\boldsymbol{o}_F, \boldsymbol{i}_F, \boldsymbol{j}_F, \boldsymbol{k}_F) \in \mathbb{F}$, where $\boldsymbol{o}$ is the coordinate vector of an origin (or base) point $\boldsymbol{o}_F$ and $\boldsymbol{i}_F$, $\boldsymbol{j}_F$, and $\boldsymbol{k}_F$ the directions of the coordinate axes (unit lengths). The set $\mathbb{F} \subset \mathbb{R}^{3\times4}$ denotes the frame space. The *direction cosine matrix* (DCM) $\boldsymbol{R}_F$ can be easily constructed as from these data as well as the rotation axis $\boldsymbol{u}_F$ and angle α_F or the related quaternion.

A special frame is the zero or world frame W in which other frames can be expressed, $W = ((0,0,0)^T, (1,0,0)^T, (0,1,0)^T, (0,0,1)^T)$. A point in a frame F is denoted as

$$ {}^F\boldsymbol{p} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = a_x\, \boldsymbol{i}_F + a_y\, \boldsymbol{j}_F + a_z\, \boldsymbol{k}_F \tag{1} $$

and for points in the world frame W the leading superscripts are omitted, ${}^W\boldsymbol{p} = \boldsymbol{p}$.

A point ${}^F\boldsymbol{p}$ measured in some frame F can be converted to world coordinates by the following transformation function:

$$ \boldsymbol{p} = rot_F({}^F\boldsymbol{p}) + \boldsymbol{o}_F = \boldsymbol{R}_F\, \boldsymbol{p} + \boldsymbol{o}_F \tag{2} $$

where $rot_F : \mathbb{R}^3 \to \mathbb{R}^3$ is a rotation function. This can be based on multiplication with the rotation matrix, by the use of quaternions, or by the Rodriguez' formula ([5]). The inverse operation can be accomplished with

$$ {}^F\boldsymbol{p} = rot_F^{-1}(\boldsymbol{o}_F - \boldsymbol{p}) = \boldsymbol{R}_F^{-1}\, \boldsymbol{p} + \boldsymbol{o}_F \tag{3} $$

where rot_F^{-1} denotes a rotation around the same axis about the negative rotation angle. This can also be done by a multiplication with the inverse rotation matrix $\boldsymbol{R}^{-1} = \boldsymbol{R}^T$.

In this document frames are not only used as transformations but also as the objects of measurement. This notation

$$ C = {}^B A \tag{4} $$

is used to denote a relative frame A measured in some other frame B. This is closely related to the notion of frames as homogeneous coordinate transform matrices which is also used in this document. A 4×4 homogeneous transformation matrix ${}^B_A\boldsymbol{T}$ also describes how A lies in B. It can be constructed from the relative rotation matrix and offset vector (see e.g. [14]). In order to work with networks of transforms the transformation matrices can be multiplied and this gives rise to so-called *transform equations* that serve as an algebra tool to solve for an unknown frame. An important equality is

$$ {}^B_A\boldsymbol{T} = {}^A_B\boldsymbol{T}^{-1} \tag{5} $$

which is the inverse transform.

In this work a frame difference function is used. A simple and intuitive frame difference function $d_F : \mathbb{F} \times \mathbb{F} \to \mathbb{R}_+$ is based on a weighted sum between the translation of the two frames d_t and a term denoting the rotational distance d_r (see e.g. [25]):

$$ d_F(F, G) = d_t + u\, d_r \tag{6} $$

where $u \in \mathbb{R}$ is a factor to be chosen manually. Usually, the Euclidean distance of the base points of a frame is used for the translation term:

$$d_t = |\boldsymbol{o}_F - \boldsymbol{o}_G|. \tag{7}$$

The first rotation distance used in this work, d_r, is based on the distances between two rotated points of one and the same original reference point using $\boldsymbol{u}_F$ and $\boldsymbol{u}_G$, the rotation axes of F and G. To guarantee equal influences of the two rotations a reference point is chosen that stays orthogonal on both rotation axes $\boldsymbol{p}_{ref} = \boldsymbol{u}_F \times \boldsymbol{u}_G$. If the two rotation axes are equivalent then a random point orthogonal to them can be chosen. The Euclidean distance of two rotated points is then used as rotation distance d_r:

$$d_r = |rot_F(\boldsymbol{p}_{ref}) - rot_G(\boldsymbol{p}_{ref})|. \tag{8}$$

Note that this function is only used in this document. Another distance metric for frames is given in [25]. This was used later in this work. It replaces the rotation distance in the frame distance formula above with:

$$d_r = \cos^{-1}\left(\frac{\text{trace}(\boldsymbol{R}_F^T \boldsymbol{R}_G) - 1}{2}\right) \tag{9}$$

where $\boldsymbol{R}_F$ and $\boldsymbol{R}_F$ are the rotation matrices of the two frames. This distance measure is known to fulfill the requirements for a distance metric.

In the following, operations on sets of frames that are relevant in this document are described.

Clustering is performed by a QT clustering algorithm that used the difference function and a threshold to count the number of frames in the vicinity of a single frame. Those are then successively removed. The result is a sparser list of frame containing fewer double or multiple appearances that are only slightly different with respect to the frame distance.

Centralizing is an operation that is similar to using PCA to align point clouds around the neutral frame. The centralizing operation used in the measurements in this document sums up all $\boldsymbol{o}_F$, $\boldsymbol{i}_F$, $\boldsymbol{j}_F$ and $\boldsymbol{k}_F$ components of the frames and constructs a new frame by finding orthonormal vectors based on the sums. The frame computed can then be used to convert all frames in the list such that they are aligned to the zero offset and neutral rotation.

5 Image Notation

A simple image with one channel of data, e.g. gray-level image is represented as a matrix

$$\boldsymbol{I} = \begin{pmatrix} i_{1,1} & i_{1,2} & \dots & i_{1,h} \\ i_{2,1} & i_{2,2} & \dots & i_{2,h} \\ \vdots & \vdots & \ddots & \vdots \\ i_{w,1} & i_{w,2} & \dots & i_{w,h} \end{pmatrix} \tag{10}$$

of type (w, h), where $w \in \mathbb{N}$ is the image width and $h \in \mathbb{N}$ is the image height. An image matrix can be accessed with

$$i_{u,v} \in C \subset \mathbb{R} \tag{11}$$

where $v \in [1, .., h]$ and $u \in [1, .., w]$ are the discrete image coordinates. Note that in typical implementations of arrays in programming languages the top left element starts with $i_{0,0}$ and the bottom right element is $i_{h-1,w-1}$. The value type of the components $i_{v,u} \in C$ is a subset of reals $C \subset \mathbb{R}$. If v and u are not important for descriptions then the image value is just referred to as i, assuming that $i = i_{v,u}$. The value i can have different actual meanings e.g. intensities, distances or temperature depending on the type of imaging sensor used.

Bibliography

[1] Sung Joon Ahn. *Least Squares Orthogonal Distance Fitting of Curves and Surfaces in Space (Lecture Notes in Computer Science)*. Springer-Verlag New York, Inc., 2004.

[2] Ronald C. Arkin. *Behavior-Based Robotics*. MIT Press, 1998.

[3] A.M. Arsenio. Embodied vision - perceiving objects from actions. In *Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003. The 12th IEEE International Workshop on*, pages 365–371, 2003.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. J.Van Gool. Speeded-up robust features (SURF). *CVIU*, 110(3):346–359, 2006.

[5] S. Belongie. Rodrigues' rotation formula. `http://mathworld.wolfram.com/RodriguesRotationFormula.html`, 2010.

[6] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24):509–521, 2002.

[7] P. J. Besl and H. D. Mckay. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, 1992.

[8] A. Blake. *Active Vision*. MIT Press, 1992.

[9] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. *Proceedings of the 11th European Conference on Computer Vision (ECCV)*, 2010.

[10] H. D. Chati, F. Muhlbauer, T. Braun, C. Bobda, and K. Berns. Hardware/software co-design of a key point detector on FPGA. *Field-Programmable Custom Computing Machines*, 23(25):355–356, 2007.

[11] F. Chiabrando, D. Piatty, and F. Rinaudo. SR-4000 TOF camera: Further experimental tests and first applications to metric surveys. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII(5), 2010.

[12] N. Christianini and J. Shawe-Taylor. *An Introduction to Support Vector Ma-*

chines and Kernel-Based Learning Methods. Cambridge University Press, 2000.

[13] Ondrej Chum and Jiri Matas. Matching with PROSAC - progressive sample consensus. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 220–226. IEEE Computer Society, 2005.

[14] J. J. Craig. *Introduction to Robotics: Mechanics and Control (third edition)*. Addison-Wesley, 2004.

[15] D. Decoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46:161–190, 2002.

[16] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[17] M. A. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. In *Proceedings of the Image Understanding Workshop (College Park, Md., Apr i980)*, pages 71–88, 1980.

[18] P. Fitzpatrick. *From First Contact to Close Encounters: A Developmentally Deep Perceptual System for a Humanoid Robot*. PhD thesis, MIT, 2003.

[19] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[20] Yoav Freund and Robert E. Schapire. A short introduction to boosting. In *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406. Morgan Kaufmann, 1999.

[21] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. The amsterdam library of object images. *Int. J. Comput. Vision*, 61:103–112, 2005.

[22] Iryna Gordon and David G. Lowe. What and where: 3D object recognition with accurate pose. In *Toward Category-Level Object Recognition*, pages 67–82, 2006.

[23] B. Graf, M. Hans, J. Kubacki, and R. D. Schraft. Robotic home assistant care-o-bot. In *Bioengineering - Integrative Methodologies, New Technologies - EMBS-BMES 2002*, pages 2343–2344, 2002.

[24] Birgit Graf, Christopher Parlitz, and Martin Hägele. Robotic home assistant Care-O-bot 3. In *HCI (2)*, pages 312–320, 2009.

[25] R. Di Gregorio. A novel point of view to define the distance between two rigid-body poses. In *Advances in Robot Kinematics: Analysis and Design*, pages 361–369, 2008.

[26] T. Grundmann, Z. Xue, J. Kuehnle, R. Eidenberger, S. Ruehl, A. Verl, R. Zoell-

ner, J. Zoellner, and R. Dillmann. Integration of 6d object localization and obstacle detection for collision free robotic manipulation. *Proceedings of the IEEE International Symposium on System Integration*, 2008.

[27] U. Hahne and M. Alexa. Combining time-of-flight depth and stereo images without accurate extrinsic calibration. *Int. J. Intell. Syst. Technol. Appl.*, 5(3/4):325–333, 2008.

[28] R. M. Haralik and L. G. Shapiro. *Computer and Robot Vision. Vol. I.* Addison-Wesley, 1992.

[29] C. Harris and M. Stephens. A combined corner and edge detector. *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[30] Rob Hess. Open-source SIFT library. `http://blogs.oregonstate.edu/hess/`, 2012.

[31] Laurie J. Heyer, Semyon Kruglyak, and Shibu Yooseph. Exploring expression data: Identification and analysis of coexpressed genes. *Genome Res.*, 9(11):1106–1115, 1999.

[32] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[33] P. Hough. Method and means for recognizing complex patterns. U.S. Patent 3.069.654, December 1962.

[34] A. Jaklič, A. Leonardis, and F. Solina. *Segmentation and Recovery of Superquadrics*. Kluwer, 2000.

[35] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[36] Adrian Peter Paul Jongenelen. *Development of a Compact, Configurable, Real-time Range Imaging System*. PhD thesis, Victoria University of Wellington, 2011.

[37] D. Kragic and H. Christensen. Survey on visual servoing for manipulation. *Computational Vision and Active Perception Laboratory*, 2002.

[38] J. Kubacki. How robot companions see the world: Methods to equip robot companions with sophisticated vision abilities. *Inspect*, 2008.

[39] J. Kubacki and W. Baum. Toward open-ended 3d rotation and shift invariant object detection for robot companions. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[40] J. Kubacki, B. Giesler, and C. Parlitz. Active autonomous object modelling for recognition and manipulation: Towards a unified object model and learning cycle. In *Autonome Mobile Systeme 2005*, pages 227–233. Springer, 2005.

[41] J. Kubacki and K. Pfeifer. Using range imaging sensors with color imaging sensors in cognitive robot companions: a new and simple calibration technique based on particle swarm optimization. In T. Ingensand and Thimo Kahlmann, editors, *Proceedings of the first Range Imaging Research Day (RIM)*, volume 1, pages 43–59, 2005.

[42] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. *Computer Vision and Pattern Recognition, 2005. CVPR 2005.*, 2:775–781, 2005.

[43] Vincent Lepetit, Pascal Lagger, and Pascal Fua. Randomized trees for real-time keypoint recognition. In *In CVPR*, pages 775–781, 2005.

[44] Point Cloud Library. `http://pointclouds.org/`, 2012.

[45] D. Lowe. Demo software: Sift keypoint detector. `http://www.cs.ubc.ca/~lowe/keypoints/`, 2005.

[46] D. G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(5):441–450, 1991.

[47] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[48] H. Bischof M. Grabner, H. Grabner. Fast approximated SIFT. In *Proc. ACCV 2006*, volume 3851 of *LNCS*, pages 918–927. Springer, 2006.

[49] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2005.

[50] H. P. Moravec. Towards automatic visual obstacle avoidance. *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, page 584, 1977.

[51] Hans Moravec. *Robot: Mere Machine to Transcendent Mind*. Oxford University Press, 1999.

[52] U. Nehmzow. *Mobile Robotics: A Practical Introduction*. Springer, 2000.

[53] S. Nene, S. Nayar, and H. Murase. Columbia object image library: Coil. Technical Report CUCS-006-96, Department of Computer Science, Columbia University, 1996.

[54] OpenCV. `http://opencv.willowgarage.com`, 2012.

[55] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *Accepted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[56] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. *Conference on Computer Vision and Pattern Recognition*, 2007.

[57] Rolf Pfeifer and Christian Scheier. *Understanding Intelligence*. MIT Press, 1999.

[58] M. Pontil and A. Verri. Support vector machines for 3d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):637–646, 1998.

[59] Erwin Prassler. Deutsche Serviceroboter Initiative. `http://www.service-robotik-initiative.de/`, 2012.

[60] PrimeSense. `www.primesense.com`, 2012.

[61] Ulrich Reiser, Christian Connette, Jan Fischer, Jens Kubacki, Alexander Bubeck, Florian Weisshardt, Theo Jacobs, Christopher Parlitz, Martin Hägele, and Alexander Verl. Care-O-bot®3–creating a product vision for service robot applications by integrating design and technology. In *Proceedings of the 2009 IEEE/RSJ International Conference on Robots and Intelligent Systems (IROS)*, pages 1992–1998, 2009.

[62] Point Grey Research. `http://www.ptgrey.com/`, 2012.

[63] Robocup. `www.robocup.org`, 2012.

[64] D. Roobaert. *Pedagogical Support Vector Learning - A Pure Learning Approach to Object Recognition*. PhD thesis, KTH Stockholm, 2001.

[65] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings of the Third Intl. Conf. on 3D Digital Imaging and Modeling*, pages 145–152, 2001.

[66] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

[67] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Technical Report 99-87, Microsoft Research*, 1999.

[68] Rolf Dieter Schraft, Martin Hägele, and Kai Wegener. *Serviceroboter: Visionen*. Springer, 2004.

[69] F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Transactions on Pattern*

Analysis and Machine Intelligence, 12(2):131–147, 1990.

[70] A.R. Specht, A.D. Sappa, and M. Devy. Edge registration versus triangular mesh registration, a comparative study. *SP:IC*, 20:853–868, 2005.

[71] Todor Stoyanov, Athanasia Louloudi, Henrik Andreasson, and Achim J. Lilienthal. Comparative Evaluation of Range Sensor Accuracy in Indoor Environments. In *Proceedings of the European Conference on Mobile Robots (ECMR)*, Sep 7–10 2011.

[72] Alexandru Telea. An image inpainting technique based on the fast marching method. In *Journal of Graphics GPU, and Game Tools 9*, volume 1, pages 23–34, 2004.

[73] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[74] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002.

[75] Christian Wallraven. *A computational recognition system grounded in perceptual research*. PhD thesis, Max-Planck Institute for Biological Cybernetics, 2007.

[76] J. Weng and Y. Zhang. Developmental robots: A new paradigm. *Proc. Second International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, 2002.

[77] Juyang Weng and Wey-Shiuan Hwang. Online image classification using IHDR. *IJDAR*, 5(2-3):118–125, 2003.

[78] Haim J. Wolfson and Isidore Rigoutsos. Geometric hashing: An overview, 1997.

[79] C. Wu. SiftGPU: A gpu implementation of scale invariant feature transform (SIFT).
`http://www.cs.unc.edu/~ccwu/siftgpu/`, 2012.

[80] W. Zhang, B. Yu, G. Zelinsky, and D. Samaras. Object class recognition using multiple layer boosting with heterogeneous features. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 323–330. IEEE Computer Society, 2005.

[81] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[82] J. Zhu, L. Wang, R. Yang, and J. Davis. Fusion of time-of-flight depth and stereo for high accuracy depth maps. *Computer Vision and Pattern Recognition*, 23-28:1–8, 2008.

Glossary

accumulator
A storage data structure that contains many single hypotheses of a solution to a problem that were derived from single estimations called "votes".

acquisition scenario
A scenario that includes a robot, an object of which the properties have to be learned, possibly a human tutor, and possibly some prepared facilities in the environment needed for learning.

active vision
Methods that make use of actuators connected to the vision system in order to simplify the vision task.

behavior-based control
A control technique that is organized in different parallel layers that each have an own goal, sometimes also their own senses, and send commands to shared motor resources.

Care-O-bot® 3
A service robot research platform developed at Fraunhofer IPA.

cluster identifier
Used in this document for a discrete number associated with a certain cluster address.

control system
A system that produces actions from received perceptions such that a goal can be met.

coordinate image
An image containing coordinate (XYZ) information at each pixel's location.

degrees-of-freedom
Number of free and independent dimensions of motion in a kinematic system.

descriptor

This means a vector containing regional information around a feature point. The information should be stable against typical image transformations and disturbances

discrete description key
A term used in this document to describe a small list of discrete numbers used as key to a list of frames.

EM clustering
Expected Maximization (EM) clustering uses probability distributions to achieve a "soft" clustering where each sample belongs to all clusters with a specific probability.

embodied vision
Vision techniques that rely on and make use of the existence of a body.

environment
This is the "external" world in which the robot acts. The robot perceives partial information about the state of the environment through its sensors and it can alter the state of the environment using its actuators.

feature frame cloud
A feature frame cloud is a light 3D representation that can be used for frame estimation. It consists of discrete key vectors that are derived from distinctive attributes pointing to lists of coordinate frames.

feature point
A local pattern in an image that can be assigned a definite image position and that can be repeatedly detected if the image is subject to transformations (also called *point features*, *interest points*, or *key points*)

field-of-view
This term refers to a subset of space that can be inspected by the vision sensors without moving them.

first-time segmentation
Used in this document to describe a segmentation of an object's view when the object itself is unknown so far.

forward transformation
A transformation that maps the internal readings of the range imaging sensor onto the real x-, y-, and z-coordinates in the sensor frame.

frame distance
A distance function to measure the difference between two coordinate frames.

geometric hashing
A method that casts pose information of point clouds into discretely accessible map entries that can be used for fast matching.

Hough transformation
An algorithm that constructs a parametric accumulating space in which all possible models that can be associated with a single measured observation (e.g. 3D point) are successively added.

information content
The property of feature points that they contain information valuable for matching purposes.

integral image
An image that contains at each pixel location the sum of all pixels in the original image that lie within the rectangle that is spanned by the origin (zero) point and the location of interest

iterative closest point algorithm
An algorithm that aligns two point clouds by iteratively estimating a transformation and applying it.

object model
A description of a specific object by a set of parameters. Models may be *explicit* (e.g. geometric descriptions) or *implicit* (e.g. neural network).

point cloud
A set of 3D points used to represent an object's shape.

QT clustering
Quality Threshold (QT) clustering groups data samples by selecting prototype samples that have a high density within a distance-based vicinity.

quaternion
A representation for 3D orientation that is closely related to the rotation axis and rotation angle.

random trees
A learning algorithm based on a collection (forest) of decision trees.

range image
An image containing the time-of-flight values of a range imaging sensor.

range segmentation
A segmentation process that separates a colored object region in a color image based on the real 3D coordinates of the areas in the environment that are associated with the pixels in the image.

RANSAC
A matching strategy based on finding a few support measurements randomly that give rise to a globally consistent model.

registration
The process of associating each point of one point cloud with a correspondence in the second "argument" point cloud.

repeatability
This refers to the property of feature points that they are detected stably.

scalability
This term addresses the capability of object recognition systems to cope with large and diverse object sets.

scale space
A 3D space of an image consisting of the two discrete image coordinates plus a scale coordinate. The scale space is built by image pyramids of down-sampled image copies of the original with decreasing size and/or successive smoothing

segmentation mask
A region in 3D space that is used for first-time range segmentation in order to acquire segmented training images.

sense-plan-act
A scheme often used in robot control systems that senses and then plans action sequences based on the previous observations and finally forwards the action command to the actuators.

service robot
A robot that is able to provide services to the human possibly involving physical manipulation of objects.

shared image
A term used in this thesis that describes a six-layer image that represents color (RGB) and coordinate (XYZ) information in congruent 2D pixel coordinate systems.

SIFT
Scale-invariant feature transform based on the *difference-of-Gaussians* (DOG) scale space and local orientation histograms.

support vector machine
: A batch learning and classification technique that has been successfully adapted to difficult recognition problems. SVMs choose some subset of the training samples (the support vectors) that describe a separating function between two classes with a maximal margin.

SURF
: Speeded-up robust features are based on fast Hessian scale space and frequency responses for local descriptors.

undistorting
: The process of removing the "pillow effect" from images that results from light refraction at the lens's surface.

vision stages
: A general concept for detecting structure in images by proceeding through: conditioning, labeling, grouping, extracting, and matching.

vision system
: A module or a set of modules that is (are) part of the overall robot control system providing basic perceptual capabilities based on imaging sensors.

visual servoing
: The topic of robot control by combining closed-loop control methods with vision sensors as input sources.