Institut für Neuroinformatik
Ruhr-Universität Bochum

On the Recognition of Objects by Contour Parts and the Early Development of Biological Contour Processing

Dissertation zur Erlangung des Grades eines Doktors der Naturwissenschaften der Technischen Fakultät der Universität Bielefeld

vorgelegt von

Carsten Prodöhl

aus Wuppertal

Juli 2004

Overview

The following work is split into two parts. The first part describes a technical computer vision method for the recognition of objects by their contour characteristics even if these contours have undergone affine transformations, are only partially visible or are partially embedded in another larger contour. In case complete closed contours should be recognized the method can use normalization procedures that reduce the computational cost substantially. The method can as well be used to do object classification or reversely be used to define clusters of similar objects and if at least a contour part is specific enough it is even possible to identify individual representatives of an object class. The first part of this work leads to the conclusion that contour processing can be an important part in solving vision problems. Little is known about how contour processing is organized on an intermediate level in biological brains. The second part of this works aims to contribute to our understanding of how the first steps of contour processing could develop in biology, by showing how the Gestalt principles of collinearity and curvilinearity can be learned from object motion without being dependent on the statistical properties of the background.

Contents

CONTENTS

Acknowledgments

My own personal intellectual development would not have been possible in the past years and this thesis could not have developed the way it has without the help of a lot of people. For this reason I would like to express my gratitude to the following persons. To me this is a personal urge and pleasure as my development and my thoughts were influenced to unmeasurable extents. Especially in the intellectually very fertile environments I had the pleasure to be in.

Hans-Joachim Prodöhl and Editha Helga Prodöhl for giving me invaluable and never ending emotional support at all stages of my life.

Christoph von der Malsburg for giving me the possibility to work in such an intellectually challenging environment. I always appreciate the long hours he spent discussing with me, although we did not always agree on some subjects and I was not always able to express my thoughts in clarity at the first time. The resulting discussions are in my eye the foundation for the emergence of most of my successful new ideas.

Helge Ritter for accepting this thesis at the University of Bielefeld.

Rolf Würtz for the many ours he spent thinking about the results of my work and for asking the right — sometimes unpleasant — questions that have improved it considerably.

Andrea Figge for her almost inexhaustible patience and endurance in supporting me.

Uta Schwalm for her support on the day to day problems encountered at the institute and her continuing effort to make the institute one of the best organized ones in the world.

Prof. Daunicht for encouraging me to study the field of computer science and helping me execute my diploma thesis.

Prof. Hoffmann for the patience and time he spent giving me some new insights into biological neurophysiology.

Michael Neef for his tireless efforts to maintain and perfect the institute's computer system which is a great benefit although our interests have not always been the same. Furthermore for his rapid hot fixes and unconventional help with the ubiquitous computer problems.

Hartmut Neven for his inspiring thought on how algorithms in computer vision should be developed.

Jan Vorbrüggen and *Christian Eckes* for showing me the problems associated with segmentation tasks before I got stuck in them myself.

Norbert Krüger for his unconventional and passionate way of thinking about problems in computer vision.

Jochen Triesch for giving me an idea of how a combination of prospective visions and solid implementation enables the creation of astonishing solutions for computer vision problems.

Michael Pötsch for his clear and inspiring ideas about software development and life in general.

Michael Rinne for his patience in teaching me the foundations of good object oriented programming although he never wanted to do that purposely.

Stefan Zadel for his inspiring theoretical and philosophical views on the future of computer vision.

Thomas Maurer for his hint on learning and using the perl language.

Axel Steinhage for many joyful hours spent after work.

Bram Bolder for a lot of fun during and after work and his effort on the implementation of a family of c++ classes for different image sources as well as his invaluable linux support.

Wilfried Horn for his help in implementing some algorithms without which the completion of this thesis would have been delayed.

Maximilian Krüger for helping me to record some image sequences.

Witali Kusnezow for his view on c++ coding which is for the normal person somewhere between ingenious and detail enamoring.

Ingo Wundrich for his recreating views on the importance of signal processing techniques in an environment of computer scientists.

Günter Westphal for providing some image data sources and his excellent way of organizing programming projects.

Andreas Tewes and Jörg Lücke for their support in strategic software development questions.

Jörg Bornschein and *Dennis Rieks* for their help on implementing an object oriented graphical user interface.

Mike Pagel for a lot of joyful days learning c++ together.

Peer Schmidt for always having a grin on his face which is an invaluable thing to have at your working place.

Hartmut Loos for numerous discussions which future path to follow.

This work would have hardly been possible without the financial support given by the *German Minister of Science and Technology* (BMFT) and the *Deutsche Forschungs Gesellschaft* (DFG).

This work has been typeset by the author in LaTeX, so thanks are due to the creators of TeX and LaTeX, who offer this high quality text processing as public domain software. All simulations have been carried out on a Intel workstation using the c++, scilab and the perl language whose qualities and power are also gratefully acknowledged.

Finally, I would like to thank the readers who appreciate the unconventional nature of this work.

CONTENTS

Part I.

Object Recognition by Contour Parts

1. Model Concepts, Extraction, Recognition and Retrieval of Natural Shapes

It is possible to store the mind with a million facts and still be entirely uneducated.

Alec Bourne, ”A Doctor’s Creed”

1.1. The Concepts of Natural Shape Recognition Approaches

A lot of different animal species and humans show the remarkable ability to cope with the problem of segmenting objects from the background in the visual input they receive. Furthermore they have the ability to recognize known objects sometimes only by their shape. On the other hand shape description is a major problem in machine perception and is a basis for the recognition, coding and retrieval of objects. It is a great challenge in computer vision to generate algorithms that are able to reliably identify ’known’ objects in the visual input in a way like humans are able to do it. Before we take a closer look at the requirements, implementations and applications of shape recognition we have to think about the concepts that are the foundations for contour algorithms especially the important question of what defines an object as ’known’ in the above mentioned sense. In terms of computer vision the most general definition would be: Some data describing the shape of an object is already stored in the memory of a computer before new visual input containing a similar or identical object is evaluated. Regardless of the model used or the parameters constituting the model the question arises of how this knowledge has found its way into the computer. One possibility is of course that a human has developed the model for a specific contour (or set of contours) and has found the right set of parameters and appropriate values for them either by computing, guessing or statistically evaluating multiple algorithmic experiments using different parameters. While this approach may yield good results for specific tasks — just think of the Hough transformation — it is often hard or impossible for the computer algorithm to extend the knowledge base beyond what was already put in, if for example a new contour with unknown properties will be presented in the future. A similar problem occurs with algorithms that need a certain amount of representative input before their statistical evaluation schemes are able to compute, extract or identify a new contour given to them as new visual input. If the statistical properties of this new contour are different from the ones of the already learned contours stored so far, it is at least questionable that these algorithms can proceed and fulfill their task. Closely related to this kind of modeling is the

idea of extracting an object contour by locally connecting simple feature detectors to form a representation for a complete complex contour, as it implies the existence of an *a priori* abstract contour description that allows the algorithm to evaluate which features should be connected and which should not. The line of thought in this case is: If a model for natural contours could be found it should be possible to compute the local interaction functions that link or do not link the simple elements together and form the local contours. While it is beyond doubt that such mechanism play an important role in low level biological contour processing (see part II) it is questionable if such an approach can be sufficient. A philosophical question arises on the nature of biological learning: How is it possible for biological systems to learn from very few examples in a relatively short amount of time without the need for a lot of repetitions. Prof. von der Malsburg used to tell his students a thought experiment illustrating this feature:

> If your friend gets attacked by a wild animal, e.g. a tiger, while you both are traveling through a jungle, you might not be able to see much of the tiger before he vanishes again — probably with your friend — and it may have been the first occasion you have ever encountered a tiger, but the next time you see a tiger you will recognize the tiger immediately. You do not need to relive that experience a large number of times before you are able to extract the relevant features from the visual input so that the next time you see a tiger you will be able to recognize the tiger immediately.

To learn from very few examples or even from one example seems to be an important ability for biological organisms. In the example mentioned above the problem splits in two important subproblems that could be solved by two more or less independent subsystems. One subsystem does the necessary computations for the recognition task and another subsystem weights the visual input and decides what is important and should be learned and what is not important. As the latter is task dependent we will only deal with it in so far as the user will have the ability to select relevant input that should be stored in a recognition data base. Coming back to the first subsystem and the specific problem of natural shape description and recognition it should be a great advantage to be able to recognize a contour just by one example seen so far without knowing much about other contours. It can be argued that evolution might have developed a set of local interactions given at birth that already contain the knowledge about natural shapes. In part II it will be shown that there is a lot of evidence in the biological literature that this is at most true for a very limited amount of shapes and furthermore the question arises how humans are able to learn such shapes as cars, ships or helicopters that are very different from every natural shape relevant during the evolution of mankind. In my opinion the real implication of the described problem is that biological learning has to be inductive, is has to be able to cope with a lot of 'exceptions'. Probably after learning a lot of individual shapes, the competition of various neural cell assemblies and the accompanying reorganization of the nervous system will result in some neuronal subsystems that will have specialized on a wide variety of similar natural shape parts. But this should be a consequence of learning each individual shape rather than be a precondition for shape learning, although once such a subsystem is existing it would probably facilitate the recognition and learning of new shapes substantially. On the other hand the found relations can probably only be of an unspecific nature — like for example the Gestalt rules of collinearity and curvilinearity treated in part II — corresponding to shape segments that are part of a lot of different shapes. Any attempt to guess the general laws describing the computations

of these general neuronal subsystems puts the really difficult task into the mind of the algorithm designer so that it will not be a feature of the algorithm itself. It further makes the implicit assumption that all shapes can be described by general laws which is at least questionable in my opinion. Coming back to the problem of creating a knowledge base desirable algorithms of contour processing should therefore be able to incorporate newly recognized contours to their knowledge base, while the way new structures are represented in that knowledge base should fulfill two requirements: First, the representation should be able to represent every possible natural contour by the same form of representation so that, second, this representation should preserve the similarity of contours 'perceived' by humans.

1.2. Requirements a Shape Model Should Ideally Meet

In the previous section one has seen that the first requirements are conceptual ones. Additionally there are further requirements to a shape recognition model which are summarized below. The shape model that is later presented in this work will either fulfill all these requirements or be extensible in future research to fulfill the remaining ones.

- The shape model and recognition algorithm should be able to deal with any contour, be able to extend its knowledge base incrementally — ideally interactively with user input — and make no *a priori* assumptions on the nature of natural object contours or depend on a representative sample of those contours.

- It should be possible to recognize efficiently complete contours that have been subject to affine transformations (rotations in the plane, scaling and translation) with a very high recognition rate.

- Furthermore it should be possible to recognized contours that have been slightly distorted or have been modified by noise.

- Ideally a continuous similarity value for all possible contour pairs should be possible to compute and the similarities computed should correspond to the impression that humans have on how similar these contours are.

- It should be possible to recognize objects only by parts to cope with occlusions or superpositions of objects.

- In addition it should be possible to recognize open contours that are parts of already stored contours. At least it should be possible to recognize those parts of the contour that are relative far away from the end points.

- Rotations in all three dimensions of the objects can probably not be covered directly although one could imagine the existence of features that are stable under three dimensional rotations to some extent. It is unlikely because not even humans can, for example, recognize a persons profile face silhouette if they have only seen the frontal face so far. But if contour information from enough different viewpoints has been acquired it should be possible to recognize objects that have undergone rotations in all three dimensions.

- If the shape model can be used for data compression of the contour information this would be useful for encoding purposes like fulfilling the MPEG-7 standard (Jeannin and Bober, 1999).
- The shape model used in tracking tasks should enable the user to generate point to point correspondences.
- If point to point correspondences can be established and stereo images are provided then it should as well be possible to estimate depth information, ideally with sub-pixel accuracy.

Only a fool relies on learning from his own mistakes. I personally always tried to learn from other peoples mistakes to prevent doing them at the outset.

Graf Otto von Bismarck, Chancellor Second German Reich

1.3. Implementations and Applications of Shape Models.

Shape representations can have a lot of possible applications. Although all of them could only be fulfilled having a shape model in the above mentioned sense a lot of models have emerged that are suitable for special applications and these will be presented in the following. Applications of shape models are for example to identify individual persons by profile face silhouettes, or the detection whether or not faces are present in an image like in (De Campos et al., 2000), or estimation of the pose of an articulated object from its silhouette (Kameda, 1993). Medical tumors can be identified by their shape like in (Korn et al., 1998) or hand gestures can be recognized like in (Pavlovic et al., 1997). Another application is the construction of 3D-models from several 2D-contours like in (Chien and Aggarwal, 1989). In the next sections a closer look is taken at the mathematical foundations of the shape models presented in the literature and furthermore on the algorithmic realization of different recognition methods.

1.3.1. Shape Models

A great challenge in itself is the development of algorithms that can first of all extract a contour from visual input. An example is the work presented in (Bell and Pau, 1990) where certain specified features are extracted that should lead to a useful curve description in a logic programming environment. While the contour extraction seems to be at first glance a precondition for contour recognition it turns out that knowledge about the possible shape of contours can help strongly in the task of extracting the 'right' contour from an image. So it is better to think of contour extraction as a test evaluating contour hypotheses with the actual low level visual input as a constraint rather than of a simple preprocessing step which may only yield good results in simple situations. The shape representation of (Chuang and Kuo, 1996) — explained in more detail below — can be used for example to extract known shapes by using the Campher distance measure on the image points and an elastic *matching* approach. Matching is defined here as the process of finding the points of two contours corresponding best with each other in terms of a defined similarity function.

The next important question to answer is the development of the shape model itself. There are two main approaches in the literature on the development of shape models: purely contour based ones and models that incorporate region-based information as well.

A region-based shape descriptor described in (Heijmanns and Tuzikov, 1998) is robust to a lot of invariance constraints like the class of affine transformations and reflections but is only limited to convex objects. In contrast to that the model described in (Zhu, 1999) learns Gibbs distributions on Markov random fields by evaluating specific properties of a collection of natural shapes. These properties include curvature on several scales and the distance between associated points that are defined by computing the medial axis of the shapes. While the model may be of great help in evaluating whether a new average contour hypothesis likely belongs to

a learned class it is not easily possible to store highly specific shapes that may not be repeating very often. Furthermore, a statistically representative gallery of know contours is a precondition for the model to work. Finally the stored models are very sensitive to changes of the medial axes of the contours that could occur due to occlusion or multiple objects forming a new combined contour.

In (Yang and Cohen, 1999a) cross weighted affine invariant moments are used to describe shapes of objects. The main advantage of the method is that it allows recognition to some extend even under perspective changes of the object of interest. On the other hand as the method is not very sensitive to small changes of the contour it loses discriminating power and it can not deal with large occlusions. In (Yang and Cohen, 1999b) locally affine invariants are computed by using the convex hull of scattered feature points. This approach can deal with occlusions but depends on the choice of the feature points and has limited discriminating power as only a convex hull of the shape is used and not the shape itself.

The problem with most methods of shape description — like Fourier descriptors or moments — is that they depend on characteristics of the whole contour and are highly sensitive to small disturbances of that contour. Therefore those methods are not suitable for partial contour recognition or recognition of contours with local variations.

There are three different ways models make use of Fourier descriptors. The traditional approach is to transform the contour coordinates to two real valued coordinate functions. The functions are either parameterized by the arclength of the contour or by using the *path length* which is the sum of the Euclidian distances of all contour pixels to their successor pixels. Another popular approach is the transformation to one real valued *turning* function based on the centroid of the contour and a description in polar coordinates. Only few models use the complex Fourier representation for recognition purposes. An interesting exception should be mentioned that is presented in (Kindratenko and Van Espen, 1996) were the natural shapes of an algae cell agglomerate could be classified by using the complex Fourier representation.

In (Arbter et al., 1990) a contour model based on normalized Fourier descriptors is presented. The parameterized x and y coordinates of the shape boundary points are each Fourier transformed. Then all normalization steps are performed in the Fourier domain. The authors claim that the results are descriptions that are invariant to affine transformations of the contours. However it is questionable how accurate this method can deal with rotations in the plane as the transformations are done directly on the pixel coordinates and no interpolation is done which would allow for pixel independent equidistant sampling. This can be a problem as for example a simple straight horizontal 8 pixel line segment transforms into a 7 pixel line segment when it is rotated by 45 degrees which results in a very different description of that same line segment in the Fourier domain. The authors even apply their methods to objects moving in 3D space with good results for almost rigid objects — like airplanes — as long as the movements and the resulting perspective projections do not lead to major occlusions. It has to be noted that the inability to deal with occlusions is an inherent feature of all models using only Fourier descriptors.

The problem of recognition of slightly varied contours has been addressed in (Gorman et al., 1988) where the contour is subdivided into several distinct parts and the first n Fourier descriptor coefficients of each part are used to characterize its curve form. The result is an alphabet of elementary curve forms that can be matched using a dynamic programming procedure. While this procedure has it's merits — especially speed — it is very sensitive to the decomposition of

the original curve into segments. Furthermore by using an alphabet of curve forms there is — without further consideration — no natural way to specify the degree of similarity of different elements of the curve alphabet even if their difference is not very high when viewed by humans. Additionally, an optimal alphabet is dependent on the task to be solved and it therefore may be too small to distinguish small differences of curve forms for some applications or too big with the result that similar curve forms are distinguished although the should be treated as equal.

An interesting extention of the method in (Gorman et al., 1988) is an approach to the problem of occluded contour recognition found in (Petrakis et al., 2002). There a contour is decomposed — using it's inflection points — in concave and convex segments which results in a code that can be matched using a dynamic programming procedure. While the proposed method is certainly an improvement over the use of raw Fourier descriptors, as it is possible to match partial contours, the method depends heavily on the stability of the inflection points. The procedure is probably more stable than the method used in (Gorman et al., 1988), but the principal problems of using a curve alphabet remain:

1. Dynamic programming is most useful if exact matches are searched for. What happens in case of misclassified or changed parts of a contour?

2. What is the optimal degree of similarity of different elements of the curve alphabet?

In (Ueda and Suzuki, 1993) the extraction of the optimum scale convex/concave structure common to shape samples of one specific class leads to a representation of this object class by one canonical form. No normalizations to rotation and scale are performed as it is hoped that the relevant features of the class are already captured by the constructed prototype. This approach is not suitable for rotational invariance and has the same disadvantages as the above mentioned algorithms as it uses a dynamic programming procedure for the matching of two contours.

In this work a method will be shown that allows the matching of occluded or extended contours without using contour points with special characteristics and therefore avoiding to be dependent on them and without the need to define an alphabet of curve forms.

There are several models using only the contour information: In (Kartikeyan and Sarkar, 1989) the contour is modeled using an autoregressive model developed for time series prediction. In (Sekita et al., 1992) PARCOR coefficients are computed which are extentions to autoregressive coefficients and can be shortly described as not only 'forward looking' but as well backward. While these are established statistical methods it remains unclear why the continuation of a contour should be predictable by these kinds of coefficients. Furthermore the discriminative power of the coefficients is probably not very high as their computation is more or less an averaging process which disregards by its very nature specific details of the contour.

The approach in (Tieng and Boles, 1997) is related to the work that will be presented here. The authors extract the border of an object from images using edge detection and tracing algorithms. The curves representation is transformed into polar coordinates to make the algorithm invariant to translation and rotation and is then sampled with a constant number of data points — that is a power of two — to make it invariant to scaling. They use the first derivative of a cubic spline function to build a wavelet transform zero-crossing representation of that object contour. The resulting dyadic wavelet representation is only used at the coarse levels, as the 2D-grid of the digital image is producing to many artefacts in the other levels. The dyadic wavelet representation is claimed to be superior to Fourier descriptor representations but can as well only match complete contours because of the use of polar coordinates.

A simple purely contour based shape representation and a metric for their comparison is proposed in (Arkin et al., 1991). The method can be computed efficiently and is based on the turning function which, is the parameterization of the original curve by its tangent angle relative to a given orientation. Invariance to rotation and starting point are achieved by testing for all possible shifts and using the minimum distance found. The method is very sensitive to nonuniform noise and needs exhaustive search methods to be invariant to rotational shifts etc. even in the case of closed contours where normalization to a canonical form is possible.

A purely contour based shape representation and matching procedure is proposed in (Mokhtarian and Mackworth, 1992) which is in some aspects similar to the model that will later be presented here. The model uses a path length representation of the curve which means that the length of the graph of the contour in the image is used to parameterize the contour in x and y coordinate functions. The representation is computed by convolving this path-based parametric representation of the curve with Gaussian functions — with the standard deviation of the Gaussian changing from a small to a large value — and extracting the curvature zero-crossing points of the resulting curves. The representation is essentially invariant under rotation, uniform scaling, and translation of the curve. The choice of the zero-crossing points for representation makes the representation of the curve unstable, although the multi scale approach can compensate that to some extend. An advantage is that the choice of the zero crossing points can lead to a compression of the contour data although the choice of the zero-crossing points seems to be arbitrary and possibly important contour information is disregarded. However, this is a very popular method and the authors have applied the model to the problems of three dimensional rotations (Abbasi and Mokhtarian, 2001) of objects, occluded object recognition (Mokhtarian, 1997), corner tracking and data compression in accordance with a MPEG-7 standardization (Mokhtarian and Bober, 2003).

In (Chuang and Kuo, 1996) a wavelet descriptor representation is used to encode the contour of a shape on a multi scale approach. The tested wavelets are orthogonal or biorthogonal which allows an easy reconstruction. The wavelet coefficients can be normalized to be invariant to translation, rotation and scaling as only closed contours are used. However the wavelet transforms are performed in the pixel domain and depend on the choice of the starting point. Although the authors claim the matching results are better for deformable contours compared to using Fourier descriptors their approach is not applied to occluded or enlarged contours.

1.3.2. Recognition Methods

There are two main approaches to solve the problem of contour recognition, one puts the emphasis on finding a canonical representative via normalization and the other is to develop a sophisticated matching procedure that copes with the different transformations the object has undergone. The idea of normalization is explained in detail in (Rothe et al., 1996). The main idea is to perform a number of normalization steps so that each contour of the same object subject to a specific number of transformations is normalized to the same canonical representative. Although even here a matching procedure is necessary to really find the wanted object in a database the matching procedure can be relatively simple as the canonical forms of two contours are either closely related to each other or not. If they are not closely related they should not belong to the same equivalence class of objects defined by the normalization steps. Obviously the main advantage is a huge reduction of the computational cost required to match objects.

It has even been argued that object recognition is the computation of invariants (Weiss, 1993). Following this line of thought the proposed framework in (Alferez and Wang, 1999) is very general and produces invariants insensitive to rigid motion, affine transform, changes of parameterization and scene illumination, noise, and perspective transform. One main disadvantage of normalization is that it must be a priori clear which transformations should be normalized. For some applications this might be inconvenient as it could turn out that some variants of a contour that are transformed into the same representative have to be distinguished. An example would be a hand gesture to the left or to the top of the head with only the hand being normalized. These two gestures could not be separated unter rotational invariance but should probably mean different things. One solution to this problem is to use only invertible normalizations and store the used parameters, so that the user can decide after the matching what equivalence classes to build. Another solution is that a good algorithm should allow the user to decide which normalization steps should be performed instead of hiding the normalization steps as the wanted invariances might change from application to application. Furthermore consider two different but dependent tasks. One is to recognize silhouettes of human faces, for example, looking to the right and the other is identifying specific persons from these profile images. The danger of normalization is that the individual differences needed for task two are 'normalized' away while solving task one.

In (Cohen and Wang, 1994) a B-spline representation of the curve is chosen for the goal of normalization under affine transformations. The problems to be solved here include estimating the control points from the data curve, and of deciding on the best order B-spline and the best number of control points to be used to model the contour. In (Wang and Cohen, 1994) a Fourier descriptor is used on the B-spline control points to achieve rotated, translated, and scaled independence for the matching process that is done via a neural network. The choice of the B-spline control points is critical and therefore a Bayesian approach is used to decide which is the 'best' B-spline representation. Unfortunately, the chosen representation is very sensitive to wrong decisions of this algorithm caused by small changes in the contour data. For this reason the B-Spline representation is used in (Avrithis et al., 2001) for the main purpose to be able to generate an equidistant sampling of the original curve and a Fourier descriptor is afterwards applied on the — via the path length — parameterized coordinates of that sampled curve. The number of sample points is fixed in this case. By using coordinate moment normalization and a phase shift in Fourier space the resulting representation can be normalized to translation, scaling, rotation and starting point invariance for closed contours. No matching procedure is given. It is not possible to deal with occluded or enlarged contours and the moment normalization can change the shape of the object significantly. While this may play a minor role for two equal contours that are rotated against each other the normalization may fail if the contours differ slightly.

1. Model Concepts, Extraction, Recognition and Retrieval of Natural Shapes

2. The Shape Model and Recognition Algorithms

What is important is to keep learning, to enjoy challenge, and to tolerate ambiguity. In the end there are no certain answers.

Martina Horner, President of Radcliffe College

When you know a thing, to hold that you know it; and when you do not know a thing, to allow that you do not know it - this is knowledge.

Confucius (551 BC - 479 BC), The Confucian Analects

2.1. The Contour Model: An Overview

In figure 2.1 a general overview over the most important steps that are performed for the various object contour recognition tasks is given. The first step to be able to process a contour is to extract potential contour points of an object from the visual input. For the purpose of this work — the recognition of objects by parts of their contour — it plays no specific role which algorithms are used to achieve the contour point extraction. There are only two important requirements regarding the output format any algorithm used for the first preprocessing stages has to fulfill to be suitable for the transformations on the contour data that will be presented below in this work:

1. For a closed complete contour the algorithms should return a counterclockwise oriented graph of the boundary points of the potential contour. In case one node has to be used several times the sequence of its edges should determine the sequence of points to follow so that it is possible to store the node data — with repetitions if necessary — in sequential form. The choice of the starting point plays no important role at this stage.

2. For open contours the counterclockwise constraint is even relaxed and the only requirement is that a sequential chain of neighboring contour points is returned, starting and ending at the start and end of the open contour.

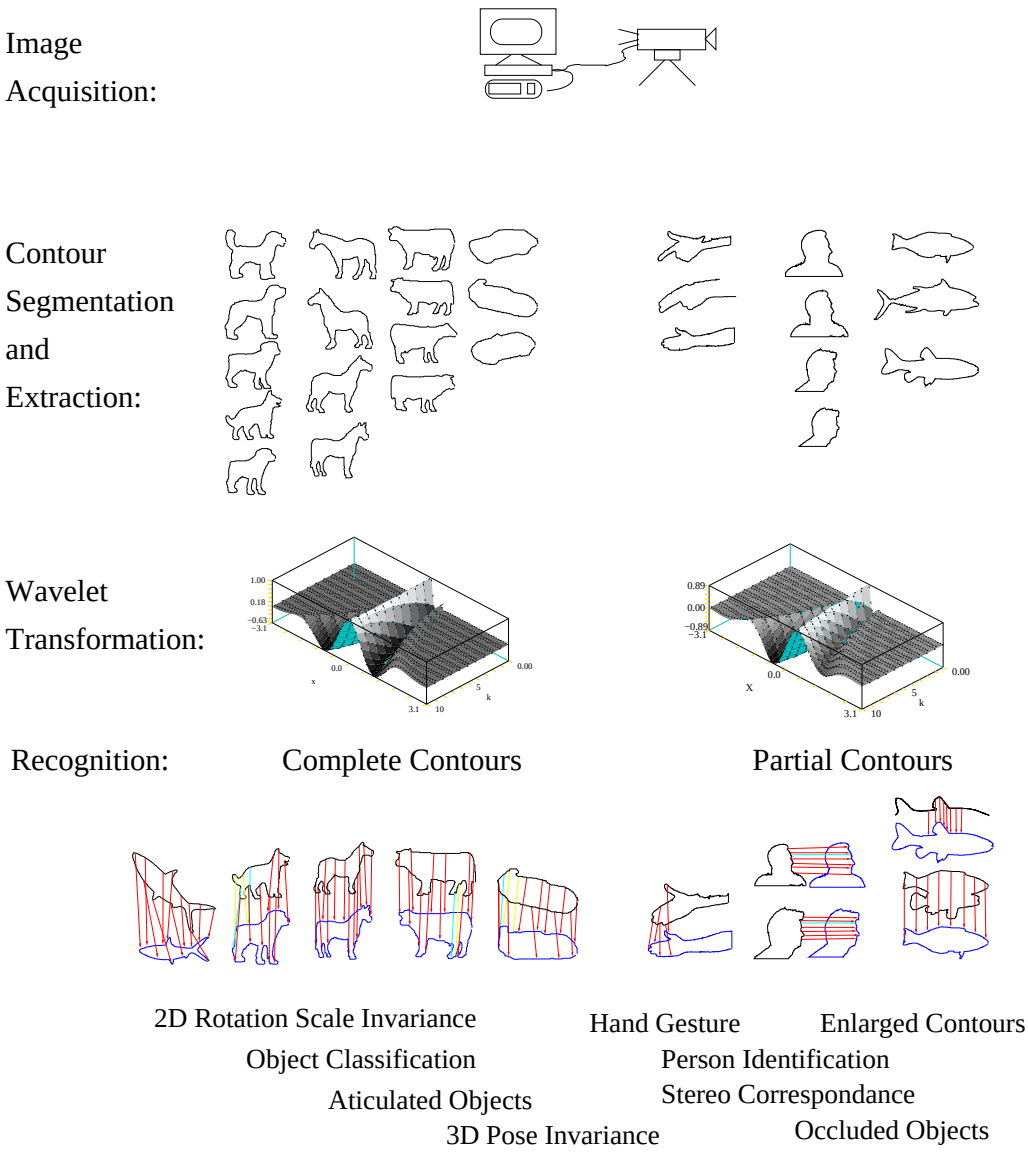


Figure 2.1.: Illustration of the complete contour recognition system. After the acquisition of a new image a segmentation is done which produces image regions that correspond to objects. The region or object boundary is extracted and transformed using one dimensional Gabor functions as wavelet kernels. The resulting representation can be used for various recognition tasks.

Of course the methods used to generate the results presented will be explained in the following chapter. The reader mainly interested in the contour representation and recognition may safely skip the following sections and read further on in section 2.6 on page 37.

2.2. Pre-Matching Processing Stages.

2.2.1. Raw Shape Extraction by Background Difference Segmentation

As one does not have a suitable contour description method yet that could be used to incorporate high level knowledge in the process of separating a figure from the background called *segmentation* one mainly has to use low level cues to segment the contour of an object from the visual input. Basically two types of cues are of interest for the extraction of an potential object contour from the visual data: edge based ones and region-based ones. Although they seem to be quite different at first glance they are related to each other as the border of two large regions forms a significant edge and in some cases a lot of edges can be grouped together and form a whole region.

A great amount of research has been done on the latter approach: grouping individual single edge parts by local interactions together to form a larger contour or even better a closed contour representing a complete object. The second part of this work can be viewed as a contribution to this line of thought as a biological explanation for simple grouping of local edge detectors based on collinearity and curvilinearity will be presented there. However, regardless if one is using biological edge detectors or classical technical edge detectors — like the Sobel operator — the amount of edges detected is very huge. Especially for textured objects it seems to be impossible — except for trivial cases — to group together the 'right' edges of an object contour in one processing step. The reason for this difficulty is that the grouping procedures have to make either crude simplifications, apply very general principles of contour continuation or they produce intermediate results that are so ambiguous — because of the large amount of possible edge continuations — that is seems unlikely that these algorithms can be successful without any *a priori* knowledge about the viewed object itself. Especially as the alternative method of generating several hypotheses for potential contour continuations still faces the question on how to decide — without user intervention — which of them is the 'right' one.

The amount of edges can not be reduced significantly by using 2D multi-scale approaches like the Mallat transformation (Mallat, 1989) as they produce a significant amount of edges on the finest scale. Even while a reduction of the number of edges can be achieved on the coarser scales the precise localization of these edges in the image data becomes harder and harder on these coarser scales and there is no natural unambiguous way to connect the edge information of the different 2D scales, as the analysis of multi-scale data representations is a difficult challenge on its own (Lindeberg, 1994). To avoid being caught in the hen-and-egg problem, of not knowing how to group edge information together without already knowing how likely contours look like, region-based segmentation methods for edge extraction under controlled circumstances are used in this work.

Of course a suitable region-based approach has to be found as most of them use some form of averaging over a neighborhood of pixels, which blurs the precise location of a possible edge. For this reason the color cue (illustrated in appendix B.1 figure B.1 on page 120) is the basic cue used in this work as it is a region-based cue — similar colors can be grouped together —

that does not reduce the accuracy as it can be evaluated on a pixel basis. However, the color cue alone is not sufficient as it is not possible for this cue — even for a good choice of threshold (figure B.1 C) — to know which object parts that have different colors should nevertheless be grouped together because they belong to the same object. Especially if an object is composed of parts with different colors. Of course, using a threshold dependent cue is a disadvantage in itself as one does not want to have the problem of finding a good threshold every time new visual input should be processed. These problems induce the question, answered below, of how one can learn — at least under controlled circumstances but without adjusting a lot of parameters, like thresholds — an object contour model by storing extracted contours that are known to belong to an object class. At the same time avoiding the problem that multiple objects may belong to the same extracted region or that an extracted region contains only a subset of the parts of an object.

2.2.2. Segmentation Based on Differences of an Image to a Learned Background.

To solve the problem of generating a gallery of nearly perfectly extracted contours that can later be used for evaluating the validity of a new contour the strategy of computing the difference of an image containing the object of interest with an image of an already known background is employed. As noted earlier, if the object has more or less the same color the color cue alone may be sufficient, but this is rather an exception than the rule.

The first precondition is the fixation of the camera and the avoidance of zooming operations while the visual input is acquired. Then an image of the background and an image of the object of interest presented before that background are recorded. Both images are transformed to the L*a*b* color space, see figure 2.2 for an illustration. The question arises: how important is the choice of the color space. The answer in short is: very important. The color model L*a*b* is based on the model proposed by the Commission Internationale (CIE) in 1931 as an international standard for color measurement. In 1976, this model was refined and named CIE L*a*b*. In this work the short form L*a*b* is used for the exact full nomenclature 1976 CIE L*a*b space. L*a*b* is the second of two systems adopted by CIE in 1976 as models that better showed uniform color spacing in their values. L*a*b* is an opponent color system based on the earlier (1942) system of Richard Hunter called L, a, b. Color opposition correlates with discoveries first made in the mid-1960s of color-opponent cells that are present in the retina, dorsal lateral geniculate body, striate cortex and the color processing areal of the primate brain area V4. For a more recent study on the spectral properties of the color-opponent cells in the various locations of the brain see (de Monasterio and Schein, 1982). Approximately retinal color stimuli are translated into differences between light and dark, red and green, and blue and yellow. The L*a*b* color space indicates these values with three axes: L*, a*, and b*. The central vertical axis represents lightness (signified as L*) whose values run from 0 (black) to 100 (white). The equations for the conversion of a RGB color value to a L*a*b* color value are given in appendix B.2. The color axes are based on the fact that a color cannot be both red and green or both blue and yellow at the same time because these colors oppose each other. On each axis the values run from positive to negative. On the a* axis whose values are in the interval [-86.2,98.2], positive values indicate amounts of red while negative values indicate amounts of green.

Illustration of the L*a*b* color space

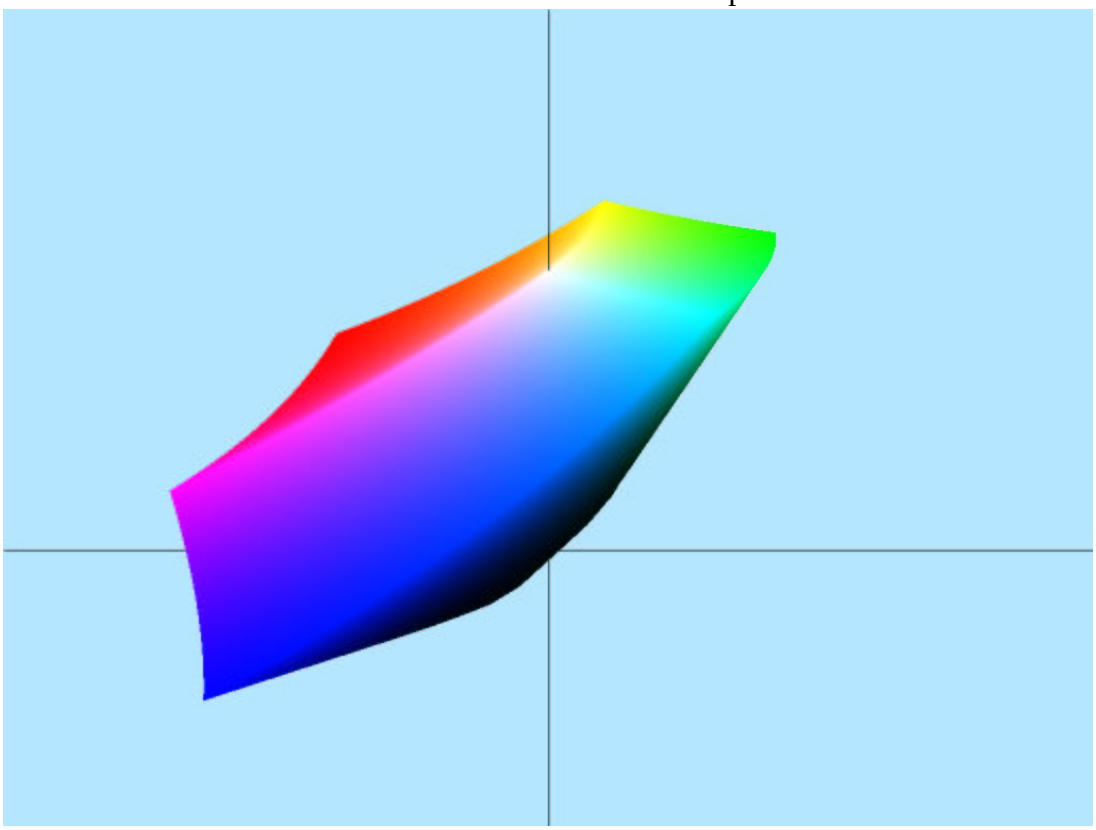


Figure 2.2.: Illustration of the L*a*b* color space. Shown is the transformation of the discrete RGB cube defined on $[0, 255]^3$ to the L*a*b* color space, which is the color space used in this work for computing color differences. Please refer to section 2.2.2 on page 28 for an in-depth explanation.

On the b* axis values are assumed that fall in the interval [-107.9,94.5], yellow is positive and blue is negative. The reason the interval boundaries for the three axes are not extended to round figures is to keep a close correlation to the biological measurements on color-opponent cells.

Of course the axes of the L*a*b* coordinate system do not correspond exactly with the psychophysical impressions of the colors which one can see in figure 2.2 which shows a view from the direction of the negative b* axis rotated by 45 degrees around the a* axis. The color yellow for example slightly deviates from the axis b* axis. For both axes a* and b*, zero is neutral gray: Therefore, values are only needed for two color axes and for the lightness or gray-scale axis (L*), which is separate (unlike in RGB, CMY or XYZ where lightness depends on relative amounts of the three color channels). L*a*b* has become very important for desktop color.

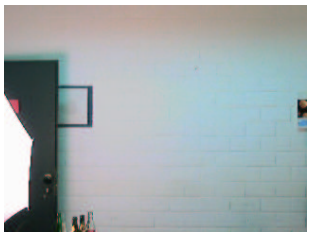
Because of its biological origins the L*a*b* color spaces together with a Euclidian metric induces a distance function between color values that correlates closely to human perception. That means that the degree of color difference can be quantified even relative to luminance differences. Note that the range of values for the color components a* and b* are each approximately twice as large as the values for luminance inducing a natural distance measure in so far as detected color differences are weighted larger that simple luminance differences. Therefore after recorded images are transformed into the L*a*b* color space the pixel wise difference of those images based on an Euclidian metric can be evaluated and the result is in a color similarity measure that corresponds to the psychophysical impressions humans have. Under these controlled conditions it is relatively easy to extract the pixels belonging to the object by using a constant threshold, that only needs to avoid the noise signals generated by the camera.

The immense gain of using the L*a*b* color space is best illustrated by showing what would have to be done to generate a psychophysically acceptable distance function for color differences if a different popular color space like for example HSI is used. This is illustrated in appendix B.3 on page 121 where a special emphasis was put on suppressing wrong detections of color changes induced by the shadow of an object.

2.3. Clustering of Pixels to Segments.

We now use an image consisting of non negative float values created by taking the absolute values of the difference image (in the L*a*b* color space) computed by subtracting the new image and the background image pixel wise, see figure 2.3 for an illustration.

Thanks to using the L*a*b* color space we can apply a constant threshold of greater or equal to four to the absolute values computed and create a binary image. Only those pixels whose absolute difference is above or equal to the threshold are marked with one. All others are marked with zero. This non negative image is the input to an algorithm for grouping neighboring pixels that are different in both images together to form segments. The first step of this grouping algorithm is to create a directed graph representation of the image implemented by an array (*graph image*) of the same size — as the images mentioned above — containing integer values. All pixels of the locations of the absolute value difference image not equal to zero are considered to be nodes of the graph and the integer value that is stored in the corresponding pixel location of the graph image represents an directed edge in so far as it is an index pointing to a connected



(a) Background

(b) New Image

(c) Difference Image

Figure 2.3.: Illustration of the computation steps for generating a background difference image. Shown are in A) the original background image. The new image B) recorded before the same background as in (A) and containing an additional object of interest. C) The difference image of the both images. Please refer to 2.3 on page 30 for an in-depth explanation.

node (pixel location). For an image of width W the index i is computed using the equation:

$$i = x + y \cdot W \tag{2.1}$$

This data structure allows the creation of directed chains in the graph terminating in a one-cycle by letting the last node of a chain point to itself (to its own index in the graph image). Now a forward pass is done on all components of the graph image starting at the top left position — represented by index zero — and ending at the down right position — represented by an index equal to the number of pixels in the image minus one — of the image. If a position is a node (absolute float value was greater or equal than the threshold) we check if its right neighboring pixel is a node as well and link the two nodes together by letting the node point to its right neighbor node and the right neighbor node point to itself. The same procedure is done for the neighboring pixel one line upward and one pixel to the right and again for the neighboring pixel one line upward and for the neighboring pixel that is located one line upward and one pixel to the left. Essentially this represents a eight pixel or — as it is sometimes called — a three times three neighborhood search. Note that the other points of this neighborhood do not need to be checked as we check them later or have already checked them in the forward pass. If one of the previous line pixels should be connected to the current pixel we reassign the whole chain starting at this neighborhood pixel to point to the current index. This enforces a monotonic ordering of the indices which means that the index a node points to is always equal or greater to its own index. After we have worked on all pixels we do a backward pass starting at the down right pixel of the image which has the aim of labeling all the pixels of the connected regions with a unique label. Pixels that were not worked on are assigned to the background by a label number of zero. If we

find a node in the graph image that points to itself we increase the label number and assign this increased number to the corresponding pixel. If the index points to another node we traverse the chain until we find the final leaf (node pointing to itself) which already has a label number assigned and assign that label number to the pixel under examination. With these two passes we have transformed the absolute value difference image using the graph image array into an array containing uniquely labeled regions which will further be referenced by as *segments*. The algorithm used for clustering the pixels of the difference image that are above a certain user defined threshold is based on the work of David Kastrup (Kastrup, 1997), which should be used for a more detailed description of the implementation. See as well the work of (Born and Völpel, 1991) for a listing of possible applications of this algorithm and see (Tarjan, 1972) for more information on the basic data structure used.

2.4. Generation of Counterclockwise Oriented Graphs Composed of Border Points of Segments.

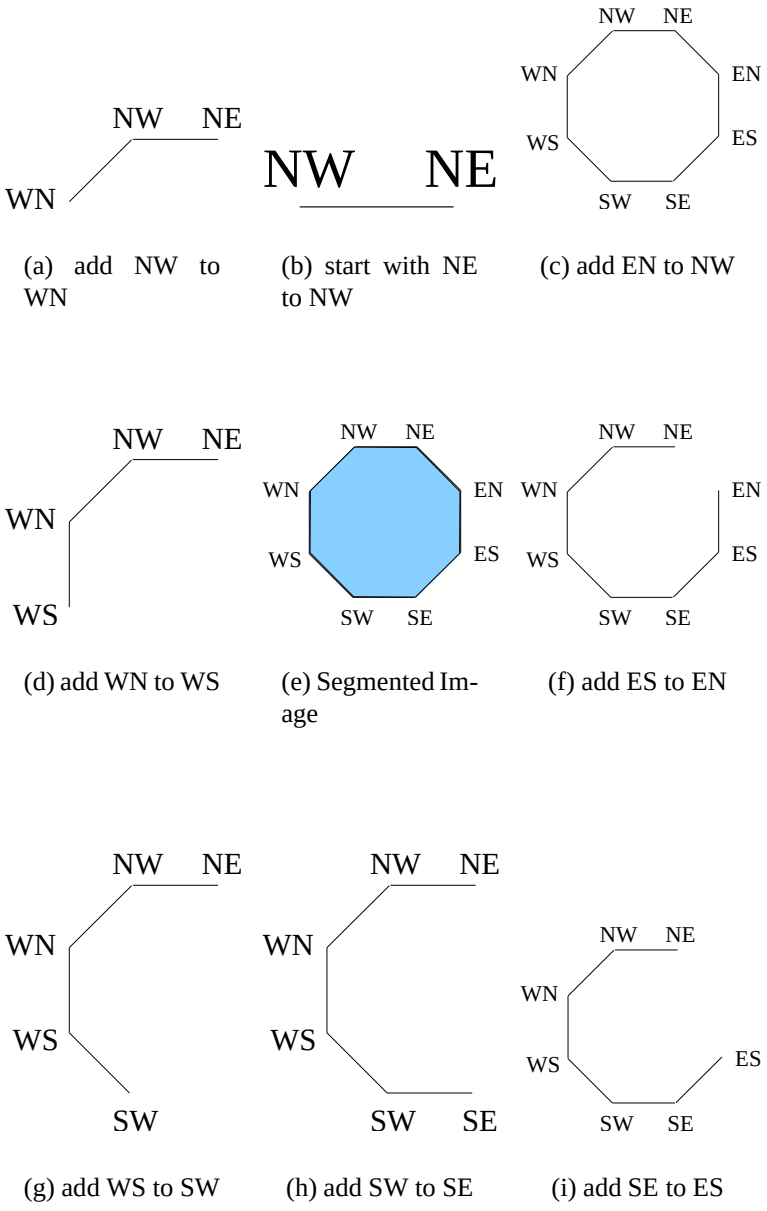
Due to noise in the camera recordings, light flickering or occlusions of the object of interest one can not assume that there is always only one segment detected even if only one object was present in the image. Once the segments of interest have been extracted from the visual input all points belonging to each segment are checked if a non-segment pixel exists in their three times three neighborhood. Pixels belonging to other segments can not be found in such a neighborhood, as this would have resulted in the two segments to be merged together into one by the algorithm presented in (2.3). All pixels that belong to the group of pixels having non segment pixels as neighbors could possibly belong to the border of that segment and hence belong to an object contour and will be called *external border pixels*. Of course there may be — for the reasons mentioned above — several very small segments and therefore a size cue is used to select segments that have a high probability — because of their relative huge size — to represent at least a part of an object. Another reason to use a size cue is that if the segment is very small and hence its resulting boundary, the recognition of the object becomes too ambiguous — due to lack of sufficient input — and it is not very useful to process it anyway. However the small segments should not be dropped completely for the purpose of segmentation as it might turn out later that this small segment could be a part of a larger object. A complete general segmentation method, however, is not the main aim of this work, so the small segments are ignored here. The size cue depends on the image size and the number of external border pixels. Experiments with images of size 320 times 240 and 512 times 512 have shown that the quotient of the square root of the number of image pixels and the number of external border pixels should at least be 15. Now a graph is formed with these external border pixels as nodes. If a node has an adjacent node in its three times three neighborhood an undirected edge between those nodes is added to the graph. Note that this is not the best way to create initial connections between nodes if we have a perfectly segmented object. There are several algorithms that can generate directly an oriented chain of border pixels that represent the contour. However, these algorithms assume a near perfect segmentation of the object from the background which is seldom the case in situations that are not completely controlled by the user. Often holes are present within the segmented regions and some borders of these regions are not smooth but scattered. Therefore another approach is chosen here which will produce oriented contours under any conditions

even at the price that some parts of the border may not be represented in the best way they could be represented. For the purpose and the examples used in this work this is sufficient as all contours are perfectly captured. An algorithm that is able to handle all possible inputs is subject to future research. Once the check for the existence of neighbors has been done for all nodes, the number of independent components of the graph and the number of cycles is analyzed by the algorithm `CyclesAndComponents` explained in detail in appendix C.1 on page 125.

It is necessary to check for the number of components of the graph and process them separately as independent components may exist — just imagine the object of interest is a tire with a hole in its middle — that are not caused by disturbances of the segmentation process but properties of the object. In any case the largest component will be used further on. To decide if additional components should be used as well a size cue similar to the one for segments described above is employed. The effect is to suppress additional components that are to small, and are probably only an artificial result of imperfect segmentation. Each of the selected components is treated separately further on and a directed graph is created using the nodes of the component with two edges connecting each pair of neighboring points (one for each direction). Note that there can be several graphs now that all belong to the same segment.

As two dimensional images are the input one can compute the extreme points of each graph which means one looks for the nodes corresponding to the pixels with the highest and lowest x and y coordinate. Even in the simple case of an octagon this computation is not unique as there may be several pixel with, for example, the highest x coordinate. Therefore four groups of pixels are build which contain the above points and these groups are labeled according to the points of the compass with west (W), north (N), east (E) and south (S). For each of these groups we compute again the extreme points which means for the W and E group the nodes with the highest and lowest y coordinate and for the N and S group the nodes with the highest and lowest x coordinate. Using this procedure eight extreme nodes named WN, WS, NE, NW ... and so an — not necessarily distinct — are obtained. Now a new directed graph called *counterclockwise oriented graph* is built where the last connection between extreme points that is added has the largest path length of adjacent extreme points. This constraint together with the counterclockwise sequence of the extreme points defines automatically the starting node used for the counterclockwise graph creation. The algorithm is illustrated in figure 2.4. Let the starting node be the NE which means the letters stand for the node farthest to the east of the nodes that are farthest to the north. One now tries to compose the counterclockwise oriented graph by trying to find a path from the NE node to the NW node, from the NW node to the EN node, the EN node to the ES node and so on. Although the idea behind this procedure can be easily described its implementation — see appendix C.2 on page 127 — is a lot more difficult to realize.

The algorithm works very well on closed contours and can as well be used for open contours, although not the full open contour may be captured as the end points of the open contour may not coincide with two of the extreme points and in the case of closed contours hairs (see appendix C.2) will be ignored if extreme points are not part of them. An algorithm for extracting the full open contour under any circumstance and all the hairs of closed contours is subject for future research. For the purpose of this work the presented algorithms are sufficient. Note that a rearrangement of the graph would probably be necessary if a more advanced algorithm is applied as now the convention is used that the first node of the graph is the starting point which is as well one of the extreme points. Another possibility is that the information which



(a) add NW to WN

(b) start with NE to NW

(c) add EN to NW

(d) add WN to WS

(e) Segmented Image

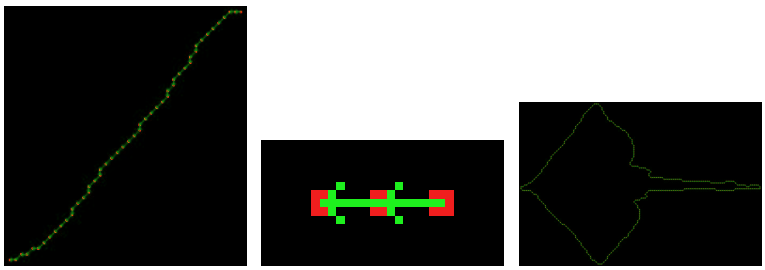
(f) add ES to EN

(g) add WS to SW

(h) add SW to SE

(i) add SE to ES

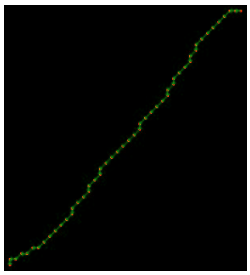
Figure 2.4.: Illustration of the computation steps for generating the counterclockwise oriented graph from an artificially created image shown in (e). The other figures show the step by step creation of the graph using the extreme points of the graph labeled after the points of the compass. In the example NE represents the point farthest to the east of the extreme points of the graph farthest to the north. Please refer to 2.4 on page 33 for an in-depth explanation.



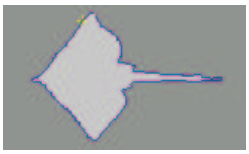
(a) add NW to WN

(b) start with NE to NW

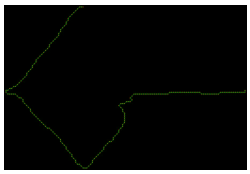
(c) add EN to NW



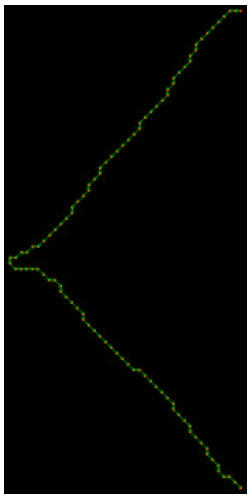
(d) add WN to WS



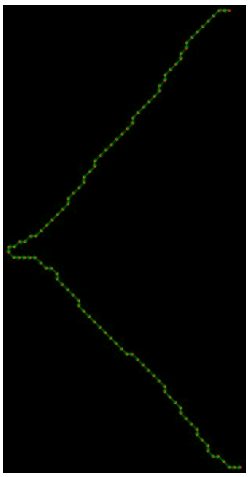
(e) Segmented Image



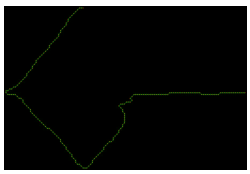
(f) add ES to EN



(g) add WS to SW



(h) add SW to SE



(i) add SE to ES

Figure 2.5.: Illustration of the computation steps for generating the counterclockwise oriented graph from the segmented image in (e). Shown is the step by step creation of the graph using the extreme points of the graph labeled after the points of the compass. Notice that some of the extreme points illustrated in figure 2.4 are almost identical in this example. Please refer to 2.4 on page 33 for an in-depth explanation.

nodes should be connected comes from other sources than the above described segmentation mechanism. For this reason the sequence of nodes created by the above algorithm is not used directly. Instead the counterclockwise oriented graph that was composed above is transformed to a sequence of points by the algorithm described in appendix C.3 on page 129. The main reason for this is that the implementation is open for input coming from other sources than the presented segmentation.

2.5. Spline Interpolation of a Chain of Contour Points.

2.5.1. From Discrete Contour Points to Equidistant Sampled One Dimensional Parameterized Data.

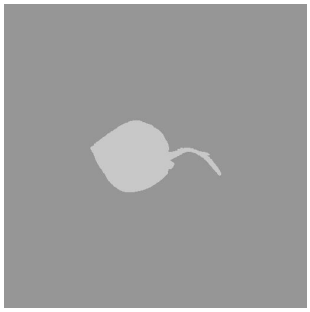
The counterclockwise oriented graph created in the previous section has been transformed into a chain of adjacent nodes and corresponding pixel coordinates, respectively. The first step required for the interpolation procedure is to split the chain of pixel coordinates into X and Y coordinate functions. The path length — computed as the sum of the distances between successive pixels as described in the previous sections — is used as abscissa for this parameterization with $t_{\max}$ being the maximum length from the first to the last pixel. The discrete coordinate functions $X(t_i)$ and $Y(t_i)$ of the contour graph are now defined on discrete points within the interval $t_i \in [0, t_{\max}]$ on the N_c indices corresponding to the number of pixels found. The next step is to embed the discrete coordinate functions $X(t_i)$ and $Y(t_i)$ into continuous ones $X(t)$ and $Y(t)$ by using a spline interpolation.

The type of spline interpolation used plays no important role for the purpose of this work and for reasons of easy availability a standard cubic B-spline algorithm (De Boor, 1972; Lyche and Schumaker, 1973) was chosen. The algorithm is using a derivative constraint that acts as a low pass filter which is described in detail in (Ooyama, 1987). As most readers are probably familiar with spline algorithms but not necessarily with derivative constraints in such algorithms their use will shortly be motivated. Consider a continuous function $u(x)$ defined in the interval $[x_0, x_M]$ and a filtered approximation of this function $\hat{u}(x)$ that minimizes the following integral:

$$\min \quad = \quad \int_{x_0}^{x_M} \left[(\hat{u}(x) - u(x))^2 + \left(\frac{l_c}{2\pi} \right)^{2k} \left(\frac{d^k \hat{u}}{dx^k} \right)^2 \right] dx \tag{2.2}$$

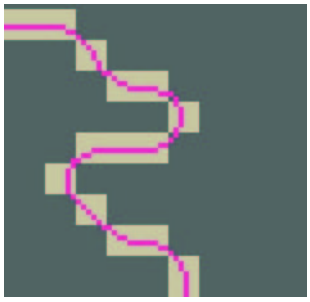
Here k denotes the order of the derivative that has the function of a low pass filter in the equation causing a $(2k)$th degree taper in the spectral response and the user definable constant l_c which is the cutoff wavelength where the amplitude response is half. If you want more insights into the derivation of this equation please refer to (Ooyama, 1987) on page 2503 where a full analysis is given. Linear combinations of cubic B-spline basis functions are substituted for $\hat{u}(x)$ in equation 2.2 with $u(x)$ being a discrete function defined on the given data points and the integral reducing to a sum. It makes only sense to use derivatives of order 1,2 or 3 as the basis functions are only differentiable up to order two at the nodes but still integrable over the domain for order three. In this work only derivatives constraints of order 1 are used.

The algorithm uses a heuristic to find automatically a sufficient number of equidistant knot points used for computing the B-spline approximation of the x and y coordinate functions.



(a) Segmented Image

(b) B-Spline



(c) Zoom

Figure 2.6.: Illustration of the B-spline interpolation. (a) Segmented Image. (b) The contour of the segmented object interpolated by the B-Spline. (c) A zoom of a part of the interpolated contour located at the upper contour of the tail of the marine animal. Please refer to 2.5.1 on page 36 for an in-depth explanation.

The first and the last knot point corresponds to the first and last data point. It is assumed that the maximum difference of the parameterized discrete coordinates, e.g. $\max(|x(t_i) - x(t_{i-1})|)$ with $i \in [1, N_c - 1]$ for x and y, respectively, multiplied by two is a reasonable cutoff wavelength l_c to avoid over- or under fitting of the data points. Note however, that if large gaps between the data points are expected one should multiply with four instead. The number of knot points minus one is equal to the number of equidistant knot point intervals and the size of these intervals is computed by the following heuristics:

- Compute the maximum number of equidistant knot point intervals so that at least one data point is in each interval.
- As long as the cutoff wavelength divided by the knot point interval size is less than 2 increase further the number of knot point intervals per cutoff wavelength, but only as long as one can maintain at least 2 data points per knot point interval.
- Do not increase the number of knot point intervals if there are already 15 or more knot points per cutoff wavelength.

If a dense sampled contour curve is the input, the algorithm reaches the final number of knot point intervals in the first step. This means as the maximum distance of two neighboring pixels is $\sqrt{2}$ the cutoff wavelength is chosen to be $2 \cdot \sqrt{2}$ and the number of knot point intervals is $N_c - 1$. So the x and y curves, respectively, will be represented using N_c knot points. The number of interpolation points N, which are later used to perform the various transformations on the contour data has not been chosen and their actual computation has not been done up till now. This number is now either chosen by the user or is computed automatically by finding the closest multiple of powers of 2,3,5,7, or 11 to the length of the path of the contour in the image multiplied by a user given constant. The special choice of these prime factors is caused by the properties of a special fast Fourier transformation algorithm that will be used later which is very efficient on data sets of magnitudes that are multiples of powers of these integers. This is the reason why, for example, the number 1536 will be often used in later sections as it is composed of prime factors 2 and 3 and is almost always greater than the path length of the contours. Having computed the B-spline approximations and having chosen the number of equidistant interpolation points N one now samples the continuous coordinate functions $X(t)$ and $Y(t)$ to obtain two vectors $\vec{X} = (x_0, \ldots, x_{N-1})$ and $\vec{Y} = (y_0, ..., y_{N-1})$ representing the interpolated contour data. A third complex vector $\vec{C} = \vec{X} + i \cdot \vec{Y}$ is constructed from those two.

2.6. Normalizations and Fourier Descriptors

2.6.1. Application of Fourier Descriptors to Equidistant Samples of X and Y Point Graph Coordinates.

With each of the vectors $\vec{X}, \vec{Y}, \vec{C}$ a discrete Fourier transform is performed. For non closed contour graphs or open contours different boundary constraints should be selectable. However currently only the *wrap around* or *cycle* boundary constraint is used where the ends of the contour are neighbors and which is often inducing a large gap and artificial high frequency responses in the Fourier domain. It is much more preferable to use for example a *mirror* boundary

constraint by traversing the contour from one end to the other and back again and this is subject to future research. The mirror boundary constraint avoids the large gap that occurs at the end of the data points relative to the first data point and is therefore better than the one currently used as less additional high frequency signals are artificially generated in the Fourier domain.

One could compute the Fourier coefficients $\breve{C}$ of a transformed $\vec{C}$ vector directly from the values of the Fourier transforms of $\vec{X}$ and $\vec{Y}$. For the sake of a more convenient notation we note the vector $\vec{C}$ and its Fourier transform $\breve{C}$ explicitly here.

$$\breve{J}(\omega) \quad = \quad \frac{1}{N} \sum_{n=0}^{N-1} J(n)\, e^{-i\,\frac{2\pi}{N}\,\omega\, n} \tag{2.3}$$

J should be substituted for $\vec{X}$, $\vec{Y}$ or $\vec{C}$. Please note that ω can only have discrete values.

2.6.2. Normalizations Applied to the Data Before the Fourier Transformations.

All contours are translated so that the center point of the contour becomes the origin. A special scale normalization is not applied as the same result can be achieved for closed contours by fixating the number of interpolation points. For open contours — especially if they should be later compared with parts of closed contours — scale normalization seems to be not a very wise strategy as it is not possible to know before the matching how much of a known closed contour will be covered by the new open contour. Therefore the possibility exist — see above — to let the number of interpolation points that should be used be dependent on the path length of the contour. All further scale invariances are then put in the matching procedure.

2.6.3. Normalizations Applied to the Model After the Fourier Transformations.

It is not necessary to do an explicit translation invariance transformation as the Fourier component with index zero has already vanished as the centroid of the contour was already shifted to the origin. In the case of closed 2D contours one can achieve rotational invariance by normalization. One has to compute a standard shift angle using the Fourier transformed values of $\breve{C}$ given by:

$$p = p(\breve{C}) \quad = \quad \left[\frac{N}{4\pi}\,(\phi_1 - \phi_{N-1})\right] \bmod 2\pi \tag{2.4}$$

In this equation $\phi_i \in [0, 2\pi)$ represents the phase of the i-th Fourier component starting at i = 0. By rotating the interpolated coordinate vectors $\vec{X}$ and $\vec{Y}$ with the angle $-p$ using the rotation matrix Q the contour is shifted to some kind of normalized orientation, which is roughly described by a rotation of the first principal axis of the contour to the horizontal axis.

$$Q \quad = \quad \begin{pmatrix} \cos(-p) & \sin(-p) \\ -\sin(-p) & \cos(-p) \end{pmatrix} \tag{2.5}$$

$$\begin{bmatrix} \vec{X}_{\mathrm{rot}} \\ \vec{Y}_{\mathrm{rot}} \end{bmatrix} \quad = \quad Q \begin{bmatrix} \vec{X} \\ \vec{Y} \end{bmatrix} \tag{2.6}$$

Afterwards the Fourier transforms $\breve{X}$ und $\breve{Y}$ are computed again using $\vec{X}_{\text{rot}}$ and $\vec{Y}_{\text{rot}}$. Note that it is not evident that equation 2.4 is suitable for achieving rotational normalization. A detailed derivation of equation 2.4 is given in (Avrithis et al., 2001).

2.7. Gabor-Descriptors: Gabor-Wavelet Transformation Applied to Pairs of One Dimensional Data

For each new contour one has to decide how many different *Gabor coefficients* — defined itself in equation 2.11 and equation 2.12 below — should be computed at each of the N interpolated points. As the same number of coefficients is computed for each point of the contour one speaks of *Gabor levels*. The different Gabor coefficient levels can be labeled using the different values for k_m that are used in their computation in equation 2.9. Using $k_{\max} = \frac{2\pi}{\lambda_{\min}}$ ($\lambda_{\min} = 4$) as the average frequency of the Gabor kernel of level zero one can compute for each Gabor level — labeled using the index m — a corresponding k_m .

$$k_m = k_{\max} \cdot \left(\frac{1}{f}\right)^m \tag{2.7}$$

In equation 2.7 f is the size factor from Gabor level to Gabor level. Typical values used for f are 2 for octave intervals, $\sqrt{2}$ or $\sqrt[4]{2}$. The last value $\sqrt[4]{2}$ results in a very dense sampling in frequency space which is useful for matching invariant size. How many different Gabor levels ($m_{\max} + 1$) should be used is computed using the equation:

$$m_{\max} = \left\lceil \frac{\log\left(\frac{N}{\sigma \cdot c_D}\right)}{\log(f)} \right\rceil - 1 \tag{2.8}$$

$m_{\max}$ is then rounded to the next highest integer. The user given constant c_D is normally set to 6 which means that the distance of $2 \cdot \sigma$ left and right of the Gabor with the largest wavelength cover approximately one third of the contour. Remember that the Gaussian function of the Gabor has decreased to five percent at the distance of two sigma so it is save to assume that the coefficients computed for this Gabor level are not significantly influenced by much more than one third of the contour. It is possible to raise the importance of smaller contour details in the similarity computation by using a larger c_D value which can be useful for specific applications. For each application task this variable is not changed. In any case the number of levels used for a contour increases linear with an increase of the logarithm of N.

2.7.1. Computation of the Gabor Kernels

The Gabor kernels used in Fourier space are:

$$\breve{\psi}_m(\omega) = \left(\exp\left(-\frac{\sigma^2}{2}\frac{(k_m - \omega)^2}{k_m^2}\right) - \exp\left(-\frac{\sigma^2}{2}\frac{k_m^2 + \omega^2}{k_m^2}\right)\right) \tag{2.9}$$

Note that these kernels are purely real valued functions and that σ is equal to π for all experiments that will be presented later in this work. By multiplying the Fourier transformed contour

points and the corresponding Gabor kernels in Fourier space and transforming the results back to the spatial domain a transformation equivalent to the *Gabor wavelet transformation* is done without the disadvantage of an reduced amount of data points.

For the different combinations of $\breve{J}(\omega)\,\breve{\psi}_m$ — with J being $\vec{X}$ or $\vec{Y}$ — inverse discrete Fourier transforms are done:

$$\upsilon = \omega\,\frac{N}{2\pi} \tag{2.10}$$

$$x_m(n) = \sum_{\upsilon=0}^{N-1}\left[\breve{X}(\upsilon)\,\breve{\psi}_m\right]\,e^{i\,\frac{2\pi}{N}\,\upsilon\,n} \tag{2.11}$$

and

$$y_m(n) = \sum_{\upsilon=0}^{N-1}\left[\breve{Y}(\upsilon)\,\breve{\psi}_m\right]\,e^{i\,\frac{2\pi}{N}\,\upsilon\,n} \tag{2.12}$$

One has computed now a number of $2\times(m_{\max}+1)$ complex valued coefficients which are denoted as $x_m\,(n^1)$ and $y_m\,(n^1)$ for an arbitrary interpolated point of the contour given by index n^1. In fact two complex values — one for x and one for y coordinates — for each Gabor level $m\in[0,m_{\max}]$ and each index $n\in[0,N-1]$ of each contour have been computed. A combination of coefficients referring to the same contour index n in vector form $\vec{f}=(x_0,y_0,x_1,y_1,...,x_{m_{\max}},y_{m_{\max}})^T$ is called a feature set or short *feast vector* or even only *feast*.

Often one wants to denote feasts that belong to different contours q, however in this work most of the time only one index n for each contour q is used at the same time so that we can introduce the following short form of notation: $x_m^q = x_m^q\,(n_q) = \vec{f}_{x_m^q}\,(n_q)$. The Gabor coefficients indexed by n_q belong to the q-th interpolated contour and the same short form can be used for the y Gabor coefficient values. Only if we refer to more than one index of the same contour the long form is used.

2.8. Matching Modi

As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality.

Everything should be made simple, as simple as possible, but not simpler.

Albert Einstein

2.8.1. Problems Associated With the Matching Procedures

Consider this situation: One has computed the Gabor coefficient values called *feast* — see the previous section — at all interpolated points of one new and unknown closed contour. Furthermore to a set of contours — called *gallery* or *target gallery* — the same processing has been applied and the resulting feast vectors at all of the interpolated contour points of these gallery contours have been stored. The question arises how one can compare the new contour with the gallery contours so that similar contours are successfully matched. The first problem to be solved — subject of the next section 2.8.2 — is how one individual feast vector belonging to one specific point of the new contour can be compared to any of the feast vectors located at the points of any of the gallery contours, so that a similarity measure between zero and one can be assigned to each of the possible feast pair combinations. The problem to be solved becomes even harder if the number of Gabor levels used for the new contour differs from the number of Gabor levels in the gallery contour, as the resulting feasts will have a different number of Gabor coefficients and therefore a different size.

However, even if this problem of comparing or matching individual feasts of different contours is solved, the important question arises how to chose the interpolated points at which comparisons should take place? To illustrate why this is a problem consider the case that we would have computed only Fourier descriptor coefficients instead of the Gabor descriptor coefficients. The Fourier descriptor coefficients are of course only defined in the frequency space and essentially there is no local information accessible except in possible phase shifts that are caused by different choices of the starting point. The advantage is that one would only have to compare this more or less unique set of Fourier descriptor coefficients. The results of the comparison would probably be very good if a procedure was found to normalize the starting point choice for the contours. On the other hand if for example half of the contour was modified there is no easy way to recognize the remaining unchanged part by comparing Fourier descriptor coefficients. However, if Gabor descriptors are used then one could compare feasts of the unchanged part of the contours but it is possible and it will be seen in the results section even quite likely, that local information at one specific point of one contour has high similarities to a lot of sites in another target gallery contour. Therefore the need arises to compare several neighboring feasts of the new contour with the gallery contour feasts. The relative distances of the new contour feasts measured by the difference of their corresponding indices in the contour parameterization define a *topological ordering*. The new challenge arises to preserve this topological ordering of the new contour feast points in the matches to be found in the gallery contour. The brute force method of evaluating the similarities associated with all possible feast

pairs is very expensive computationally and one still faces the question of how the topological ordering can be preserved. The elementary part of a topology preserving matching algorithm is to find the highest similarity between some new contour feasts — given by different indices — and gallery contour feasts that are each part of specific but yet unknown ranges of gallery contour point indices under the constraint that the relative topological ordering of the new contour feasts is preserved. So for the computation of these ranges one has to solve two major problems: First, the computation of these ranges is dependent on the relative sizes of the two contours and of some other variables and this problem will be subject of section 2.8.3.

The second problem in the computation of these ranges occurs on an even larger scale: the starting point problem. Not in all cases it is known a priori in which index interval range of the gallery contour one should look for the best match for any given new index feast to be matched. Therefore, one tries different index offsets around which to place the boundaries of the corresponding index range. This starting point search procedure is necessary in general as one can not assume that for example the starting index zero of the interpolated point indices in the new contour and in the gallery contour correspond to the same point of these contours even if both contours are derived from the same object. The reason is that the location of the zero index on the object contour depends on the segmentation methods that have been used. For special cases like for example the case of closed rotationally normalized contours, which both represent the complete object, this procedure is not necessary.

If it is a priori not known in which interval range of the gallery contour one should look for the right match a starting point finding procedure has to be done, before any fine scale matching is applied. The aim is to find a rough index correspondence between the new and the gallery contour that probably will lead to the highest similarity between those two contours when finer matching procedures are used. This procedure for finding an appropriate starting point correspondence is explained in detail in section 2.8.4.1. The matching of *complete contours* that represent one object is certainly a major task in contour matching and a matching procedure to solve this problem is given in the following sections. However, in many applications like individual profile silhouette recognition of faces the users wants only a specific part of the new contour to be matched against the stored gallery contours. Therefore the possibility of interactively choosing a interval range for comparison is introduced called *selected feasts* matching. Furthermore, for the purpose of segmentation the need arises to find good matches for contours of occluded objects or for contours that are composed of many objects. In the last case there are probably large parts of the contour identical to parts of contours stored in the gallery but one does not know which ones. Therefore the possibility of matching a certain user definable percentage amount of the new contour with the gallery contours called *percentage matching* is introduced. All of these matching procedures like complete contour, selected feasts or percentage matching are described in section 2.8.5.

2.8.2. Similarity Computation Between Two Gabor Coefficient Vectors (Feasts) of Different Sizes: The Discrete Scale Concept.

Before one can compute the similarities between two computed Gabor coefficient vectors the scale concept has to be introduced. It is best illustrated by an example. Consider the Gabor coefficient vectors of two contours with N^1 and N^2 interpolated points. To compute the similarity of two points belonging to two contours at index n^1 and n^2, respectively, one first

looks at the feast vectors $\vec{f}_1$ and $\vec{f}_2$ that have been computed at the corresponding points. $\vec{f}_1 = \left[(x_0^1, y_0^1) \ldots \left(x_{m_{\max}^1}^1, y_{m_{\max}^1}^1\right)\right] \in (\mathbb{C} \times \mathbb{C})^{m_{\max}^1+1}$ and $\vec{f}_2 = \left[(x_0^2, y_0^2) \ldots \left(x_{m_{\max}^2}^2, y_{m_{\max}^2}^2\right)\right] \in (\mathbb{C} \times \mathbb{C})^{m_{\max}^2+1}$. Contour point indices have been omitted for a more convenient notation.

If, for example, $m_{\max}^1 = 9$ and $m_{\max}^2 = 6$ there are four different possibilities of scale shifts — abbreviated as "*scales*" in the following — to compare the two Gabor coefficient vectors with each other. If one starts the comparison computation with the pair (x_0^1, y_0^1) and (x_0^2, y_0^2) it is a match on scale zero. If first (x_0^2, y_0^2) is used together with (x_3^1, y_3^1) for the similarity computation then a match on scale three is performed. For the possible scales $s \in \mathbb{N}_0$ the following formula holds true: $0 \leq s \leq |m_{\max}^1 - m_{\max}^2|$. If one wants to compare single feasts in a scale invariant way a similarity for each possible scale can be computed and the maximum of all similarities is taken. However this procedure is not recommended as it is better to iterate on all scales and match all chosen feasts of the new contour in a topology preserving way on one fixed scale with the gallery contours. Afterwards one decides which scale produced the best matching result. The alternative is a match result that has feast correspondences on different scales which is rarely useful to the user.

The similarity computation for two feasts on scale s — denoted as $S_s(\vec{f}_1, \vec{f}_2)$ — is given in the equations below. First the maximum number of levels available for comparison is computed: $m_{\min} = \min(m_{\max}^1, m_{\max}^2)$. For each level we compute a non-normalized similarity of the Gabor coefficients separately for the x and y coefficients und multiply the results, so that a high similarity is only the result if both, the similarities of the x coefficients and of the y coefficients, are high. Note that the similarity computation is done component by component, which means that x and y coefficients measures belonging to their corresponding feast components are first combined and then the other components are added. It turns out that this procedure is much superior to the alternative way of computing first a similarity measure for all x coefficients and then combining this similarity with a similarity for all y coefficients. Without loss of generality one assumes that $m_{\max}^1 \leq m_{\max}^2$. If this condition is not true $\vec{f}_1$ and $\vec{f}_2$ are swapped. The notation of $\vec{f}_1, \vec{f}_2$ and of the indices is dropped in the notation of the right side of the subsequent equations.

$$P_s(\vec{f}_1, \vec{f}_2) \quad = \tag{2.13}$$

$$\sum_{m=0}^{m_{\min}} \left|x_m^1\right| \left|x_{m+s}^2\right| \cdot \cos\left(\phi\left[x_m^1\right] - \phi\left[x_{m+s}^2\right]\right) \cdot \left|y_m^1\right| \left|y_{m+s}^2\right| \cdot \cos\left(\phi\left[y_m^1\right] - \phi\left[y_{m+s}^2\right]\right)$$

The difference of the phases $\phi\left[z_m^1\right] - \phi\left[z_{m+s}^2\right]$ of two complex valued coefficients is most important for the similarity computation and not their absolute distance in the complex plane. This phase difference is mapped onto the interval $[-1, 1]$ by using the cosine function. The reason why the absolute values of the coefficients are used as well in the formula for similarity computation is that most of the times a more reliable phase estimate goes hand in hand with a higher absolute value.

Using the above formula and an upper estimate 1 for the cosine function one obtains the

maximum value that could be computed by the above formula:

$$M_s(\vec{f}_1, \vec{f}_2) \;=\; \sum_{m=0}^{m_{\min}} \left[\, |x_m^1| \cdot |x_{m+s}^2| \,\right] \cdot \left[\, |y_m^1| \cdot |y_{m+s}^2| \,\right] \tag{2.14}$$

If M_s is equal to zero S_s is set to be zero as well, else one can divide (2.13) with (2.14) and a number in the interval $[-1, 1]$ will inevitably be the result. This number is shifted to the interval $[0, 1]$ by:

$$S_s(\vec{f}_1, \vec{f}_2) \;=\; \left(\frac{P_s}{2M_s} + \frac{1}{2}\right) \in [0, 1] \tag{2.15}$$

2.8.3. Computation of a Target Gallery Index Range

For the computation of similarities between a feast associated with a point of the new contour given by its index and a specific part of the gallery contour the method of computing a target index range is used. For different new contour feasts these target ranges represent the part of the target gallery contour that should be the target of each individual match. Recall that is it possible to associate a contour part with an index range as the indices label the subsequent interpolated points of the contour. Therefore neighboring indices are associated with neighboring points of the interpolated contour. The alternative to match one feast against all feasts of a gallery contour is not suitable as this prevents the possibility of more complex topology preserving matching procedures. Furthermore it would increase the computational costs substantially. For each feast of the new contour that should be matched it is therefore better to chose an index range R_s — given in equation 2.22 — that limits the number of feasts that may take part in the matching procedure. The range itself is dependent on the selected scale s (see section 2.8.2) and the number of interpolated points of the new (n_C) and the gallery contour (n_G). By controlling the size of this ranges one can find the right balance between topology preserving and invariance to slight distortions of a contour in the matching process. The index range size determines as well the overlap of different target index ranges that will be associated with neighboring new contour feasts indices. The procedure of computing an adequate target range of indices is easiest illustrated if we think of the case of a complete closed contour that should be matched using feasts at R — defined by the user — equally spaced indices. In all index computations for closed contours a circular metric is of course assumed. That means all computed indices are used modulo the maximum number of indices N of the corresponding contour. Ranges that would fall even partially outside of the natural index range $[0, N-1]$ are ignored for open contours. The number of indices n_C of a new closed contour is subdivided into R interval ranges of size $n_I = \frac{n_C}{R}$ and the centers of these interval ranges are computed using the following equation:

$$d_r(n_C, R) = \frac{n_C}{R}\left(r - \frac{1}{2}\right) \quad \text{with} \quad r \in \{1, \ldots, R\} \tag{2.16}$$

In the complete contour matching mode these centers $d_r(n_C, R)$ are assumed to index the feasts of the new contour chosen for the matching procedure. The aim is now to find the interval range R_s of gallery contour indices for every one of these chosen new contour feasts. Within the interval R_s the best match is searched for. Let the index of the current new contour feast

be n_i^1. The index for the gallery feasts n_j^2 is initialized with the lower boundary of the interval R_s. The search process itself evaluates the similarity of the corresponding new contour feast $f_{n_i^1}$ with the gallery contour feast $f_{n_j^2}$ by using equation 2.15. The process is repeated with further indices that are computed by adding an increment inc to the last index n_j^2 used as long as the upper index range boundary is not reached. By this procedure several different gallery contour feasts $f_{n_j^2}$ are addressed. In the actual computation not only the parameter for the index incrementation inc is used but as well an offset parameter O. Both are controlled by higher level procedures and can for an easier understanding be ignored at first reading as their purpose will be explained in later sections. The maximum similarity found for all of those feast pairs is assigned to the maximum similarity of the new contour feasts and the gallery contour part indexed by R_s. For an increment of inc = 1 and O = 0 the maximum similarity is denoted as:

$$S_s(f_{n_i^1}, f_{R_s^2(O,n_i^1)}) \quad = \quad \max_{n_j^2 \in R_s} (S_s(f_{n_i^1}, f_{n_j^2})) \tag{2.17}$$

To make the whole procedure reusable for other matching modi one does not assume that one already knows a relationship between the indices of the chosen feasts of the new contour even so this relation is already known for the example case of complete closed contours that is used here for illustration. To get a first starting value for the size of the wanted interval one first computes the distances of the indices of the chosen new contour feasts $\Delta d_r = d_{r+1} - d_r$ using the mentioned periodic boundary constraints in case of closed contours. The case can occur that only one feast is chosen for matching. Then the computation of the index interval is trivial as of course the whole contour is a possible target for the matching procedure. The next step is twofold: first map the selected new contour indices using the selected scale and the relation of the number of points of the two contours on the indices of the gallery contour. The mapping target is some kind of basis index ($c_M \cdot d_r$) around which to compute the interval R_s with c_M given by:

$$c_M \quad = \quad \begin{cases} f^{-s} & \text{for} \quad n_G \leq n_C \\ f^{s} & \text{for} \quad n_G > n_C \end{cases} \tag{2.18}$$

Starting at the basis index $c_M \cdot d_r$ the second step is to compute the lower I_s^{low} and upper bound I_s^{up} using the factors p_{near} and p_{far} and the distances of the indices Δd_r:

$$p_{\text{near}} \quad = \quad \begin{cases} c_c \cdot f^{-s} & \text{for} \quad n_G \leq n_C \\ c_c \cdot f^{s-1} & \text{for} \quad n_G > n_C \end{cases} \qquad p_{\text{far}} = \begin{cases} c_c \cdot f^{1-s} & \text{for} \quad n_G \leq n_C \\ c_c \cdot f^{-s} & \text{for} \quad n_G > n_C \end{cases} \tag{2.19}$$

c_c is a safety factor that adds a tolerance to the intervals. Recall that f was the quotient of sigmas used in the computation of subsequent Gabor levels. Dropping the index r on the left-hand side of the equations the wanted interval bounds are then given by:

$$I_s^{\text{low}} \quad = \quad c_M \cdot d_r \quad - \quad \Delta d_r \cdot c_O \cdot p_{\text{near}} \tag{2.20}$$

$$I_s^{\text{up}} \quad = \quad c_M \cdot d_r \quad + \quad \Delta d_r \cdot c_O \cdot p_{\text{far}} \tag{2.21}$$

$$R_s(O) \quad = \quad [I_s^{\text{low}} + O\,,\; I_s^{\text{up}} + O] \tag{2.22}$$

c_O is a constant correction factor controlling the search interval size in the gallery contour relative to the distance of the neighboring feasts in the new contour. Buy increasing c_O more

flexibility is given to the matching procedure and if c_O is set to zero a rigid template matching would be done. O is an offset given by the higher level routines (see 2.8.4.1) to cope with the possibility that different points of the interpolated contours have been associated with the zero index. Even for contours that are derived from the same object one can not be sure that the index corresponding to the zero index of the new contour can roughly be found in the interval range of the zero index of the gallery contour as this depends on the segmentation and graph creation method that has been used. The procedure dealing with this problem is called the *starting point move* and will be explained in detail in the next section 2.8.4.1.

Let there be L interval ranges for which similarities have been computed. In the complete closed contour example $L = R$. One can now compute a similarity $S_{s,O,inc,p}(C_C, C_G)$ of the new C_C and the gallery C_G contour for a specific scale s that was chosen. The other parameters are a specific choice of new contour feast to match $f_{n_i^1}$ with $i \in [1, L]$, a specific interval range to interval range mapping given by the offset parameter O, a specific accuracy given by the increment parameter inc and a given percentage parameter p whose introduction will be explained in detail in section 2.8.5.3. L_p is the percentage p of L rounded to the nearest integer but in any case it is not less than one.

$$S_{s,O,inc,p}(C_C, C_G) \;=\; \max\left(\frac{1}{L_p}\sum_{i=u}^{u+L_p} S_s(f_{n_i^1}, f_{R_s^2(O,n_i^1)})\right) \quad \text{with} \quad \mathrm{u} \in [1, \mathrm{L}] \tag{2.23}$$

For closed contours all ranges R_s^2 can be computed because of the use of circular boundary conditions. In case of open contours only if all L_p summands had their R intervals fall in the legal index range then the resulting similarity was used for the $\max$ operator. This is necessary to avoid matching over the borders or match results with fewer than the wanted number of matched feast pairs. Furthermore u for open contours falls in the interval $[1, \max(L - L_p, 1)]$.

2.8.4. Efficient Multistage Matching: The Starting Point Move and Scale Move, Rotation Move and the Corresponding Points Move.

2.8.4.1. Starting Point Move or Scan Move

The different procedures to find the best matching results under different conditions are called *moves*. The motivation for this designation is that the whole process can be compared to a template matching procedure which means that a rigid template contour — one of the gallery contours — is *moved* around the new contour. The aim is to find the interval range to interval range correspondence with the highest similarity of the template and the new contour. The foundation on which all moves fall back in the end is the algorithm presented in section 2.8.3. The designation *move* is of course only a crude approximation as this algorithm is much more complex than a template matching algorithm. Recall that the individual links — feast to feast matches — are allowed to float freely within the limiting index interval associated with the index of each chosen new contour feast.

The starting point move is sometimes necessary to find the parts of the gallery contour that should be matched in detail with the chosen feasts of the new contour without having to compare all possible feast pair combinations. As there is a sequential ordering of the points of the contour preserved in the indices used to address them, the only thing necessary is to find an offset that has to be added to the gallery contour indices so that the best match of the zero

index of the new contour is found in an interval around that offset. This leads to the idea that it should be sufficient to find the gallery index of the best match of one chosen feast and simply compute the offset by taking the difference of the two indices. Unfortunately, this procedure would be too inaccurate as the matching result of only one feast is too ambiguous. Even to find the right offset $O_{\max}$ it is necessary to match several feasts in a topology preserving way. For this reason the matching procedure described in section 2.8.3 is used with different offsets O_r and an increment inc substantially greater than one — which prevents an explosion of the computational costs — to find the best interval range to interval range mapping for the matching process.

It is possible to use an increment greater than one because of the special nature of the Gabor feast coefficients and the way similarities are computed in equation 2.15. The result of this combination is a slowly decreasing similarity around the best matching position. This would not be the case for all other possible kernels of wavelet transformations, especially not for orthonormal ones. The reason for the decreasing similarity is that the matching success for the high frequencies is more or less random when using large increments. As the positions matched are so far away in terms of sigma of the high frequency kernels that the computed coefficients are virtually not affected by the contour information at the right point making the resulting contribution to the overall similarity more or less random. The effect is on average a decrease of the computed similarity of the best match result found.

Furthermore as the amplitudes of the Gabor coefficients decrease strongly with an increase of the frequency of the Gabor kernels the computation of the similarity value computed is designed to decrease only by a relatively small amount if only coefficients of high frequency kernels are changed. The effect of choosing large increments is therefore comparable to matching low pass filtered versions of the contours. The similarity computation is done in equation 2.15 that computes the similarities of two feasts. To be more precise the similarity is computed by equation 2.13 and 2.14 and the mentioned property is achieved as these equations give coefficients with larger amplitudes a larger contribution to the overall similarity computation than coefficients with smaller amplitudes.

The starting point move is not always necessary. For example if translation, scale and rotation normalization can be used it is possible to resample the resulting normalized contours. In case it is known that similar contours are examined the condition is fulfilled that an interval around the zero index of the gallery contour is the interval range where the best match for the feasts associated with the zero index of the new contour should be found. An example where the starting point move is therefore not necessary is the usage of the matching algorithm for tracking purposes when a new contour is matched with the contour extracted from a previous frame of a continuous video stream. Given a sufficient frame rate relative to the speed of the object the above mentioned condition should be fulfilled anyway and the computational costs associated with the matching process can be reduced substantially by omitting the starting point move. Lets now take a closer look on how the starting point move is implemented. As the search interval sizes are related to — and generally larger than — the distances of the feasts in the new contour equation 2.16 on page 45 is used again to compute the various offsets O_r for open and closed contours. These offsets are tried in the starting point move together with an increment for the gallery index computations that is equal to the square root of the interval range size $\frac{n_C}{R}$ used for the offset computation.

$$O_r \quad = \quad d_r(n_C, R) \tag{2.24}$$

$$inc \quad = \quad \max\left(\sqrt{\frac{n_C}{R}}, 1\right) \tag{2.25}$$

2.8.4.2. Corresponding Points Move

If a starting point point move — see previous section — was already done the R different similarities computed for the different offsets O_r are compared. The offset $O_{\max}$ associated with the highest similarity is used for a final move together with an increment of $inc = 1$ that finds the best matching point to point correspondences. If no starting point move should be done the similarity for an offset equal to zero is computed. The work itself is again done by using the algorithm from section 2.8.3. This time the computed similarity is finally assigned to the match of the new and the gallery contour and returned to the higher level routines.

2.8.4.3. Scale Move

The full scale move is only necessary if the same object is expected to occur in significantly different sizes and it is pointless to do a scale normalization by using a constant number of interpolation points. This is for example the case if an open contour is a part of a complete closed gallery target contour as then it is obvious that it makes no sense to sample both contours — the complete and the partial contour — with the same number of interpolation points. The implementation of the scale move is simple and straightforward. Compute all the possible scales s as shown in section 2.8.2 and do the matching tasks presented in the previous sections on each of these scale for each pair of a new (C_C) and a gallery contour (C_G). The scale $s_{\max}$ which produces the maximum similarity is finally associated with the similarity of this contour pair.

$$S(C_C, C_G) \quad = \quad \max\left(S_s(C_C, C_G)\right) \quad \text{with} \quad \mathrm{s} \in [0, \left|\mathrm{m}^1_{\max} - \mathrm{m}^2_{\max}\right|] \tag{2.26}$$

One can speak of a best match on scale $s_{\max}$ and with this information and the factor f of the Gabor transformation it is possible to estimate the scale factor between the two contours.

If a specific scale s_U was preset by the user no matching on all scales is done but only on this specific scale. The similarity associated with the contour pair is then simply equal to the similarity computed for the user given scale.

$$S(C_C, C_G) \quad = \quad S_{s_U}(C_C, C_G) \tag{2.27}$$

2.8.4.4. Rotation Move

The aim of the rotation move — like the scale move — is to have an alternative to cope with rotations in the plane in case it is pointless to do a rotation normalization. Especially for open contours or closed contours that are suspected to correspond at least partially to a part of a larger contour — for example a hand of a human that is part of an arm or the whole body contour of a person — there is no obvious way to do a rotation normalization. Therefore a rotation move is necessary and is given here although no simulation results for this move will be presented in the results sections.

For significant rotations in three dimensions views from a lot of different viewing angles or several different gallery contours of the objects are necessary for the recognition. Is this information not provided it is generally impossible to predict the transformation of an object

contour under three dimensional rotations without already knowing what kind of object it is, an initial guess of the rotation parameters and a three dimensional model of this object added to the algorithms knowledge base. This has to do with the fact that object parts can appear or dissappear which no possible transformation can cope with having just one contour as knowledge base. The alternative is to treat different poses of the contour independently in the matching by comparing the contour to contours of different rotated versions of the object so that the object class or identity, and the rotation parameters are both estimated after the matching is done by using the best match found.

In contrast to all other moves the rotation move is done before the Gabor coefficients have been computed. Theoretically it is possible to do transformations on the Gabor coefficients directly to account for different rotations but this has not been realized and can still be subject to future research although the rotation of the contour data itself is very simple. However, if the starting point and rotation problem could be addressed simultaneously the development of transformations dealing with different rotations on the Gabor coefficients themselves could lead to a substantial reduction of the associated computational costs. But up till now the implementation is computationally expensive as the contour is transformed using different rotation angles applied to the rotation matrix given in equation 2.5 and equation 2.6.

The different rotation angles are computed starting at zero and adding the user given constant p_{inc} successively until the range of $[0, 180]$ degrees has been covered. Then the algorithms of the previous section are applied on the rotated contours in accordance with the user given choices which normalizations and algorithms should be applied. Note that is only necessary to cover the rotation from $[0, 180]$ degrees as the Gabor coefficients for a contour that is rotated by 180 degrees will give the same similarity values as the ones from the unrotated contour if the starting point move is done. The highest similarity for all of those rotated contours is then assigned to the new contour - gallery contour pair under examination. Usually it is sufficient to use a rotation increment of 10 to 20 degrees but we will see later that the discriminating power of the resulting similarity values is task dependent and even more dependent on the number of contours used in the gallery. Therefore the value can be changed by the user to suit the specific requirements of each matching task.

Rule for success: Never hunt two rabbits at the same time.

Graf Otto von Bismarck, Chancellor Second German Reich

2.8.5. Local versus Global: Matching of Selected Feasts, Matching of Complete Closed Contours and Percent Matching.

A lot of research focuses on finding a general matching procedure for shapes that is sufficient for all possible tasks. However, the first question to be answered should be ”can such an algorithm exist at all?” In this work this question can not be answered with mathematical precision but from the experience gained from a lot of experiments the impression has occurred to me that such a general algorithm is probably not existing without being configurable by the user because of the task dependent nature of the evaluation of what is to be considered a successful match result. Therefore at least the intention of the user would have to be an input to such an algorithm which is reflected in the work presented by several users definable constants and different modi that can be chosen. While the ability of invariant matching especially for the class of affine transformations either by normalization or by matching procedures was the main topic of the last sections the problem of the different nature of objects is the motivation for the different matching procedures presented in this section. For example some objects are only characterized by their crude form. Let us just think of an ellipsoid or other elementary geometric objects. There are no distinguished object contour parts that could be recognized alone that at the same time give statistically significant evidence that the object under examination is an ellipsoid as every part of this contour can be found in many other objects as well. One needs to examine almost the whole contour in order to be sure what kind of object is present. Not to mention that several objects for example an apple and an orange show only small differences in their contours as large parts are practically identical. On the other hand, consider for example a shark. Humans and the presented matching procedures only need to see the dorsal fin of a shark to recognize the object as a shark. In other cases there is no single contour part that is sufficient but if several different distinguished contour parts can be found, that are in loose but defined geometric relations to each other, this can be sufficient for the object to be recognized. Therefore the more important question seems to be: What are the elementary matching operations that should be possible to perform so that more complex matching task requirements can be implemented by a user or even furthermore be found unsupervised using an *organic computing* (http://www.organic computing.com/, 2000) approach.

2.8.5.1. Complete Closed Contour Matching

The reader should already have understood at this point the algorithms used to implement the modus of matching one complete closed contour with another one as all examples in the previous sections starting at section 2.8.3 have been based on this case. Under these conditions both new and gallery contour can be normalized to reduce the computational costs. The match results can not only be used to classify objects that belong to different classes but it is as well possible to identify individual objects that belong to the same object class. Furthermore it is possible to define object classes by using the matching results reversely. If the computed similarity of two contours belonging to different objects is high enough the two objects can be labeled as belonging to the same object class. Naturally, the question arises, how high the similarities should

be so that two objects should be considered to belong to the same class. As will be shown in detail in the results section 3.1.2 for matching with 25 different interval ranges on the data set of 1100 marine animals from the SQUID database (Mokhtarian and Kittler, 1996a; Mokhtarian and Kittler, 1996b) similarity values of greater equal 0.93 seem to be closely related to what humans would consider to be the same object class and object posture. On the other hand if the similarity value is below 0.90 it is highly unlikely that the two objects belong to the same object class or at least that they have a similar posture. Of course there cannot be an absolute threshold describing what should be grouped together as this is task and user dependent. In the range of 0.90 to below 0.93 another matching step would have to be done that is specific for the task to be solved and which is therefore not part of the presented work. If another task dependend matching step is not a valid choice then one should use 0.92 as a single threshold. This threshold captures most of the right object matches while the number of wrongly classified objects is still relatively low.

Quite contrary to the concept of thresholds it is especially the strength of the presented matching algorithm to compute a continuous similarity measure between arbitrary contours. If the classification of objects is wanted the user can find thresholds or threshold ranges for each matching task that reflect the wanted generality of the clusters, trigger specific further matching operations or see at least that no sharp defined clusters exist and therefore realize that his classification approach was ill defined — at least in terms of object shapes — right from the beginning.

2.8.5.2. Matching of Selected Feasts

In contrast to the matching of complete contours the *selected feast* matching approach is most useful if one or more contour parts exist that alone have a high significance for object identification purposes or if only the differences of a specific contour part are of interest. Examples are individual profile face recognition or hand gesture recognition which will be given later in section 3.2.1. An advantage of this matching mode is that what is supposed to be a high significant contour part can either be chosen interactively by a user or be selected by supplying one or several image points in absolute coordinates. If several image coordinates are given they are used successively and for each of these points the shortest distance to the segmented and interpolated contour is computed to identify a corresponding contour point. The index of this contour point is used further on and around this index at most A with $A \in \mathbb{N}_0$ additional indices to the left and right are chosen. This procedure leads at most to $2 \cdot A + 1$ chosen feasts for each selected point. The full number of points is always used for closed contours as circular boundary conditions are assumed but for open contours indices outside of the index boundaries are ignored. The distances of these additional indices are computed using the parameter R that is used to divide the total number of indices to give the index distance increment. As a rule of thumb one uses a higher value for R for example 50 instead of 25 for complete closed contour matching so that the feasts are closer to the selected index. Then the similarity is computed using the algorithms of the previous sections. If several points where selected the highest similarity is finally used.

2.8.5.3. Percent Matching

From a programmers point of view the percentage matching is as well an extention to the complete closed contours matching as to the selected feasts matching. However, from a users point

of view the percentage matching is an alternative to the selected feasts matching. The advantage of selected feast matching that the user can decide which part of the new contour should be used for matching explicitly can sometimes be a disadvantage. Imagine for example an unsupervised recognition task were it is known that a part of the contour can be found again but it is not known which one or where it is located in the image. Here the percentage matching comes into play. However, you need to have a guess of how much — as a percentage — can probably be found or to be more precise how many neighboring feasts should take part in the matching procedure. For a complete contour recognition task with equally spaced feast indices the percent of the contour and the percentage of the feast correspond with each other. If the user has chosen other feast indices probably not equally spaced there will be a difference as the matching procedure really focuses on the percent value of the number of feasts covered and not directly on the percent value of the part of the contour that is covered. Once you have the percent estimate you can call the above mentioned algorithms with this percentage value and the algorithm will find the best part of the contour for each matching task taking only the given percent value of chosen neighboring feasts into account in the final similarity computation. Of course even if the estimated percentage value of how much of the contour can be found is correct one can not expect a recognition rate of 100% if for example two superimposed objects are segmented as one object with only one contour and the database is sufficiently large, see section 3.2.4 for an example. This is due to the fact that the remaining contours parts are not always statistically significant for object recognition. However in a lot of cases where the algorithm will still find the right object even humans would fail if the had only the combined contour as input for their decision making.

3. Results of the Contour Recognition Tasks

You can know the name of a bird in all the languages of the world, but when you're finished, you'll know absolutely nothing whatever about the bird... So let's look at the bird and see what it's doing – that's what counts. I learned very early the difference between knowing the name of something and knowing something.

Richard Feynman (1918 - 1988)

3.1. The Contour Model Applied to the Recognition of Complete Closed Contours

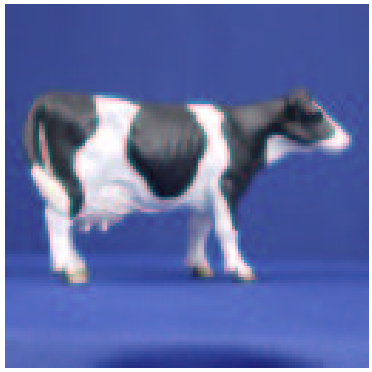
3.1.1. Illustration of the Gallery Images Used

Two benchmark databases well known in the literature were used to compute the results presented below. The SQUID database (Mokhtarian and Kittler, 1996a; Mokhtarian and Kittler, 1996b) consists of 1100 views on marine animals and is illustrated in figure 3.2. This database was mainly chosen because of its huge size, so that statistical evaluations of recognition rates are meaningful. As the object contour parameterizations were already provided several affine transformations of each contour could be computed without much effort. Notice however, that the parameterizations were not used directly. Instead silhouette images were recomputed from the original and transformed parameterizations so that the same preprocessing steps necessary for newly recorded camera images could be applied. The other benchmark database used was 'ETH80' illustrated in figure 3.1. It is provided by Professor Bernt Schiele (Leibe and Schiele, 2003) at ETH Zürich. This database contains only eight object classes like cars, cows, horses or dogs. Each class has approximately ten different representatives. For each representative a lot of views from different 3D viewpoints are provided.

3.1.2. Classification of Similar Contours to Form Object Clusters

For large databases — like the SQUID database — one important question arises from a computational point of view. Can the contents of the database be clustered? This is an important question because if the contents can be clustered hierarchical recognition algorithms can possibly be applied for specific tasks. The result will be either a gain in recognition speed or the possibility to handle even more data. Of course, clusters generated by choosing example representatives are by no means disjoint. Therefore the user has to carefully select typical examples for the clusters he wants to generate. Examples for clusters are given in figure 3.3. Here one can see in figure 3.3 a group of similar looking sharks. It is not possible to give a general threshold for arbitrary clustering task, as the threshold is dependent on the aims of the user and on the specific features of the objects in question. However, similarities below 90% do not seem to justify an addition to a cluster of the contour in question. For a lot of experiments a reasonable threshold seems to be 93% to be sure that the object in question belongs to the cluster. For the range between 90% and 93% no general rules can be given and it is suggested that another matching step is done focussing on the needs of the specific user task. If another task dependend matching step is not a valid choice then one can use 92% as a single threshold.

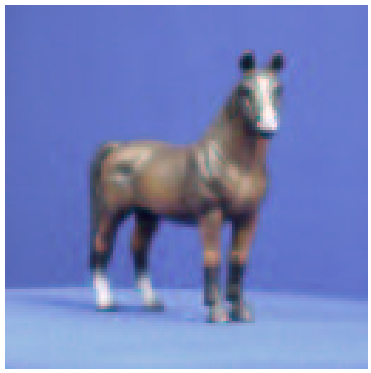
(a) A car.



(b) A cow.

(c) A pear.

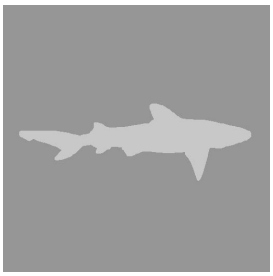
(d) A dog.



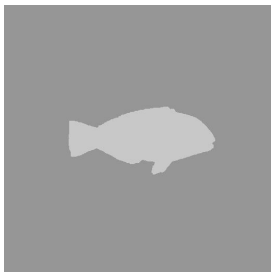
(e) A horse.

(f) A tomato.

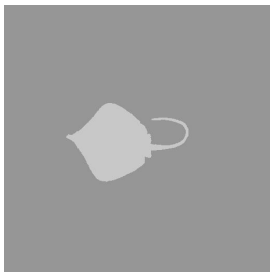
Figure 3.1.: (a) -(f) are example images of the ETH80 database used for object recognition tasks presented in later sections.



(a) A shark.

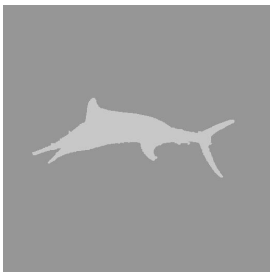


(b) A fish of unknown type.



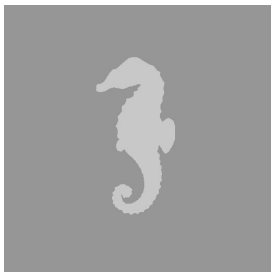
(c) A skate.

(d) A fish of unknown type.

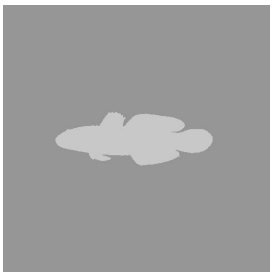


(e) A fish of unknown type.

(f) A fish of unknown type.

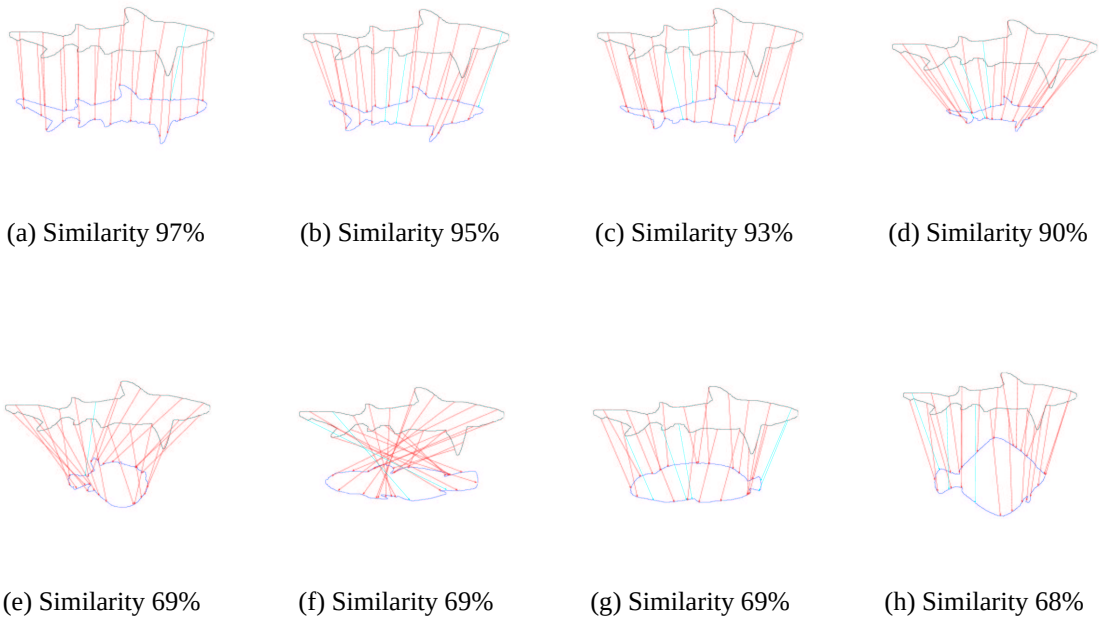


(g) A sea horse.



(h) A fish of unknown type.

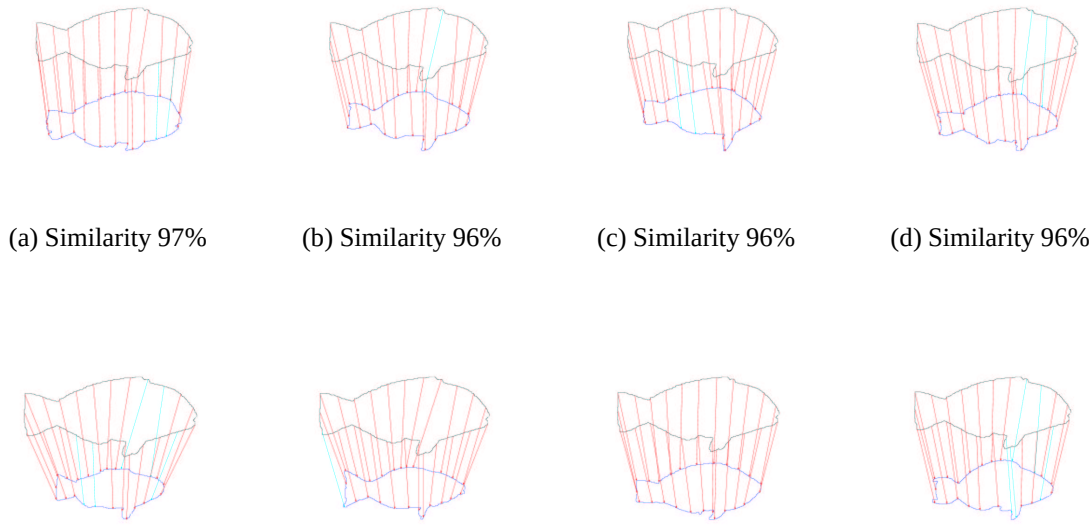
Figure 3.2.: (a) -(h) are example images taken from the SQUID database used for object recognition tasks presented in later sections.



(a) Similarity 97% (b) Similarity 95% (c) Similarity 93% (d) Similarity 90%

(e) Similarity 69% (f) Similarity 69% (g) Similarity 69% (h) Similarity 68%

Figure 3.3.: Classification of similar contours: The 'shark' group as an example for clustering object contour data. (a) - (d) The four best matching not identical objects in the SQUID database. The reference contour used for generating this cluster is shown at the top of the figures. Red arrows indicate similarities of 90% and above. To be more precise only a cluster of similar looking marine animals was created that all have a similar posture. Furthermore the cluster does not cover all marine animals that are in zoological terms classified as sharks even if they have the right posture, as there may be sharks that have a totally different looking silhouette. To create such a zoological cluster a group of examples for all 'exceptions' would have to be used as a gallery, rather that using only one contour for comparison. (e) - (h) Examples to illustrate the degree of dissimilarity of bad matching objects to the chosen shark contour. The results were obtained by matching complete contours that were subdivided into 24 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. Each contour was interpolated by 1536 number of points. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.



(a) Similarity 97% (b) Similarity 96% (c) Similarity 96% (d) Similarity 96%

(e) Similarity 95% (f) Similarity 95% (g) Similarity 95% (h) Similarity 95%

Figure 3.4.: Classification of similar contours: Another example for clustering object contour data. (a) - (h) The best matching eight not identical objects in the SQUID database. The reference contour used for generating this cluster is shown at the top of the figures. Red arrows indicate similarities of 90% and above. As the chosen contour is very typical for marine animals the found cluster size increases substantially. Not even all good matches are shown. The results were obtained by matching complete contours that were subdivided into 24 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. Each contour was interpolated by 1536 number of points. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.

(a) Similarity 95% (b) Similarity 94% (c) Similarity 94% (d) Similarity 93%

(e) Similarity 93% (f) Similarity 93% (g) Similarity 73% (h) Similarity 73%

(i) Similarity 72% (j) Similarity 72%

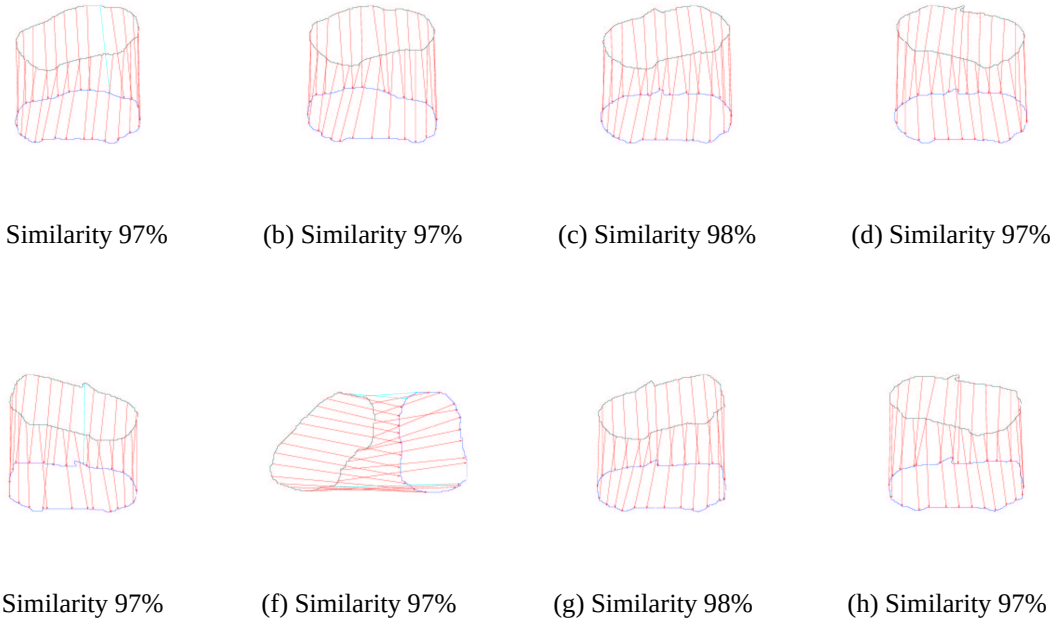
Figure 3.5.: Classification of similar contours: The 'skate' group as an example for clustering object contour data. (a) - (f) The best matching not identical objects in the SQUID database. Red arrows indicate similarities of 90% and above. (g) - (j) Examples of bad matching results. The results were obtained by matching complete contours that were subdivided into 24 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. Each contour was interpolated by 1536 number of points. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.

3.1.3. Complete Contour Matching of 3D-Postures of Different Individual Representatives of the Same Object Class

The results shown in figure 3.6 were created by using the ETH80 database. For objects of the class of automobiles each set of images showing one specific type was split in two parts in a way that views showing slightly different postures were separated. One part was used for the creation of a gallery and the other part is used to be matched against that gallery. In effect each of those two sets never contained the same 3D posture. The matching was done to test the robustness of the algorithm to rotations in three dimensions. For a rigid object like an automobile the results are very good and even a reasonable solution for the problem of corresponding points is found.

The results shown in figures 3.7, 3.8, 3.9, 3.10, and in appendix A.1 were generated using ten different representatives of each object class taken from the ETH80 database. The complete set of images for three representatives each was used to build an object contour gallery. The other representatives of the same class not already in the gallery were subsequently matched against the created gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated with relatively high accuracy.

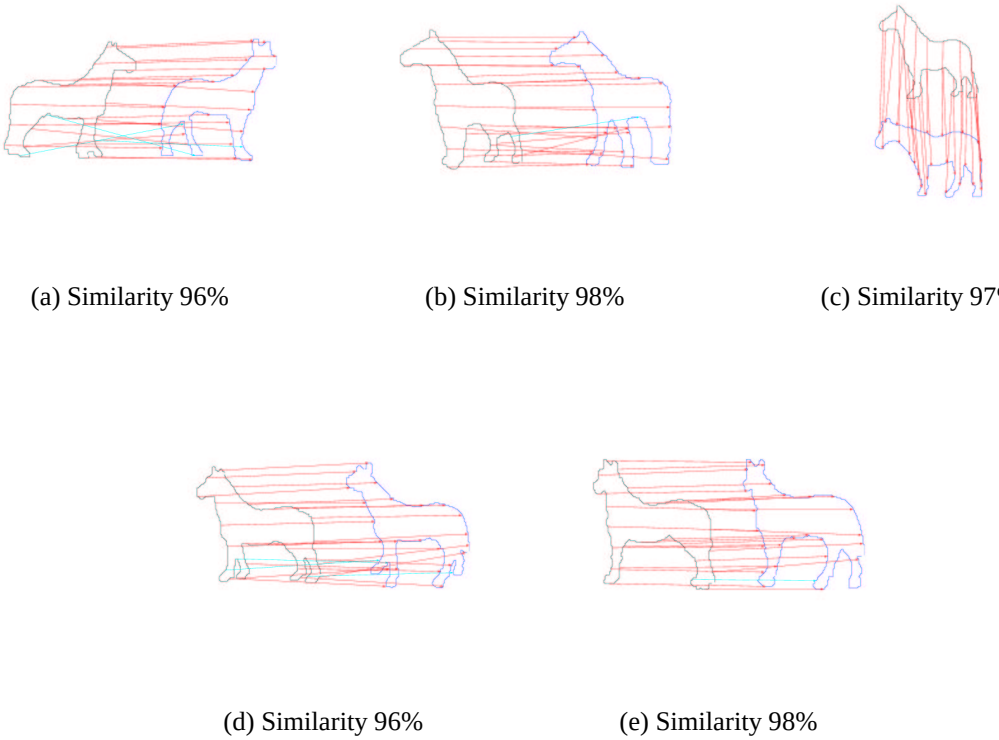
Notice that in figure 3.7 (d) a slightly different posture was found than the object in the original image had. The same effect can be found in figure 3.8 (a) and figure 3.10 (e) and (f). For articulated objects this is an inherent feature of the matching procedure if the exact posture is not already part of the gallery. Therefore, a view from a slightly different perspective may be more similar that the view from the exact 3D posture if for example some extremities have a different configuration. Taking into account the effect shown in figure 3.6 that the matching process can tolerate slight 3D rotations this is a wanted feature. If exact postures should be estimated then they have to be part of the gallery.



(a) Similarity 97% (b) Similarity 97% (c) Similarity 98% (d) Similarity 97%

(e) Similarity 97% (f) Similarity 97% (g) Similarity 98% (h) Similarity 97%

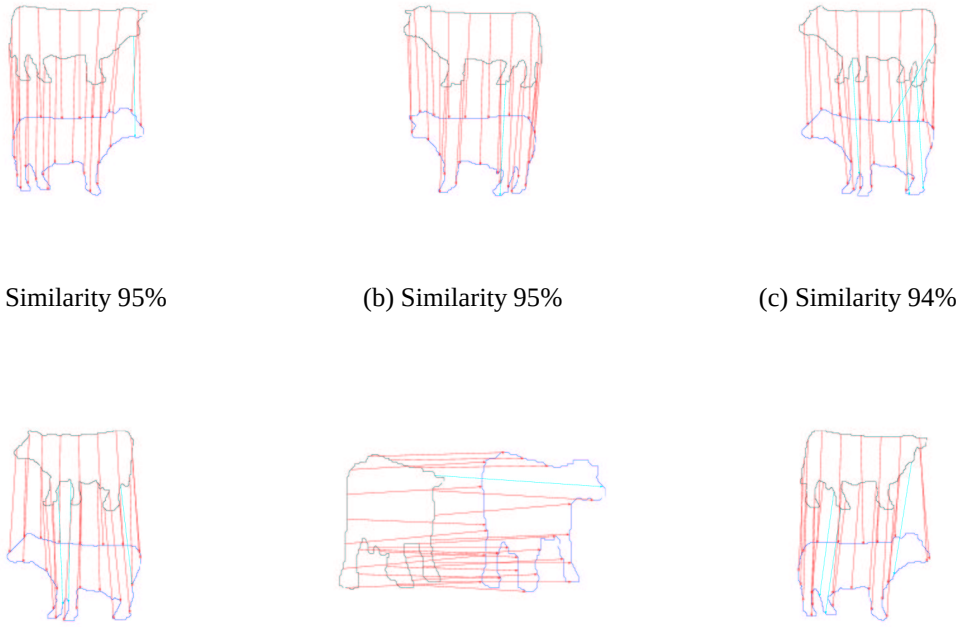
Figure 3.6.: Example for matching different three dimensional postures. The reference contour used for generating these results is shown at the top of the figures. Red arrows indicate similarities of 90% and above. The car matching example is given to illustrate the ability of the algorithm to tolerate rotations in three dimensions. Each individual automobile dataset from the ETH80 database was split in two disjunct parts, so that the same posture was not present in both datasets. It can be seen that postures with small changes of the rotation angles are found and that The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.



(a) Similarity 96% (b) Similarity 98% (c) Similarity 97%

(d) Similarity 96% (e) Similarity 98%

Figure 3.7.: Example for matching different three dimensional postures. The reference contour used for generating these results is shown at the top of the figures. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for three representatives was used to build a gallery for horse silhouettes. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated. Notice that in (d) a posture was found that is slightly different to the posture the object in the original image had. For articulated objects this is an inherent feature of the matching procedure if the exact posture is not part of the gallery already and therefore a view from a slightly different perspective may be more similar that the exact 3D posture. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.



(a) Similarity 95% (b) Similarity 95% (c) Similarity 94%

(d) Similarity 96% (e) Similarity 92% (f) Similarity 93%

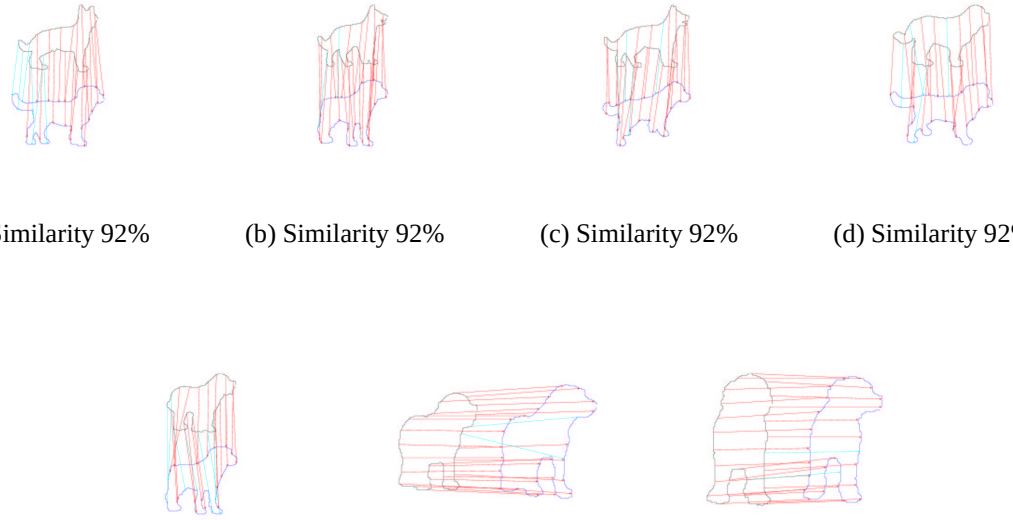
Figure 3.8.: Example for matching different three dimensional postures of cows. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for three representatives was used to build an object contour gallery. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated with relatively high accuracy. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were $\sigma = 3.1414$ and $f = 1.18925$. f is the factor between two Gabor levels in the transformation.

(a) Similarity 96% (b) Similarity 93% (c) Similarity 94% (d) Similarity 96%

(e) Similarity 96% (f) Similarity 94% (g) Similarity 97% (h) Similarity 94%

(i) Similarity 93% (j) Similarity 91% (k) Similarity 96%

Figure 3.9.: Example for matching different three dimensional postures of pears. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for one representative was used to build an object contour gallery. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well. In this case a highly unstructured contour like that of a pear was matched. The results show that even large variations in the shape of the pear can be matched fairly well if the reference contour is chosen well. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.



(a) Similarity 92% (b) Similarity 92% (c) Similarity 92% (d) Similarity 92%

(e) Similarity 92% (f) Similarity 97% (g) Similarity 95%

Figure 3.10.: Example for matching different three dimensional postures of dogs. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for three representatives was used to build an object contour gallery. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated with relatively high accuracy. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were $\sigma = 3.1414$ and $f = 1.18925$. f is the factor between two Gabor levels in the transformation.

3.1.4. Matching Normalized Contours

3.1.4.1. Matching of Contours Rotated in the Plane

An object contour has been extracted from the input data and this contour should afterwards be used to recognize a particular object out of a gallery database. The question arises to what extent can this contour be normalized? For normalization to affine transformations in the image plane the first prerequisite for the presented normalization methods is that the contour is closed. Information about the openness of the contour can either be supplied by the user or a heuristic can be used to check this condition automatically. In case it is possible to assume that a complete contour of an object is present and has only undergone affine transformations the presented normalization methods can be applied.

Three separate tasks are set up to examine the effect of the normalization methods: scaled objects, rotated objects and objects that are scaled and rotated. The recognition rate for these tasks using the SQUID database to be able to do large scale tests comes close to 100% and sometimes even reaches that number.

In figure 3.11 (a) an example picture of a rotated and scaled set of images is shown that was used in a matching task. In figure 3.11 (c) the particular matching result for this image against a normalized version of the whole SQUID contour database is shown. Notice that the point to point correspondences a fairly close to the optimal and the overall similarity is 97%. This percentage value is simply the similarity measure given in earlier chapters expressed as a percentage.

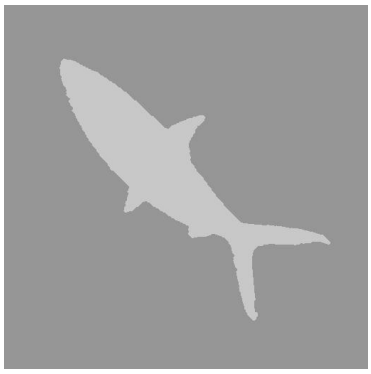
In figure 3.11 (b) statistics is done on the outcome of all matching results like in (c). Imagine a match was done for all contours of the gallery and therefore for each gallery contour a similarity has been computed. If the gallery objects are now sorted for a decrease of this computed similarity one can compute the position at which the right match has occurred. If this is the first position one can speak of a successful match. The amount of occurrences of this phenomenon divided by the total number of individual contour pair matches is called the recognition rate. It is difficult to reach a recognition rate of 100% using the SQUID database as this database contains some objects several times in different scales or rotations.

Even more useful information is obtained analysing the number of all instances, where the correct object occured at recognition position n or less. This accumulated sum of all right matches is plotted against the recognition position and the resulting plot is called *accumulated right match plot* which is shown, for example, in figure 3.11 (b). While the recognition rate is very important for unsupervised recognition tasks the accumulated right match plot gives important information whether the recognition output is useful as a pre-screening tool for humans.

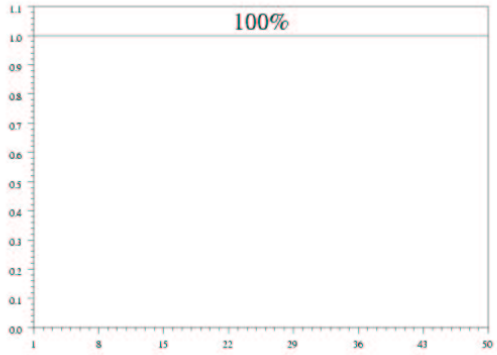
3.1.5. Matching Non-Normalized Contours

3.1.5.1. Matching of Non-Normalized Contours Rotated in the Plane

To test which rotations steps can be tolerated by the feature set an examination of the recognition rate on rotated contours was done when no rotation normalization and no rotation matching have been applied. While the recognition rate illustrated in figure 3.14 and figure 3.13 drops off on average linearly by two percentage points per degree of rotation the reduction is less step for small angles. For rotations of less than 10 degrees the reduction is smaller as still



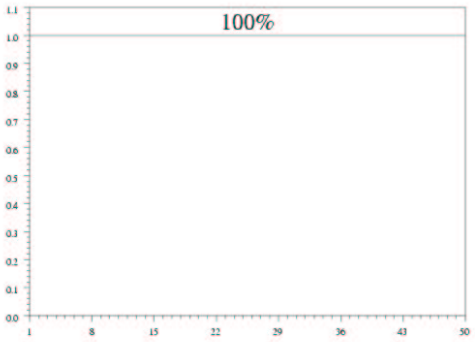
(a) Rotated and Scaled Silhouette



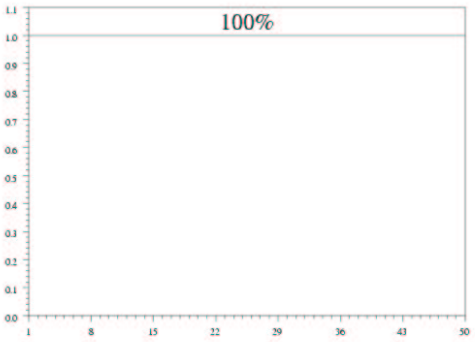
(b) Accumulated Right Match Positions

(c) Example Matching Result

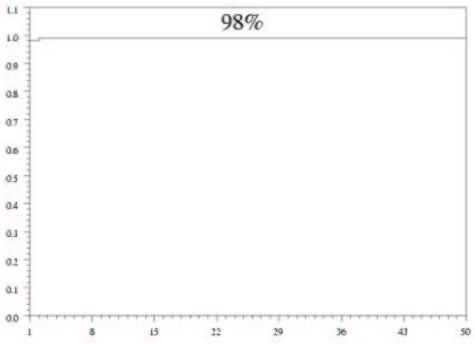
Figure 3.11.: The match result was obtained by matching the complete contour of the shown objects. The number of interpolation points, e.g. indices of feasts, was set to the user defined value of 1536 which has implicitly the function of a scale normalization. The total number of indices divided by the user given constant of 25 gives the number of feasts positions to match. A rotation normalization as described in section 2.6 was applied to the new contours and the gallery contours used. As the number of interpolation points was fixed for all contours recall from section 2.8.2 that only one scale is defined and this scale zero is used in the matching algorithm. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.414 were f is the factor between two Gabor levels in the transformation. This is an example of a match cross run using 100 marine animal silhouettes that were rotated and scaled — example in a) — against a database of 1100. b) The recognition rate is 100%. This type of diagram is called 'Accumulated Right Match Positions'. It is explained in detail in section 3.1.4.1 and is used often in this work to provide more information about the recognition process than only the recognition rate would give in case 100% are not reached. c) Example matching result with the best point to point matches found shown as links. Red indicates a similarity above 90%. The corresponding points were found with high accuracy.



(a) rotated contours

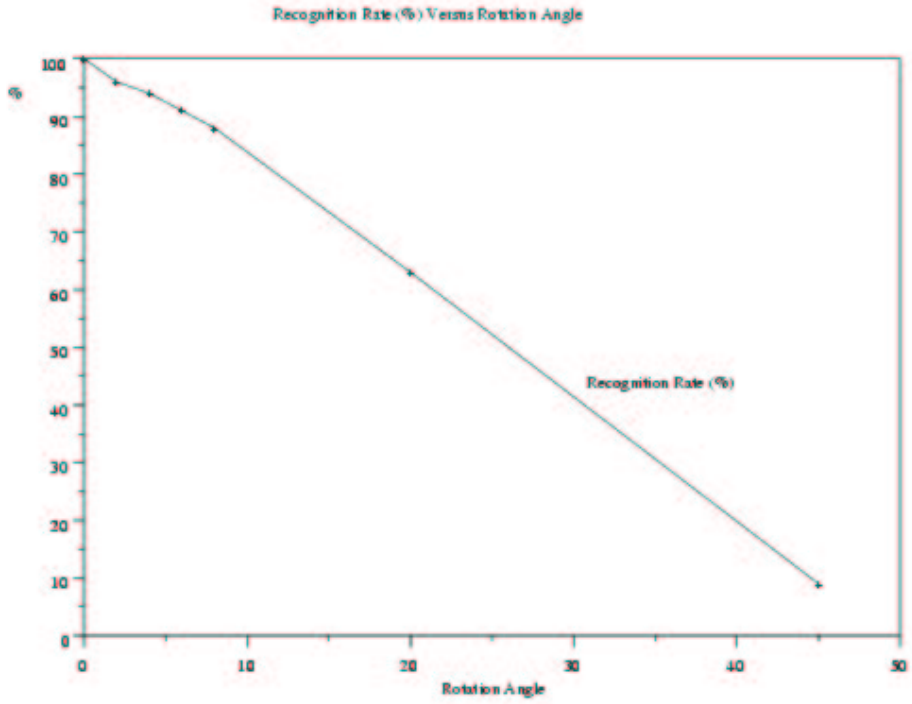


(b) rotated and scaled contours



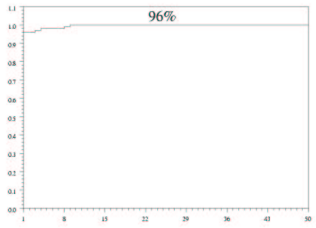
(c) scaled contours

Figure 3.12.: Accumulated right match positions and recognition rates for the tasks of rotated and scaled contours using the normalizations for scale and rotational invariance described in section 2.6. Different samples of 100 transformed contours were match against the 1100 marine animal contours from the SQUID database. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.

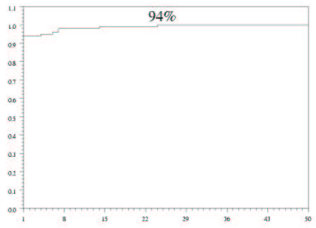


(a) Dependency of the recognition rate (%) on the rotation degree

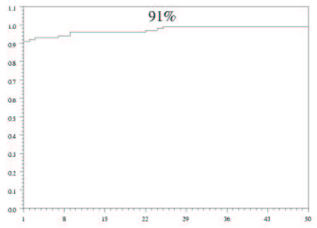
Figure 3.13.: Shown is the dependency of the recognition rate in case complete contours rotated in the plane are matched but no rotation normalization has been done and no rotation move either.



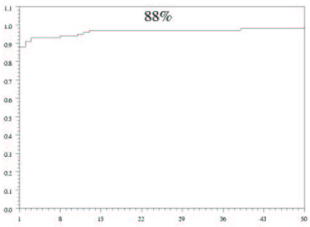
(a) Rotation angle 2 degrees.



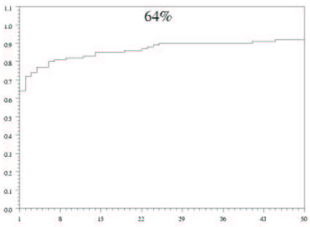
(b) Rotation angle 4 degrees.



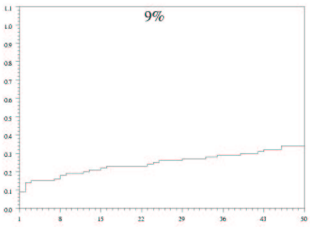
(c) Rotation angle 6 degrees.



(d) Rotation angle 8 degrees.



(e) Rotation angle 20 degrees



(f) Rotation angle 45 degrees

Figure 3.14.: Accumulated right match position for different rotation angles of the objects when no rotation normalization was applied. The rate of successful matches strongly decreases if the rotation angle is increased beyond 10 degree. The results were obtained by matching complete contours that were subdivided into 24 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.

approximately 88% recognition rate can be achieved. Even more important is the fact that within the first fifty match positions shown in figure 3.14 the recognition rate still reaches 100%. The accumulated right match positions plot was explained in detail in section 3.1.4.1. For rotations of 20 degrees the recognition rate is already reduced substantially to 64% and drops down to 9% for 45 degrees. In both cases a recognition rate near 100% is not reached within the first 50 match position indicating that no meaningful matching result was computed.

3.1.5.2. Matching of Non-Normalized Scaled Contours

The matching of normalized scaled contours can easily be done by fixating the number of interpolation points used. The recognition rate is of course near 100% and therefore results for this task are not shown explicitly. Instead the focus of attention is turned to the task of matching non-normalized contours that differ by a scale factor. The results are shown in figure 3.15 and show that for complete closed contours it is sufficient to use a value of $\sqrt[4]{2}$ for the parameter f which determines the relation of successive Gabor levels. The computed recognition rates have a low of 0.93 which is a good result.

(a) scale factor 1.02 (b) scale factor 1.04 (c) scale factor 1.06 (d) scale factor 1.08

(e) scale factor 1.10 (f) scale factor 1.12 (g) scale factor 1.14 (h) scale factor 1.18

(i) scale factor 1.22 (j) scale factor 1.2

Figure 3.15.: Illustration of scale invariant matching if no scale normalization was done. Used for matching were images showing a scaled object. The accumulated right match positions are shown for the various scale factors. The results were obtained by matching complete contours that were subdivided into 24 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were $\sigma = 3.1414$ and $f = 1.18925$. f is the factor between two Gabor levels in the transformation.

3.2. The Contour Model Applied to the Recognition of Contour Parts

3.2.1. Recognition of Known Specific Object Parts

3.2.1.1. Solving the Problem of Multiple Occurrences of the Same Gesture

The hand gestures in figure 3.2.1.2 and figure 3.17 where recognized by giving the algorithms once a hint supplied by the user to use a point of the contour most to the left of the contour. At this point selected feasts matching was done. This is probably not the best method for this application task and there were only 12 different hand gestures in the database. Nevertheless, these experiments are a proof of concept that hand gesture recognition is possible with the presented contour recognition algorithms. The problem of recognizing equivalent gestures done by different people and recorded at different dates under different conditions is a hard one due to the natural variations from time to time and from individual to individual. The results presented show a remarkable ability of the presented algorithm to cope with variations in these hand gestures.

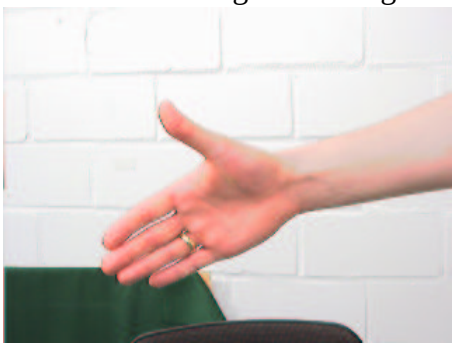
3.2.1.2. Solving the Correspondence Problem for Stereo Images

Even for stereo image pairs the correspondence problem can be solve provided the object contours have a high enough degree of specificity. Of course if one matches too ellipsoidal contours on another the target points are not specific enough in all cases. Fortunately, object contours for real life objects of interest in an application task have a high enough degree of specificity in most of the cases. If one encounters a problem of missing specificity of the object contour the only choice is to focus on object features within the object of interest. If these features are as well not highly specific then the task of identifying individual objects of the same object class is probably not solvable by humans either. Just think of a task like identifying individual eggs.

The remarkable ability of humans to increase their discriminating power by learning the importance of smaller details can be modeled in the proposed model by changing parameters affecting the number of Gabor levels to be computed. However general rules or automatic procedures controlling the decisions on when to change these parameters are outside the scope of this work as they require a complex interactive controlling process. Note that some results for the hand gesture recognition task suggest, for example figure 3.17 (f), the need to define individual graphs for each object or, in this case, hand gesture. These graphs should represent a collection of the important points of the object to recognize for a given task. The development of such a graph representation is not a major problem but is currently not implemented and is therefore subject to future research. In figure 3.17 (f) the result is perfectly right even though a different gesture has been found. The reason for this is that the points to use for matching provided by the automatic matching procedure for selected feasts did not select enough points belonging to the upper part of the hand gesture and therefore differences there were ignored. The parts that were examined match perfectly well. If the selected points were the right ones the results indicate a remarkable ability of the algorithm to deal with slight changes in the overall gesture.

For the distinguishing part of the human profile face the person identifying and correspond-

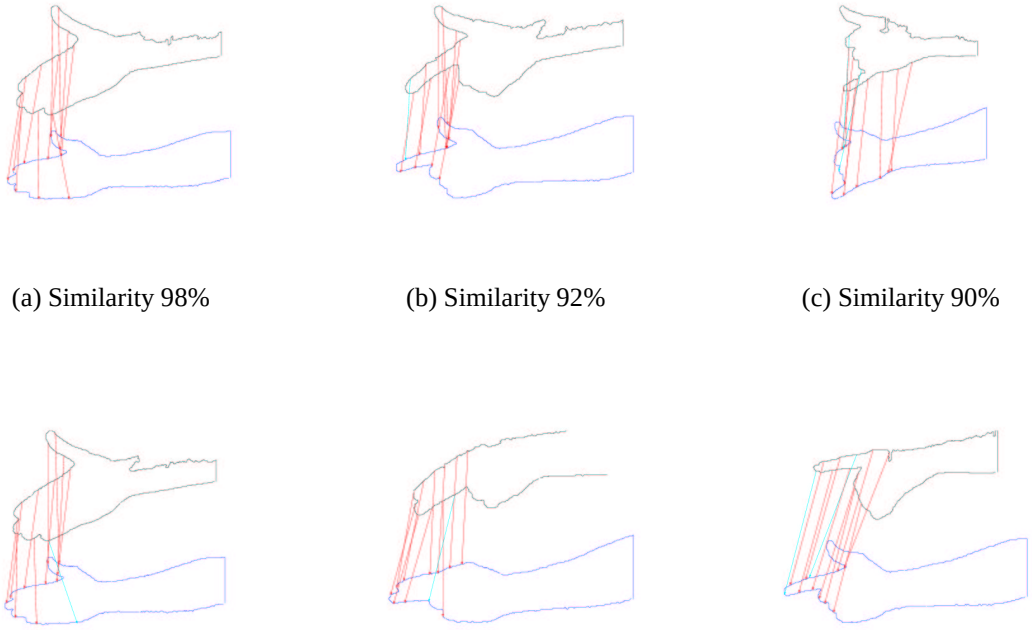
The new hand gesture image



Matching result for a hand gesture

Figure 3.16.: Illustration of the hand gesture recognition task. The number of interpolation points, e.g. indices of feasts computed for this contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from. The match result was obtained by matching selected feasts. The point in the new image for the base feast of the matching was given by the user in so far as it is the point of the contour closest to the user given image pixel [300,140]. The index of that corresponding point is called base index. 10 additional feasts (five to the left and five to the right) were used for matching around the base index and distance to the base index is a multiple of the total number of indices divided by the user given constant of 50. The matching algorithm itself was done on all possible scales if the sizes of the two contours was different. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.414 were f is the factor between two Gabor levels in the transformation.

ing points problem are easily solved in figure 3.18 and figure 3.19. Even the slight 3D rotation of the faces and the different contour sizes caused by the different zooming factors can be compensated. Notice that no normalizations were done so that the whole result is an effect of the matching algorithm. Furthermore it should be possible to solve the correspondence problem for arbitrary images. If it is possible to find a large enough edge in both images using edge detectors and edge continuity algorithms and these edges have somewhat more structure than a straight line or an arc the matching procedures that were presented in the earlier chapters can be applied.

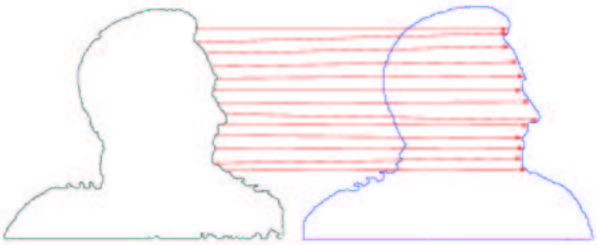


(a) Similarity 98% (b) Similarity 92% (c) Similarity 90%

(d) Similarity 96% (e) Similarity 94% (f) Similarity 92%

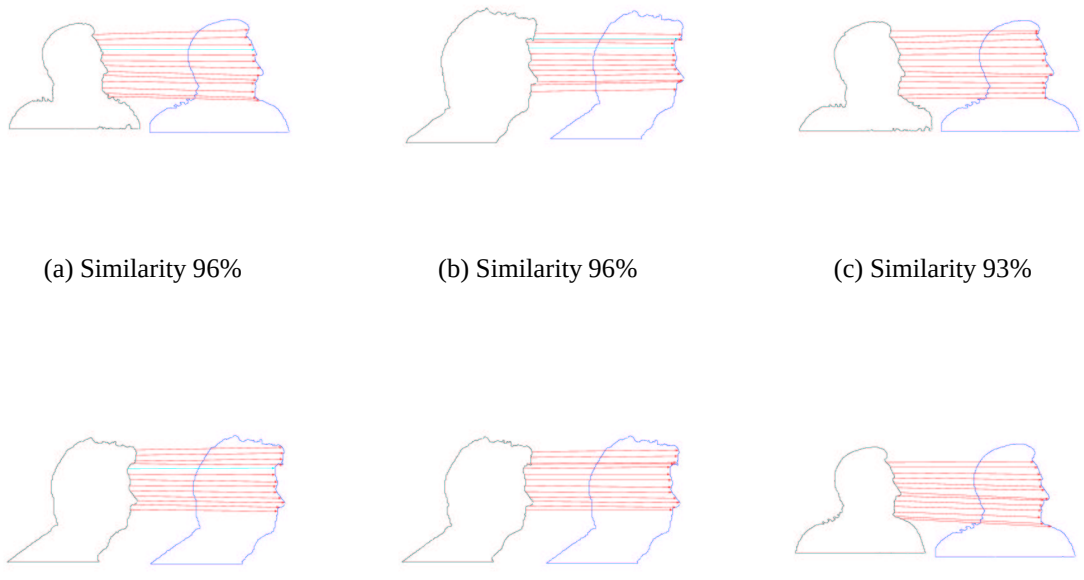
Figure 3.17.: Illustration of the hand gesture recognition problem. Red arrows indicate similarities of 90% and above. The point in the new image for the base feast of the matching was given by the user in so far as it is the point of the contour closest to the user given image pixel [0,120]. A set of twelve different hand gestures performed by two individuals were recorded with a stereo camera pair using different zooming factors. Notice that no normalizations were done so that the whole result is an effect of the matching algorithm. Six typical matching results are shown. The results can probably be improved further by defining graphs for each hand gesture that capture the important points. The results were obtained by matching selected points of the contours by using 11 feasts including and around each selected point. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.41425. f is the factor between two Gabor levels in the transformation.

(a) The profile face image



(b) Matching result for the profile face

Figure 3.18.: Illustration of profile face matching. The number of interpolation points, e.g. indices of feasts computed for this contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from. The match result was obtained by matching selected feasts. The point in the new image for the base feast of the matching was given by the user in so far as it is the point of the contour closest to the user given image pixel [250,110]. The index of that corresponding point is called base index. 10 additional feasts (five to the left and five to the right) were used for matching around the base index and distance to the base index is a multiple of the total number of indices divided by the user given constant of 50. The matching algorithm itself was done on all possible scales if the sizes of the two contours was different. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.414 were f is the factor between two Gabor levels in the transformation.



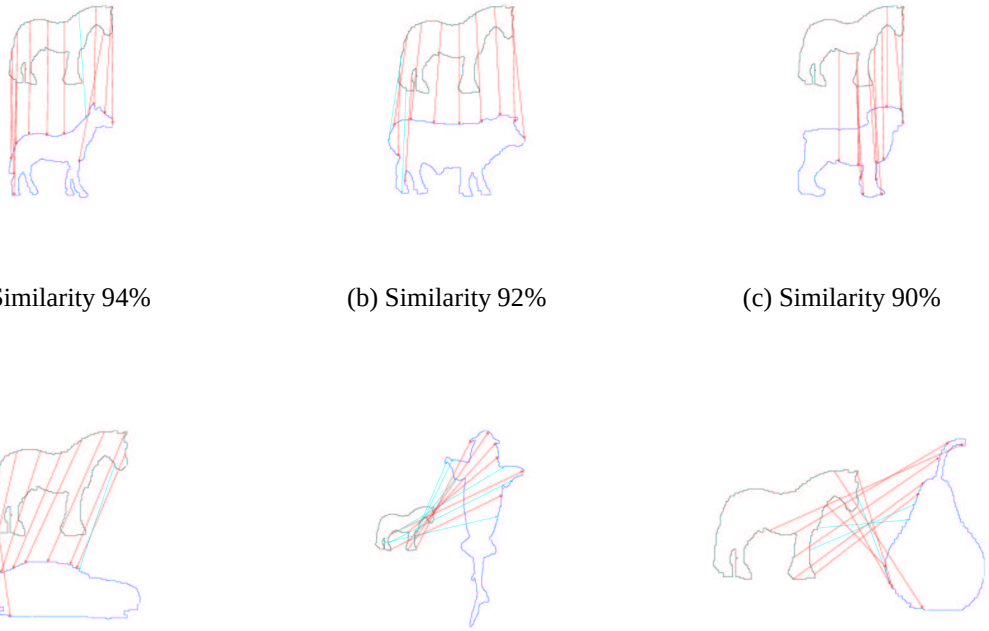
(a) Similarity 96% (b) Similarity 96% (c) Similarity 93%

(d) Similarity 96% (e) Similarity 99% (f) Similarity 99%

Figure 3.19.: Illustration of how the stereo correspondence problem can be solved. Red arrows indicate similarities of 90% and above. For two individuals image sequences were recorded with a stereo camera pair using different zooming factors. For the distinguishing part of the human profile face the corresponding points problem can be solved easily. Even the slight 3D rotation of the faces and the different contour sizes caused by the different zooming factor can be compensated. Notice that no normalizations were done so that the whole result is an effect of the matching algorithm. The results were obtained by matching selected points of the contours by using 13 feasts including and around each selected point. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were $\sigma = 3.1414$ and $f = 1.18925$. f is the factor between two Gabor levels in the transformation.

3.2.2. Recognition of Different Objects

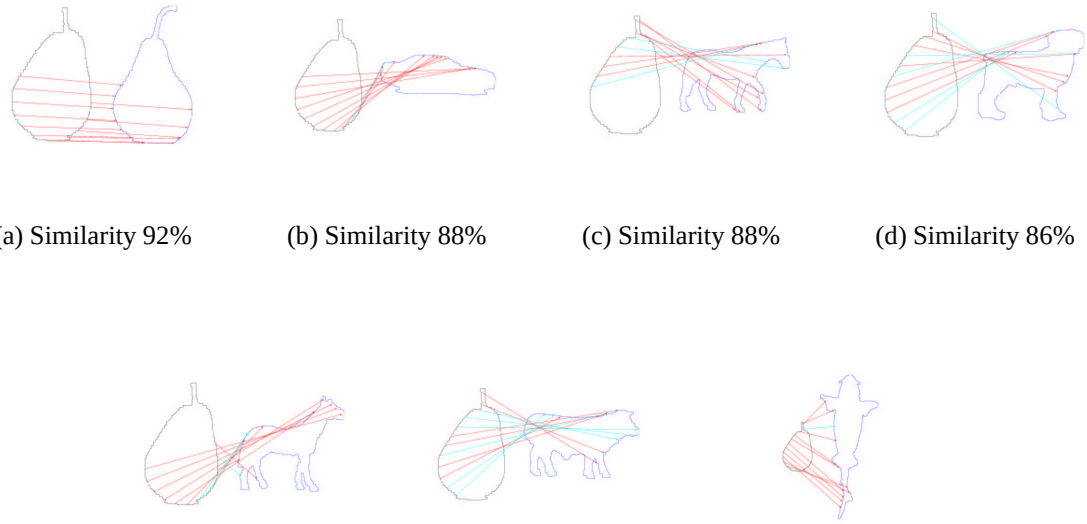
The results shown in figures 3.20, 3.21, 3.22 and 3.23 were generated using one representative of each object class taken from the ETH80 and SQUID databases to create a gallery of contours to match against. Other images of different representative of each object class contained in the ETH80 database were then matched against this gallery. The task was to recognize the type of object given the posture of the object so only the best matching object is relevant in evaluating the matching success. However, for example for horses and cows the results indicate that the posture plays an important role in the recognition process. Therefore again the need arises to define specific object graphs that capture the important points to look for especially for articulated objects. By this procedure it should be possible to bias the matching process in favor of, for example, differences in the head form of the animals while similar postures of the legs alone should not lead to high similarities. Such an approach is subject to future research.



(a) Similarity 94% (b) Similarity 92% (c) Similarity 90%

(d) Similarity 89% (e) Similarity 84% (f) Similarity 78%

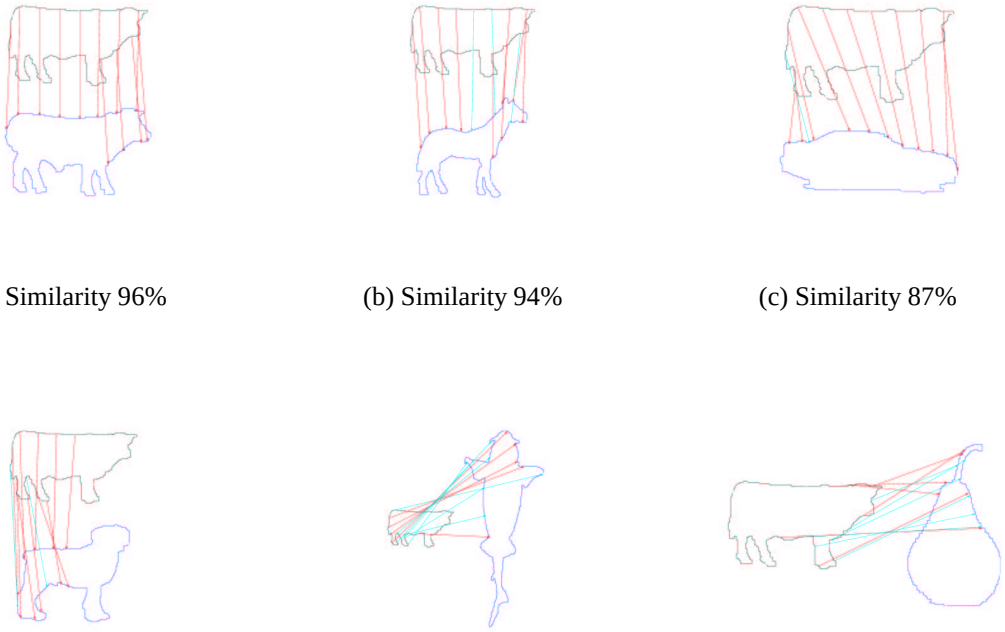
Figure 3.20.: An illustration for the object recognition task is given. A gallery of six different object types was generated. The given contour was matched against all of these object contours. The results show that the right object class had the highest similarity and therefore the object class was identified. Notice that the second best match (b) had as well a relative high similarity. An even better separation can probably be achieved if a specific object graph can be used for matching that emphasizes the object specific details, for example the different head forms of horses and cows. The results were obtained by matching complete contours that were subdivided into 25 interval ranges. The best matching contiguous 50% of these feast , as described in section 2.8.5.3, were used for the similarity computation of the contours. Each contour was interpolated by 1536 number of points. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.



(a) Similarity 92% (b) Similarity 88% (c) Similarity 88% (d) Similarity 86%

(e) Similarity 85% (f) Similarity 84% (g) Similarity 80%

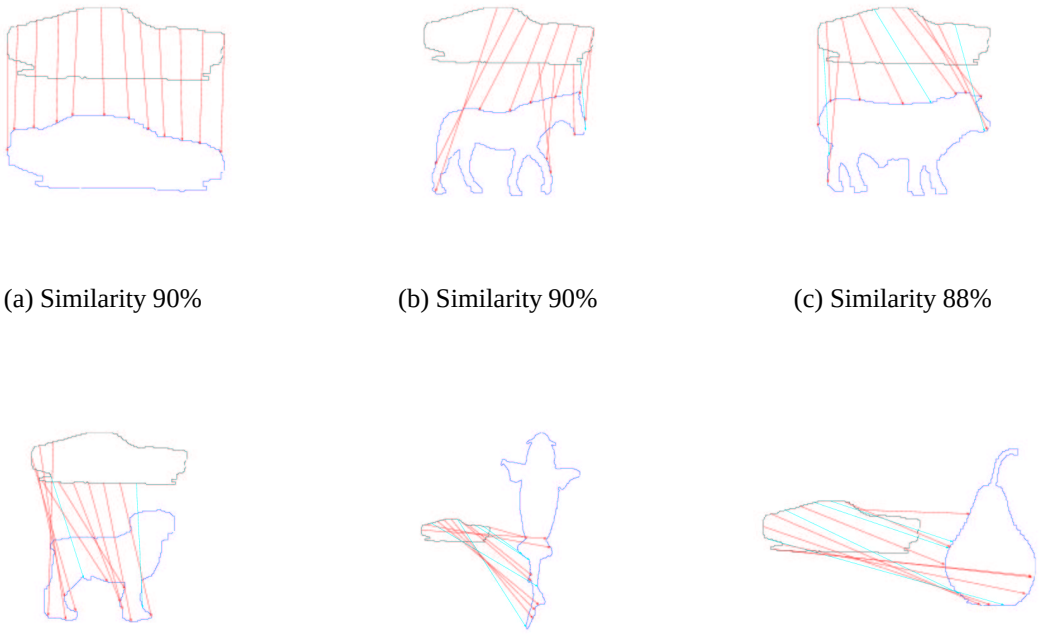
Figure 3.21.: An illustration for the object recognition task is given. A gallery of six different object types was generated. The given contour was matched against all of these object contours. The results show that the right object class had the highest similarity and therefore the object class was identified. The results were obtained by matching complete contours that were subdivided into 25 interval ranges. The best matching contiguous 50% of these feast , as described in section 2.8.5.3, were used for the similarity computation of the contours. Each contour was interpolated by 1536 number of points. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.



(a) Similarity 96% (b) Similarity 94% (c) Similarity 87%

(d) Similarity 83% (e) Similarity 81% (f) Similarity 74%

Figure 3.22.: An illustration for the object recognition task is given. A gallery of six different object types was generated. The given contour was matched against all of these object contours. The results show that the right object class had the highest similarity and therefore the object class was identified. The results were obtained by matching complete contours that were subdivided into 25 interval ranges. The best matching contiguous 50% of these feast , as described in section 2.8.5.3, were used for the similarity computation of the contours. Each contour was interpolated by 1536 number of points. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.



(a) Similarity 90% (b) Similarity 90% (c) Similarity 88%

(d) Similarity 84% (e) Similarity 81% (f) Similarity 73%

Figure 3.23.: An illustration for the object recognition task is given. A gallery of six different object types was generated. The given contour was matched against all of these object contours. The results show that the right object class had the highest similarity and therefore the object class was identified. The results were obtained by matching complete contours that were subdivided into 25 interval ranges. The best matching contiguous 50% of these feast , as described in section 2.8.5.3, were used for the similarity computation of the contours. Each contour was interpolated by 1536 number of points. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.

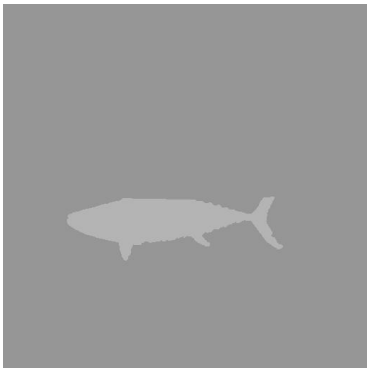
3.2.3. The Occlusion Task

In typical applications for example segmentation tasks the objects to recognize are often occluded to some extent. To find robust algorithms that can identify objects under these conditions is an important goal. One other application is the process of forming a complex segmentation from input of low level segmentation results for example generated by a homogeneous color cue. Most of the times different parts of an object have different colors. If one large enough segment can be found that captures a significant part of the object contour a hypotheses can be generated on what kind of object is present and where to look for the other parts of the object in the image.

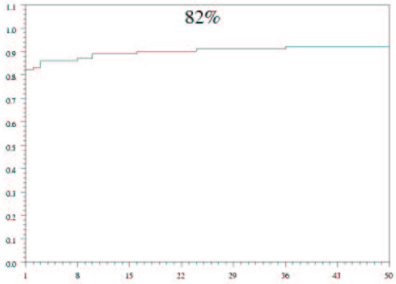
The results for the occlusion task presented in figure 3.24 till figure 3.29 were obtained by matching different percentage amounts of feasts. The results indicate a negative dependency of the recognition rate on the percentage amount of feasts used. The optimal percentage of feasts to use in the experiments shown was 20% which was the lowest number used. However percentages below 10% should not be used (not shown) in any case or to be more precise less that 5 feast to feast matches make the matching results very unreliable.

Recognition rates of approximately 70% can easily be achieved by using approximately 20% of the contour for the matching process. This number is very remarkable as the maximum reachable is unknown. Some objects are so small or have such an unstructured contour that even humans can probably not achieve a recognition rate of 100% for this task.

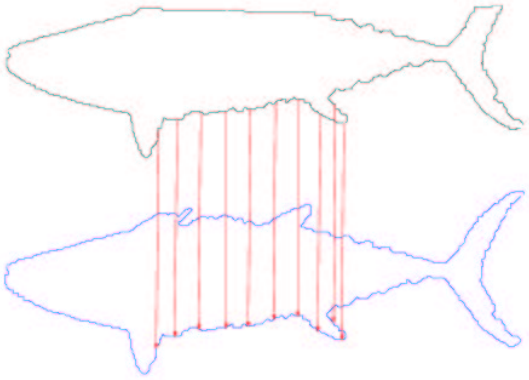
The recognition rate can be made approximately constant by using selected feast matching for the given input data shown in figure 3.30. As the occlusions are always done horizontally matching at the point of the contour that is closest to the middle of the bottom of the image is a useful hint. Although it is of course only applicable to this data set. In addition the results indicate that the choice of an ellipsoid makes the the occlusion task probably more difficult. The reason for this is probably that the smooth borders artificially introduced by the occlusion are matched by accident to other real object contours that do not have a high degree of specificity like arcs or straight lines. The effect gets stronger if the translation between the objects gets smaller and therefore the occluded part of the object gets larger.



(a) Original Image

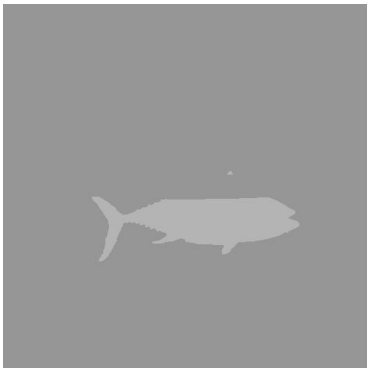


(b) 46 pixels translation

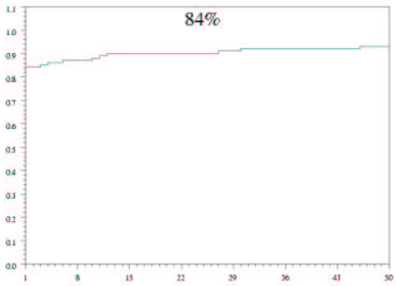


(c) Similarity 100%

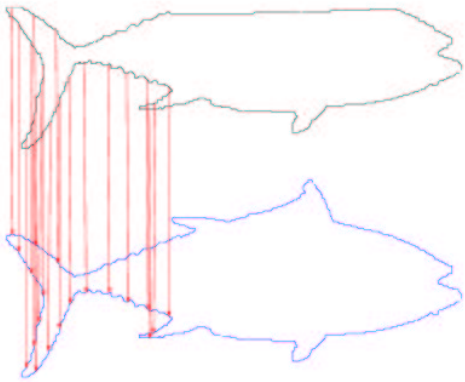
Figure 3.24.: Accumulated right matches for one occluded object recognition task. Used were only a percent amount of all feast to find the best matching object. (a) A new image of the given task occluded by an ellipsoid (not shown) whose center was moved 46 pixels away from the center of the object. (b) Accumulated right match positions. The percentage of right matches on rank one is given explicitly. (c) Example matching result illustrating the point to point match positions for one specific match. The results were obtained by matching complete contours that were subdivided into 50 interval ranges. The best matching contiguous 20% of these feast , as described in section 2.8.5.3, were used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.41425. f is the factor between two Gabor levels in the transformation.



(a) Original Image

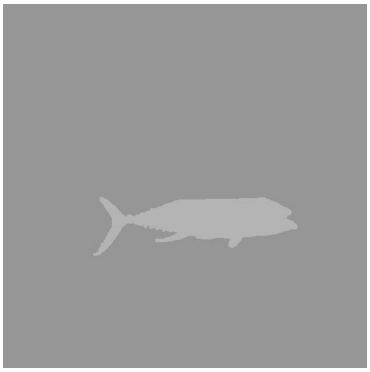


(b) 46 pixels translation

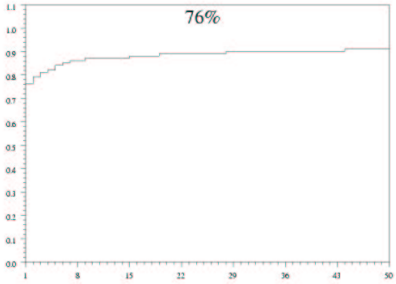


(c) Similarity 100%

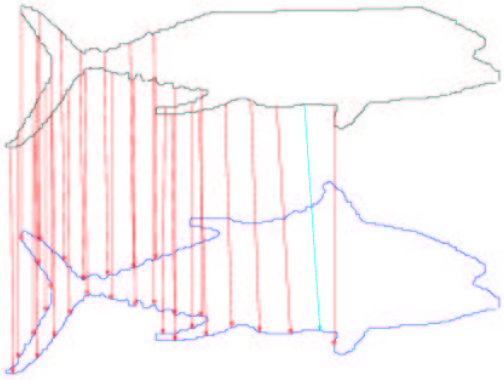
Figure 3.25.: Accumulated right matches for one occluded object recognition task. Used were only a percent amount of all feast to find the best matching object. (a) A new image of the given task occluded by an ellipsoid (not shown) whose center was moved 46 pixels away from the center of the object. (b) Accumulated right match positions. The percentage of right matches on rank one is given explicitly. (c) Example matching result illustrating the point to point match positions for one specific match. The results were obtained by matching complete contours that were subdivided into 50 interval ranges. The best matching contiguous 40% of these feast , as described in section 2.8.5.3, were used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.41425. f is the factor between two Gabor levels in the transformation.



(a) Original Image



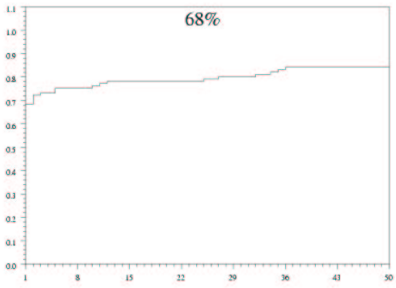
(b) 46 pixels translation



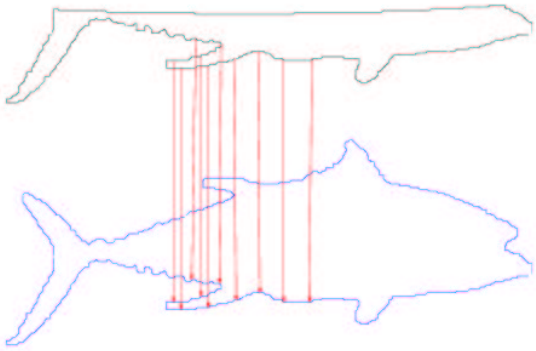
(c) Similarity 97%

Figure 3.26.: Accumulated right matches for one occluded object recognition task. Used were only a percent amount of all feast to find the best matching object. (a) A new image of the given task occluded by an ellipsoid (not shown) whose center was moved 46 pixels away from the center of the object. (b) Accumulated right match positions. The percentage of right matches on rank one is given explicitly. (c) Example matching result illustrating the point to point match positions for one specific match. The results were obtained by matching complete contours that were subdivided into 50 interval ranges. The best matching contiguous 60% of these feast , as described in section 2.8.5.3, were used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.41425. f is the factor between two Gabor levels in the transformation.

(a) Original Image

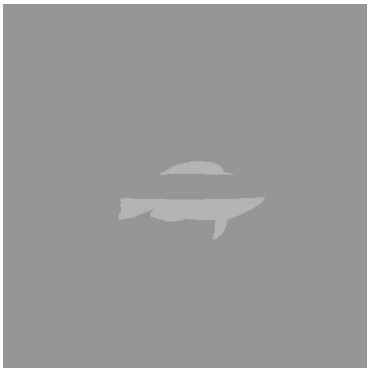


(b) 16 pixels translation

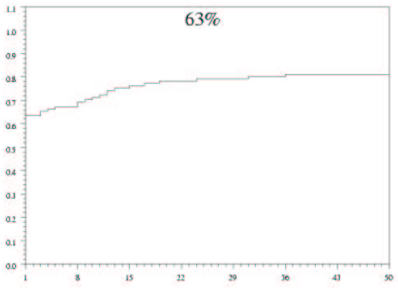


(c) Similarity 100%

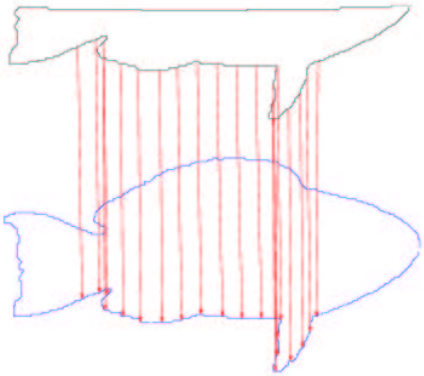
Figure 3.27.: Accumulated right matches for one occluded object recognition task. Used were only a percent amount of all feast to find the best matching object. (a) A new image of the given task occluded by an ellipsoid (not shown) whose center was moved 16 pixels away from the center of the object. (b) Accumulated right match positions. The percentage of right matches on rank one is given explicitly. (c) Example matching result illustrating the point to point match positions for one specific match. The results were obtained by matching complete contours that were subdivided into 50 interval ranges. The best matching contiguous 20% of these feast , as described in section 2.8.5.3, were used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.41425. f is the factor between two Gabor levels in the transformation.



(a) Original Image



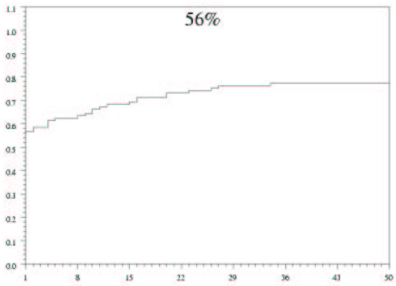
(b) 16 pixels translation



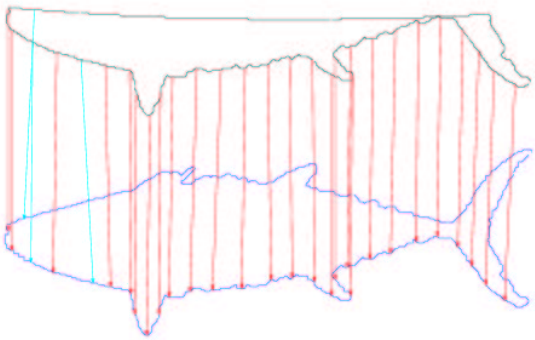
(c) Similarity 100%

Figure 3.28.: Accumulated right matches for one occluded object recognition task. Used were only a percent amount of all feast to find the best matching object. (a) A new image of the given task occluded by an ellipsoid (not shown) whose center was moved 16 pixels away from the center of the object. (b) Accumulated right match positions. The percentage of right matches on rank one is given explicitly. (c) Example matching result illustrating the point to point match positions for one specific match. The results were obtained by matching complete contours that were subdivided into 50 interval ranges. The best matching contiguous 40% of these feast , as described in section 2.8.5.3, were used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.41425. f is the factor between two Gabor levels in the transformation.

(a) Original Image

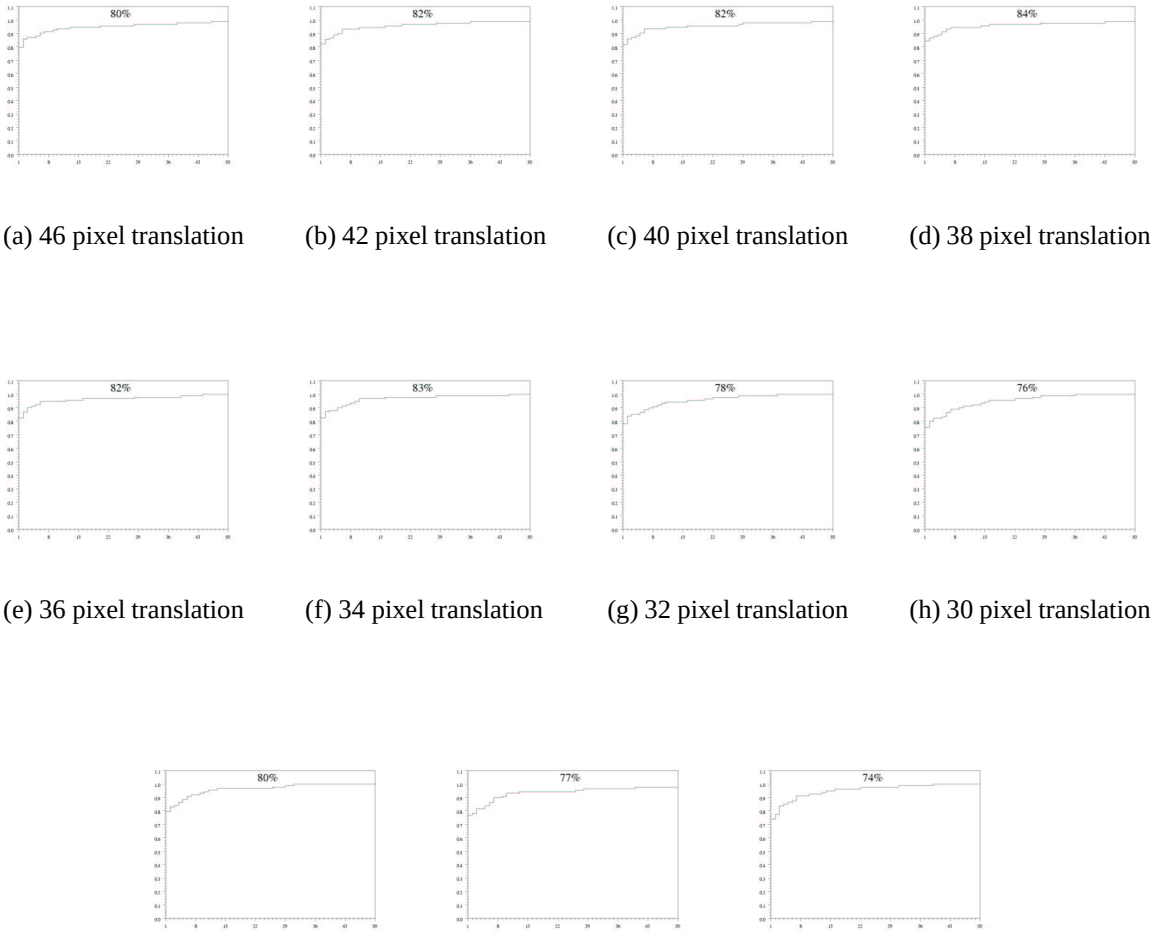


(b) 16 pixels translation



(c) Similarity 96%

Figure 3.29.: Accumulated right matches for one occluded object recognition task. Used was only a small percent amount of all feast to find the best matching object. (a) A new image of the given task occluded by an ellipsoid (not shown) whose center was moved 16 pixels away from the center of the object. (b) Accumulated right match positions. The percentage of right matches on rank one is given explicitly. (c) Example matching result illustrating the point to point match positions for one specific match. The results were obtained by matching complete contours that were subdivided into 50 interval ranges. The best matching contiguous 60% of these feast , as described in section 2.8.5.3, were used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.41425. f is the factor between two Gabor levels in the transformation.



(a) 46 pixel translation (b) 42 pixel translation (c) 40 pixel translation (d) 38 pixel translation

(e) 36 pixel translation (f) 34 pixel translation (g) 32 pixel translation (h) 30 pixel translation

(i) 26 pixel translation (j) 20 pixel translation (k) 16 pixel translation

Figure 3.30.: Selected point matching using the contour points nearest to the middle of the bottom of the image. The matching results are nearly independent of the translation in pixels between the object and the occluding ellipsoid. The results were obtained by matching selected points of the contours by using 9 feasts including and around each selected point. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.41425. f is the factor between two Gabor levels in the transformation.

3.2.4. The Multiple Objects Task

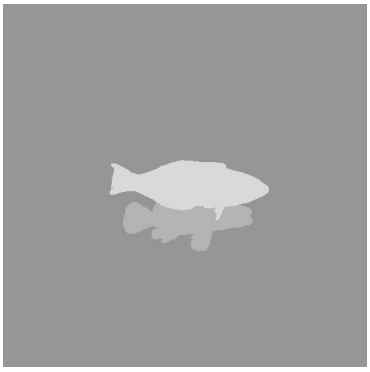
Another important problem in segmentation tasks is for example the occurrence and subsequent segregation of multiple objects in an image. Like in the occlusion task it would be very helpful to be able to build a hypotheses based on a part of an extracted enlarged contour composed of multiple objects.

A reduction of the percentage amount of feasts to be used for matching covering only 30% of the contour leads to a slight reduction in the recognition rates as it is illustrated in figure 3.34. The reason for this effect is probably that the computed path length was not multiplied with 1.5 like in the results given in figure 3.33. Taking the results of the occluded matching task into account feast percentages of 40% used for matching seem to be a reasonable choice anyway regardless of the specific matching task or parameters used. To have a constant percentage number of feasts for all matching tasks is important because in real life applications it is of course not always known if an occlusion problem or an enlarged contour problem is present especially in segmentation tasks.

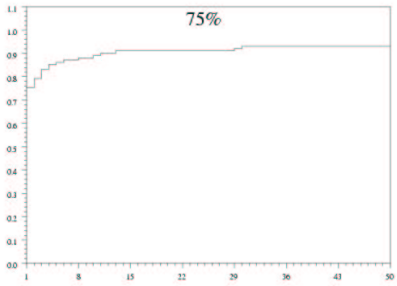
3.2.5. Matching of Open Contours

Although large parts of the work on a robust algorithm to match open contours remains a part of future research it is even with the current approach possible to give in figure 3.35 more than a proof of concept that the algorithm is able to deal with open contours successfully. This is important because if the concept can deal with open contours a new realm of possible input sources becomes available. For example the output of low level edge detectors which can be used by line completing algorithms. The presented method can then be used to judge which of the several possible line continuations is the most likely and should be chosen in the current user given context. Alternatively the different possible continuations can be assigned probability values computed from the similarities found matching in the current context. Furthermore the length of the line segment and the relative frequency of good matches found should be taken into account. The context given by the user can be implemented by choosing a desired composition of the target gallery database for matching.

For open contour or edge matching some effort has still to be done on the algorithms creating a single straight line of contour points to be interpolated by the B-spline algorithms. Furthermore different border conditions have to be implemented as a Fourier descriptor is used for the computations. The results shown were done with the wrap around border condition which is clearly not the best choice as to much artificial high frequency responses are induced in the Fourier domain. Furthermore it is probably a good idea to select Gabor values computed at different contour points to extend the matching algorithm to be able to match successfully even at the terminating ends of the open contour were no reliable low frequency information is available.



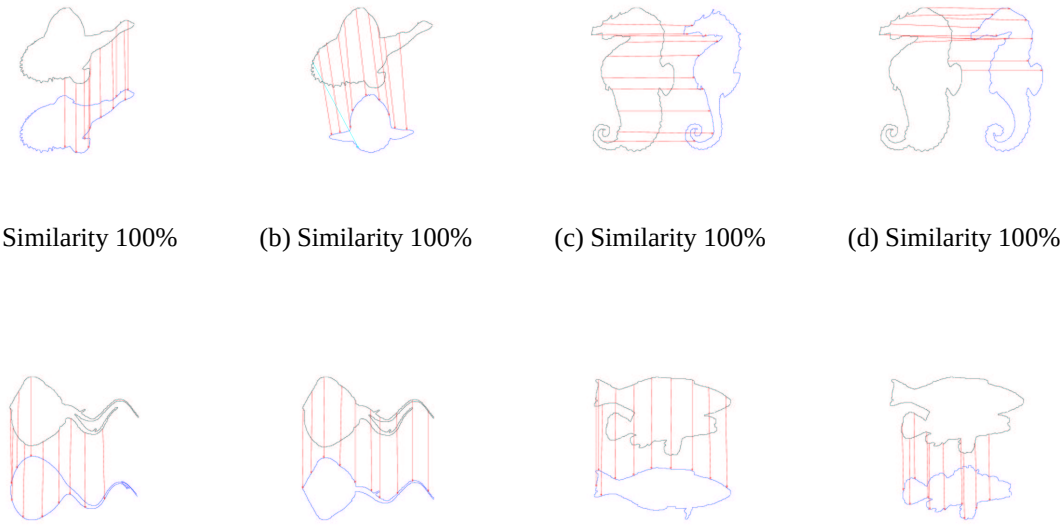
(a) Original Image



(b) 36 pixel translation

(c) Similarity 100%

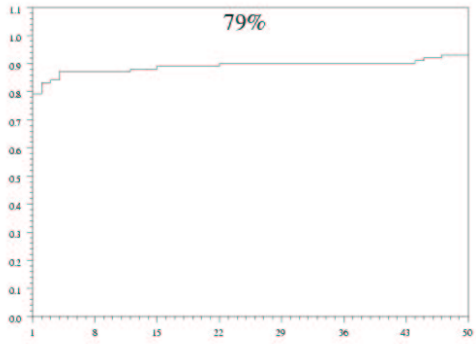
Figure 3.31.: Illustration of an enlarged contours recognition task. Used was only a small percent amount of all feast to find the best matching object. (a) A new image of the given task showing two objects superimposed on another with the center of the second object moved 36 pixels away from the center of the first object. (b) Accumulated right match positions. The percentage of right matches on rank one is given explicitly. (c) Example matching result illustrating the point to point match positions for one specific match. The results were obtained by matching complete contours that were subdivided into 25 interval ranges. The best matching contiguous 40% of these feast , as described in section 2.8.5.3, were used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.41425. f is the factor between two Gabor levels in the transformation.



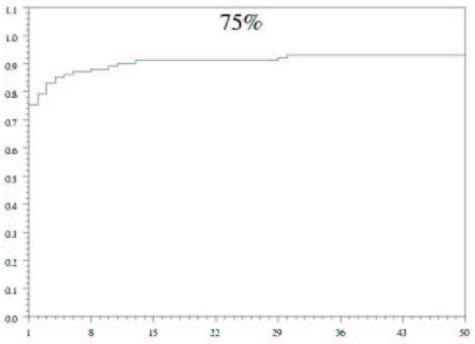
(a) Similarity 100% (b) Similarity 100% (c) Similarity 100% (d) Similarity 100%

(e) Similarity 99% (f) Similarity 99% (g) Similarity 100% (h) Similarity 100%

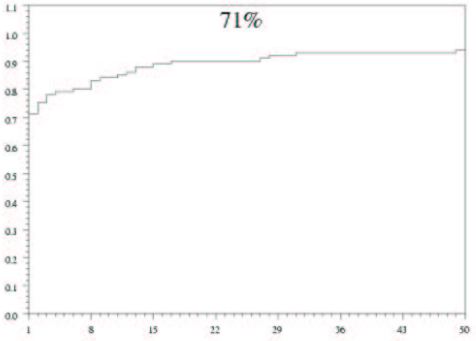
Figure 3.32.: Examples for the multiple object recognition task. Shown is the best and the second best match for an enlarged contour composed of two objects. In the presented examples both objects could be recognized exceptionally well. Of course this is not always possible. The recognition rates for finding at least one of the two objects to be the best match are given for various translations between the objects in figure 3.33. The results were obtained by matching complete contours that were subdivided into 25 interval ranges. The best matching contiguous 40% of these feast , as described in section 2.8.5.3, were used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.41425. f is the factor between two Gabor levels in the transformation.



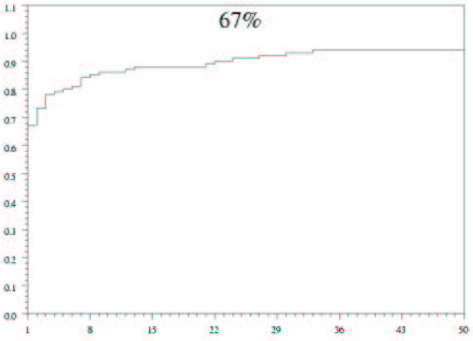
(a) 46 pixel translation



(b) 36 pixel translation

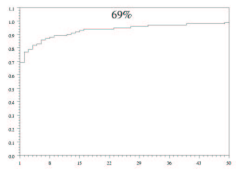


(c) 26 pixel translation

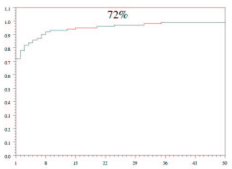


(d) 16 pixel translation

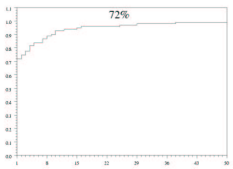
Figure 3.33.: Illustration of accumulated right match positions and recognition rates for the enlarged contours recognition task. Used was only a small percent amount of all feast to find the best matching object. Used for matching were images showing two objects superimposed on another with the center of the second object moved away from the center of the first object by different pixel amounts. The results indicate that there is only a marginal difference in the recognition rates for different degrees of superposition. The results were obtained by matching complete contours that were subdivided into 50 interval ranges. The best matching contiguous 22% of these feast , as described in section 2.8.5.3, were used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.41425. f is the factor between two Gabor levels in the transformation.



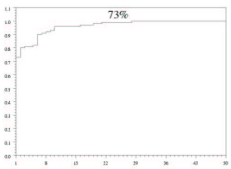
(a) 46 pixel translation



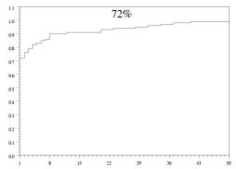
(b) 44 pixel translation



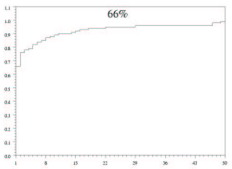
(c) 42 pixel translation



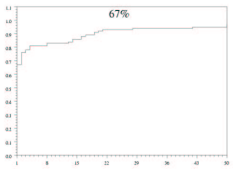
(d) 40 pixel translation



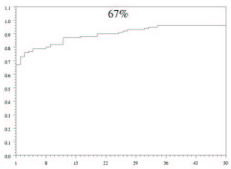
(e) 38 pixel translation



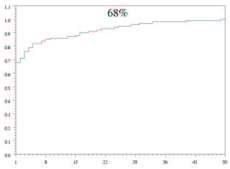
(f) 36 pixel translation



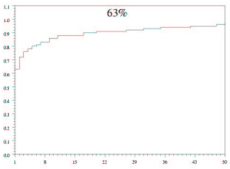
(g) 34 pixel translation



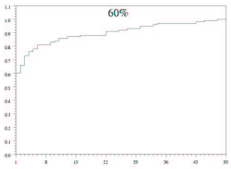
(h) 32 pixel translation



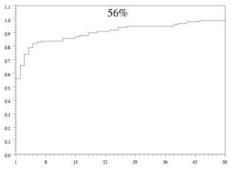
(i) 30 pixel translation



(j) 26 pixel translation

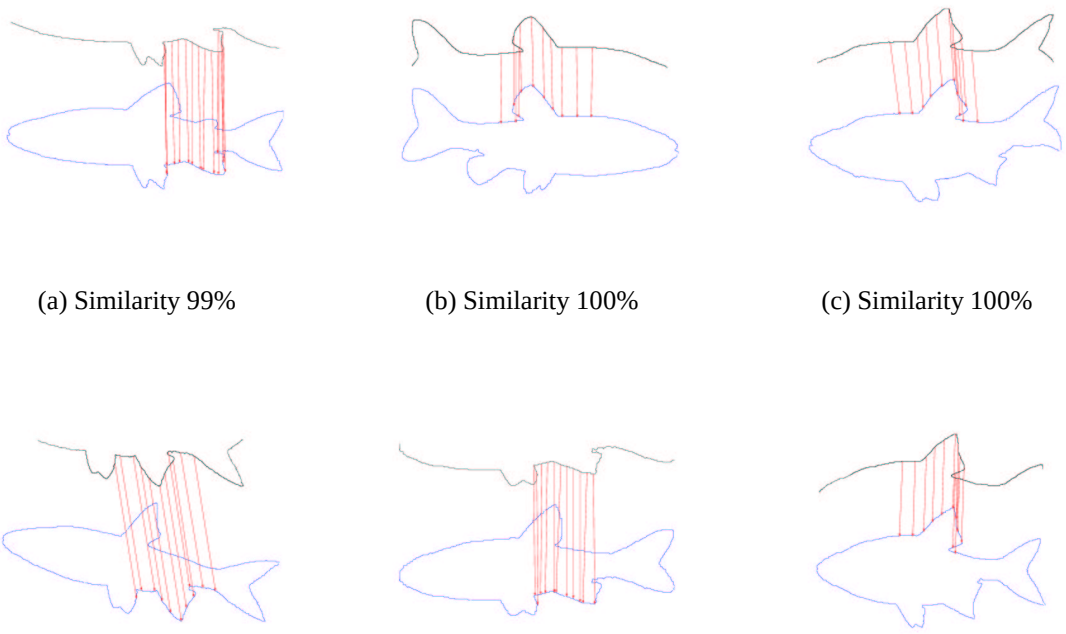


(k) 20 pixel translation



(l) 16 pixel translation

Figure 3.34.: Illustration of accumulated right match positions for the enlarged contours recognition task. Used was only a small percent amount of all feast to find the best matching object. Used for matching were images showing two objects superimposed on another with the center of the second object moved away from the center of the first object by different pixel amounts. These computations were done to examine the dependency of the recognition rate on different pixel translations The results were obtained by matching complete contours that were subdivided into 48 interval ranges. The best matching contiguous 30% of these feast , as described in section 2.8.5.3, were used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.



(a) Similarity 99% (b) Similarity 100% (c) Similarity 100%

(d) Similarity 100% (e) Similarity 99% (f) Similarity 100%

Figure 3.35.: Illustration of a recognition task involving the matching of open object contours or edge continuations that are used as input in the matching procedures. Shown are the point to point matches found. The results give a proof of concept that the presented algorithms can as well by used in recognition task involving open contours. This may be useful for example in cases where the input to match does not come from segmentation algorithms but is generated by edge detection algorithms. The results were obtained by matching selected points of the contours by using 11 feasts including and around each selected point. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.

4. Discussion

4.1. Properties of the Presented Model

It has been shown in the previous chapters that the presented model and algorithms for recognition of complete and only partially present contours can solve various important tasks in computer vision. First of all complete contours of an object can be used to recognize a particular object even if it has undergone affine transformation, e.g. was scaled or rotated in the plane. Not only is the object recognized but as well a continuous similarity measure is computed for the whole contour which can be used to cluster large databases of object data into certain groups. Furthermore even individual point to point correspondences are computed, which each have a similarity value, which can be used as a measure of reliability for the individual point to point matches.

If the object of interest has undergone rotations in 3D space it was shown that the object class and the object posture can be detected fairly well. Even if the exact posture is not present in the gallery database used for matching it can be estimated fairly well if the changes of the rotation angles are not too large. Furthermore it was shown that the object posture can be estimated if a sample of already known representatives of each object class is used as a gallery. Not only can the object class be estimated but as well a similar posture is found. In fact object class and posture are matched simultaneously.

The recognition of complete contours that have only been scaled and rotated in the plane can be done efficiently by normalizing the contours. The recognition rate for these tasks on large scale tests using the SQUID database comes close to and sometimes even is 100%. In cases when no normalization is possible matching algorithms have been proposed which can for example compensate for changes in scale and rotation in the plane.

To test which rotations steps can be tolerated by the feature set an examination of the recognition rate on rotated contours was done when no rotation normalization and no rotation matching have been applied. On average a two percentage points drop per additional degree of rotation was found for non-normalized contour matching.

To account for changes in the scale of the contour when no normalization has been applied the discrete scale concept was introduced. It has been shown that matching across all discrete scales produces good results regardless if the value of $\sqrt[4]{2}$ is chosen for the parameter f which determines the relation of successive Gabor levels. In this case the recognition rate drops only to 0.93.

The results for the hand gesture recognition task give more than a proof of concept that these task is solvable by the proposed algorithms but the results for this task as well suggest that it would be a great benefit to define individual graphs for each object or in this case hand gestures. These graphs should represent a collection of the important points of the object to recognize for

a given task. It is the opinion of the author that not only contour parts that have structure corresponding to a high information content relative to other contours by some measurement have to be considered for these graphs but it must be possible to encode especially the small detail differences — if needed — relevant to the user for a specific. The development of such a graph representation is not a major problem but it is currently not implemented and is therefore subject to future research. On a two GHz-CPU personal computer the current non-optimized implementation needs less than a second to compute the similarity value between two contours and needs a time amount in the range of several minutes to compare a contour with one thousand contours. If the above mentioned graph representation would be implemented the computational costs could be reduced by one or two powers of ten as the amount of points to be checked is reduced significantly. If the selected points were the right ones the results indicate a remarkable ability of the algorithm to deal with slight changes in the overall gesture.

The recognition of individual persons by their profile face images is another task that can be solved excellently. It was shown that the algorithm can deal with slight rotations in 3D that occur for example in stereo images of the same object. It was further shown that variations of the scale can be compensated and that very precise point to point correspondences can be found. The latter is an important problem in stereo vision.

In typical applications for example segmentation tasks the objects to recognize are often occluded to some extent. To find robust algorithms that can identify objects under this condition is an important goal. If the un-supervised percentage matching algorithm is used recognition rates of approximately 70% can easily be achieved by using approximately 20% of the contour for the matching process. This number is very remarkable as the maximum reachable is unknown. Some objects are so small or have such an unstructured contour that even humans can probably not achieve 100% if large parts of such objects are occluded. If user intervention was added and therefore the selected feasts algorithm is usable the recognition rate is almost independent from the degree of overlap of object and occluding ellipsoid as long as some parts of the contour are unaffected by the occlusion.

A proof of concept was given that the algorithm is able to deal with open contours. However, some effort has still to be done for open contour matching on the algorithms creating a single straight line of contour points. Furthermore different border conditions for open contours should be implemented as a Fourier descriptor is used in the computations. And it is probably a good idea to select Gabor values computed at different neighboring contour points to extend the matching algorithm to be able to match successfully even a the terminating ends of the open contour where no reliable low frequency information is available.

It was shown that the requirements a shape model should meet given in section 1.2 on page 17 are all fulfilled by the presented model. The only exception is the possibility to code the object of interest effectively for data compression purposes like fulfilling the MPEG-7 standard. This problem has to be addressed in detail in the future but the possible solution has been given above in the hand gesture section by stressing the need to define specific object graphs. Whether this can only be done task specific for each individual object by the user or an automatic way can be found were the user only has to supply a gallery of objects of interest is an open question. In any case the coding should not be dependent only on the object itself like in (Mokhtarian and Bober, 2003) were the zero crossing of a specific transformation are used but as well on the other contours relevant in the context of interest.

4.2. Relation to Other Models

The matching algorithm presented here is on an abstract level similar in spirit to the matching algorithm presented in (Lades et al., 1993) which originates from an early work done by Prof. von der Malsburg (von der Malsburg, 1981). The implementation of the matching algorithm was inspired by (Wiskott et al., 1997), (Wiskott and Malsburg, 1993) and (Wiskott, 1996).

The work presented here is related to the work of Loos (Loos, 2002). There a whole object graph is generated by a growing neural gas algorithm that represents the thresholded pixels of a difference image. The advantage of the method is certainly the 2D-graph matching approach that makes it possible — at least theoretically — to incorporate different 2D features in the matching process. However, the structure of the graph itself is totally dependent on the thresholded pixels of the difference image. Therefore it is very unlikely that two graphs of the same object generated under two different conditions will be equal or at least similar. So even if the object could be recognized by either graph in a new situation there is no direct way to compare the two different graphs or to merge them to a common graph. Another difference to the work presented here is the relative crude nature of the contour approximation. A typical number of nodes being used is 30. This is a rather low number compared to the typical number of contour interpolation points which is in the interval of 512 to 1536. The low number of nodes used reduces the discriminating power when two objects are compared as large parts of the contour are not covered at all. For example the recognition of individual profile faces is probably not at all possible with this approach. As graphs of the same object generated under different conditions may be quiet different it is not possible to give reliable point to point correspondences between different graphs. Even if the same graph is used, for example, for matching two stereo images the corresponding points found are probably crude estimates of the real corresponding points and therefore a depth estimation is highly unreliable. An advantage of his approach is that it can be done with little or no information about the scene and that it is computationally less costly than the presented algorithms.

4.3. Choice of Parameters

Important questions arise on what is the best choice for some of the parameters. The reader may have already noticed that for f — the factor between two Gabor levels to use (see equation 2.7 — most of the times $\sqrt{2}$ was chosen but sometimes $\sqrt[4]{2}$. To use the value $\sqrt[4]{2}$ for f increases of course the number of Gabor levels substantially and leads therefore to an overall increase of the computational cost. However, the sampling of the Gabor kernels in frequency space is so dense then that a change in scale of the contour can nearly fully be compensated by the scale matching algorithm. As it was shown this is not important for matching complete contours as the context information is very strong in this task. It plays however an important role if only contour parts or open contours are matched. So while it is safe to use $\sqrt[4]{2}$ especially for partial or open contour recognition it is much faster and usually sufficient to use $\sqrt{2}$ for the factor f. Especially when normalization is possible it is sufficient to use $\sqrt{2}$ and one can even try 2 which may work well if the objects to distinguish have strongly different contours.

If small differences of contours or probably only of certain parts of a contour should be distinguished it is useful to reduce the percentage part of a contour that influences each computed

Gabor coefficient by increasing c_D (see equation 2.8) and simultaneously to compensate for the reduction of Gabor levels by using a lower f for example $\sqrt[4]{2}$. The effect is a higher sensitivity to smaller contour details. However, notice that this is not always desired as the ability to classify and probably even to recognize contours by there overall shape at all decreases and can even vanish when adopting these parameter changes.

4.4. The Presented Shape Model and Object Segmentation.

Segmentation is usually defined as the segregation of a figure from the background of the visual input examined. However using only this criterion it has turned out that segmentation is an ill defined problem and therefore still unsolved in computer vision. The main reason is that the intention of the person doing the segmentation, all the background knowledge he has and his experience acquired so far modify the visual input that is received in a way that first of all enables a successful segmentation. It is not finding an object in the visual input and then identifying it but rather identifying and therefore finding it. However, this process — even if the intention of the user is known — cannot be modeled computationally using only the low level visual input without using a high level knowledge base. This is the reason why some part of the segmentation community regard object segmentation and object recognition or classification to be the same, especially if they are not interested in a segmentation result that is accurate to the level of single pixels. On the other hand there are a lot of segmentation methods that only deal with the low level information and probably some general rules mostly derived from physics. The output they deliver is pixel oriented but there is no general method that succeeds for all segmentation tasks, and there probably never will be.

Therefore it is a great challenge to create a feedback loop so that the shape model combined with the low level segmentation process itself could help to extract contour information with pixel accuracy in cases where other low level methods fail. However, the low level segmentation method presented in this work is not easily extensible to incorporate the contour information as a much more complex multi cue approach is needed. Nevertheless, to combine the presented shape recognition method with a suitable low level segmentation method remains a very interesting task for future research.

In the presented model the acquired gallery contour represents a high level knowledge base and as the gallery can be created by the user even the users intention can be represented. Furthermore, this high level knowledge base can be acquired without any further programming effort by algorithm designers by just learning examples that were segmented under favorable circumstances and adding them to a database.

4.5. Organization of the Database Architecture

As it has been shown that a reasonable human-like clustering of the object postures is possible by their respective contour similarities it would only be consequent to reflect that in the organization of the database entries as well. However up till now only a straight forward approach has been used to store all contours linearly and even all search operation are done in that fashion. One could think of a model that first of all for each found cluster one representative is matched followed by all contours that could not be clustered up till now. The results can be sorted and

only on the best matching results — if they represent clusters — is a subsequent search done to find the really best matching entry. This approach is as well subject to future research.

4.6. Combining Contour and Area Information

One important result is that contour recognition works best on profile images or side views of objects. This can be verified for the task of recognizing humans by their profile face. But even horse, car, cow or dog silhouettes are best recognized using their side views. Frontal views do not differ that much and therefore to have a really robust object recognition or segmentation method the presented one dimensional approach should be combined with matching methods using intra-object 2D data. However, here a real challenge is given as it is not clear how such a combination should be realized at all. Maybe another approach is more promising. If it is possible to have two camera images of the same scene from different viewpoints so that reliable input for both approaches can be recorded then a combination of the results can be done at a later stage. By this approach an excellent recognition rate for human frontal detection can probably be achieved on datasets expanded by a magnitude of three or four powers of ten compared to current sizes of datasets for frontal face identification tasks that are of an order of several ten thousand individuals. Such an approach would make face recognition applicable to real life large databases, although of course this assumption has to be confirmed by further large scale tests.

4.7. The Ideal Meta-Matching Algorithm

Consider the situation that an autonomous robot system with vision capabilities enters a new environment. First of all it has to build a representation of the outside world to be able to navigate and to identify objects. It is the opinion of the author that a one step feedforward algorithm to solve this problem is questionable or at least very unlikely to exist at all. If the environment was encountered before, things may be different, but that is not the situation we examine right now. The first step should probably be an holistic approach to guess a context for example by examining the properties of 2D Fourier coefficients or other crude cues. If a context has been selected the system can select certain databases for segmenting and recognizing individual image contents. If a stereo camera pair is present various edge hypotheses generated by edge detection and edge continuation can for example be matched to form a first hypotheses on which points correspond with each other. Form this point on various methods could probably be used to generate a depth map. If certain edge forms or other cues are already known for the selected context to be reliable depth cues a depth map may even be formed only by one camera image. Now the selected databases can be used to identify objects within the scene. Depending on the context the selected databases may only contain objects expected to be found in such an environment and the algorithms may have an expectation on where to find them in the image. Now object classes may be identifiable using all kinds of preprocessing results like low level segmentation, edge color or texture cues. The effect is the generation of even more competing hypotheses which can be examined further by using more detailed databases and parameter adaptions useful for recognition of these object classes or even to identify individual representatives. If the detection of the object classes fails the initial hypotheses have to be assigned a low value of

reliability and in the worst case the search has to be extended on the whole knowledge base represented in the system. In some situations the described process may even process object candidates that are not already part of the knowledge base for example if external motion is present. Once reliable scene and object estimates have been found the important yet unsolved task remains on how this new knowledge may be used to adapt the internal representations and the databases used. The intention of the design of the presented algorithms can be understood best if this general approach of generating and evaluating hypotheses is kept in mind. It is in strong contrast to the traditional approach in mathematics and computer science which is always assuming some presuppositions to be given and not to be questioned in the further processing steps. Unfortunately experience with real life applications in computer vision teaches us that this assumption is rarely true and that one has to constantly revalue the hypotheses that have lead to the preliminary results so far. As in the described scenario above the knowledge base already present in the system plays an important role and very interesting questions arises:

- What should such an knowledge base look like?
- How can the knowledge base finally be built up incrementally by the system?
- What is the minimal starting knowledge that needs to be given by the computer scientist to the autonomous system.

While these questions have to remain largely unanswered at least for the problem of object recognition by contour parts a contribution was made on what the data model and algorithms should look like to be usable in the various stages of hypotheses testing where it is useful to apply them. In part II a very special problem related to the third question will be examined taking a more neurophysiological perspective.

A. Further Examples of Matching Results

A.1. Further Results on the Recognition of Different 3D Object Postures.

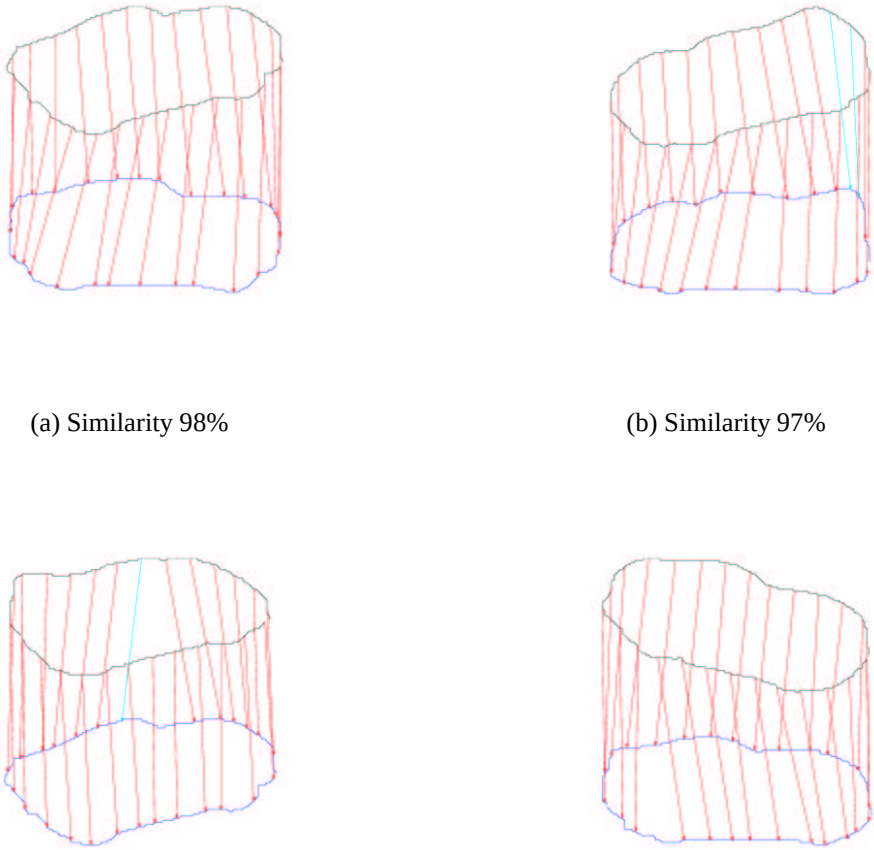
A. *Further Examples of Matching Results*

(a) Similarity 90% (b) Similarity 98% (c) Similarity 95% (d) Similarity 96%

(e) Similarity 95% (f) Similarity 97% (g) Similarity 97% (h) Similarity 96%

(i) Similarity 95% (j) Similarity 98%

Figure A.1.: Examples for matching different 3D postures for automobiles. Red arrows indicate similarities of 90% and above. The car matching example is given to illustrate the ability of the algorithm to tolerate rotations in 3D. Each individual automobile dataset from the ETH80 database was split in two disjunct parts, so that the same posture was not present in both datasets. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were $\sigma = 3.1414$ and $f = 1.18925$. f is the factor between two Gabor levels in the transformation.



(a) Similarity 98%

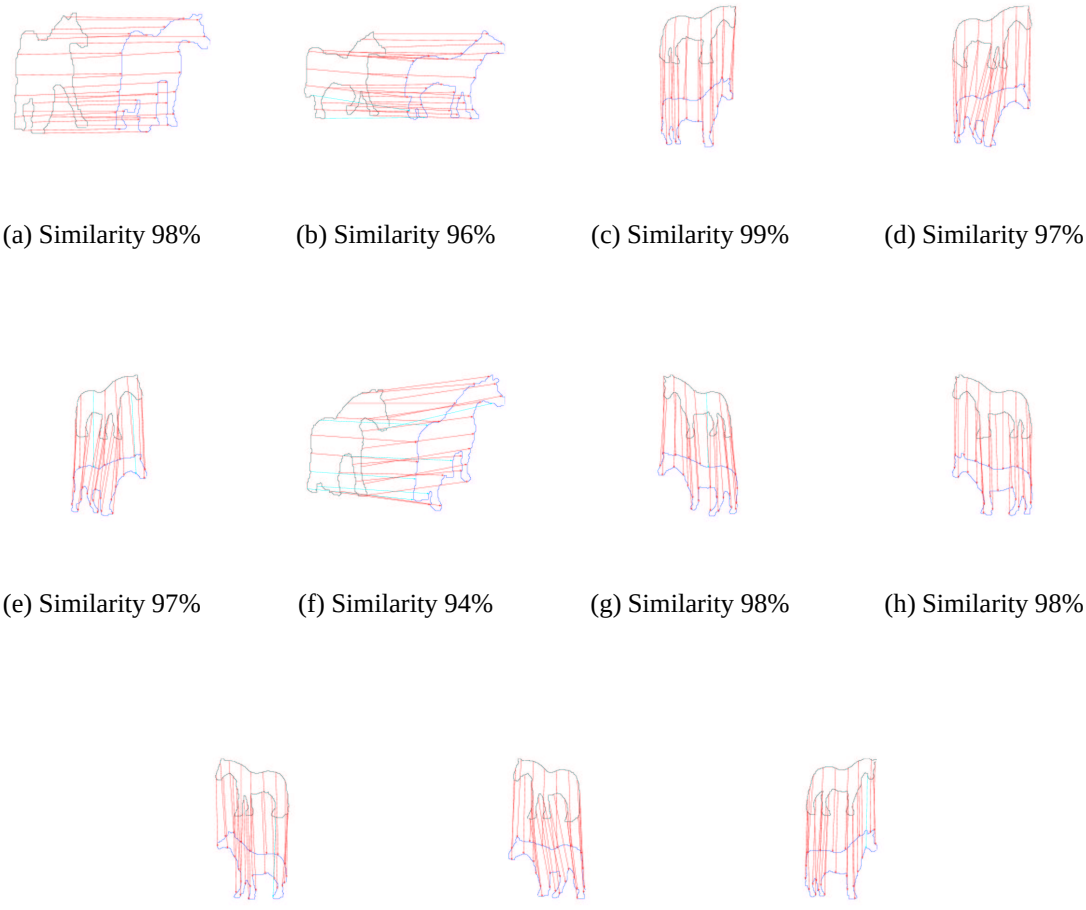
(b) Similarity 97%

(c) Similarity 95%

(d) Similarity 97%

Figure A.2.: Further examples for matching different 3D postures of cars. Red arrows indicate similarities of 90% and above. The car matching example is given to illustrate the ability of the algorithm to tolerate rotations in 3D. Each individual automobile dataset from the ETH80 database was split in two disjunct parts, so that the same posture was not present in both datasets. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.

A. Further Examples of Matching Results

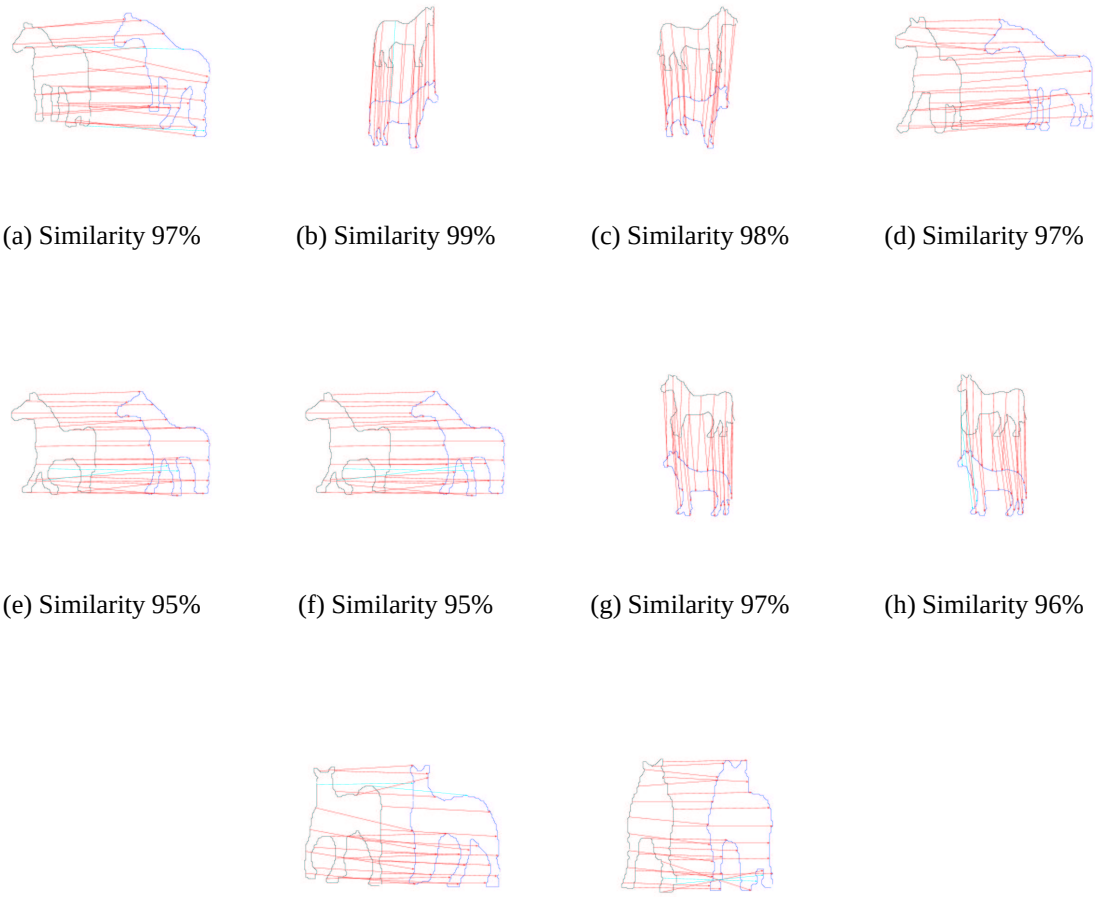


(a) Similarity 98% (b) Similarity 96% (c) Similarity 99% (d) Similarity 97%

(e) Similarity 97% (f) Similarity 94% (g) Similarity 98% (h) Similarity 98%

(i) Similarity 98% (j) Similarity 98% (k) Similarity 98%

Figure A.3.: Examples for matching different 3D postures of horses. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for three representatives was used to build an object contour gallery. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated with relatively high accuracy. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were $\sigma = 3.1414$ and $f = 1.18925$. f is the factor between two Gabor levels in the transformation.



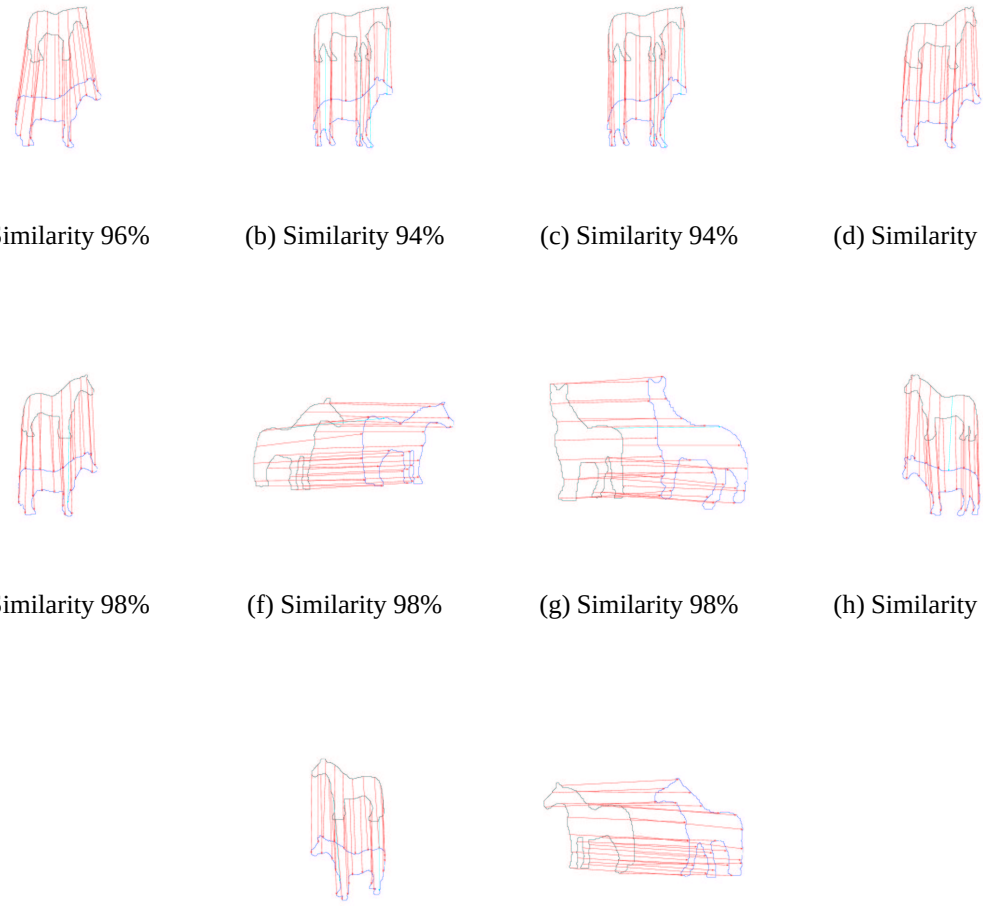
(a) Similarity 97% (b) Similarity 99% (c) Similarity 98% (d) Similarity 97%

(e) Similarity 95% (f) Similarity 95% (g) Similarity 97% (h) Similarity 96%

(i) Similarity 96% (j) Similarity 92%

Figure A.4.: Examples for matching different 3D postures of horses. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for three representatives was used to build an object contour gallery. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated with relatively high accuracy. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were $\sigma = 3.1414$ and $f = 1.18925$. f is the factor between two Gabor levels in the transformation.

A. Further Examples of Matching Results

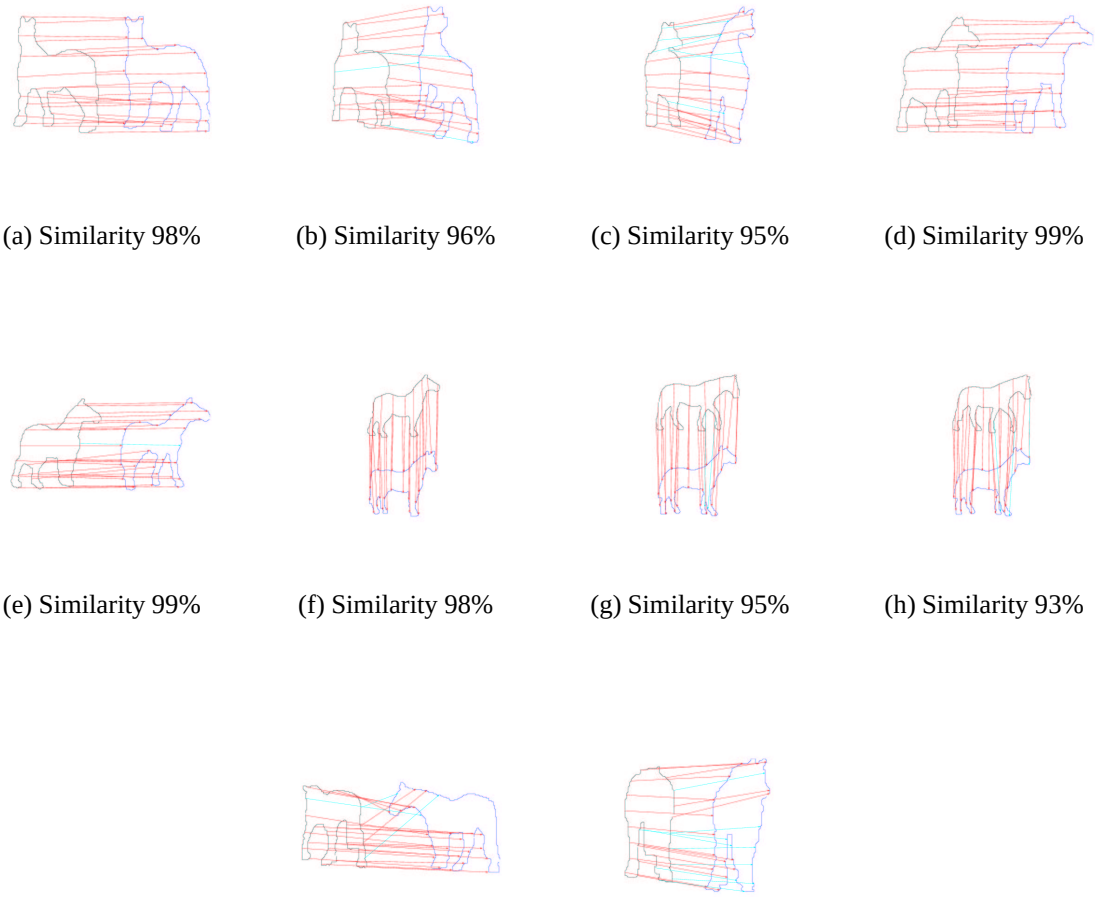


(a) Similarity 96% (b) Similarity 94% (c) Similarity 94% (d) Similarity 99%

(e) Similarity 98% (f) Similarity 98% (g) Similarity 98% (h) Similarity 98%

(i) Similarity 97% (j) Similarity 98%

Figure A.5.: Examples for matching different 3D postures of horses. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for three representatives was used to build an object contour gallery. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated with relatively high accuracy. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were $\sigma = 3.1414$ and $f = 1.18925$. f is the factor between two Gabor levels in the transformation.



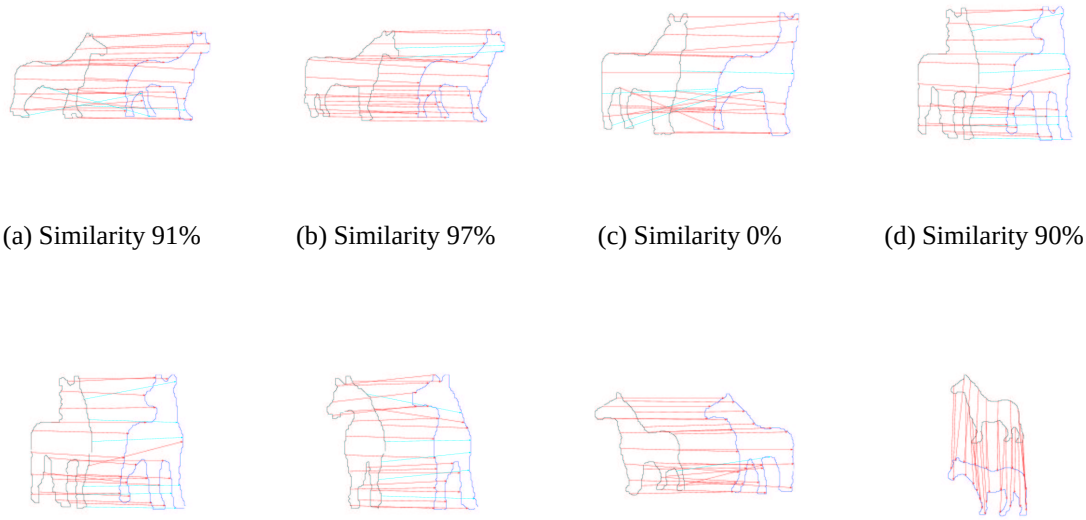
(a) Similarity 98% (b) Similarity 96% (c) Similarity 95% (d) Similarity 99%

(e) Similarity 99% (f) Similarity 98% (g) Similarity 95% (h) Similarity 93%

(i) Similarity 94% (j) Similarity 94%

Figure A.6.: Examples for matching different 3D postures of horses. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for three representatives was used to build an object contour gallery. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated with relatively high accuracy. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were $\sigma = 3.1414$ and $f = 1.18925$. f is the factor between two Gabor levels in the transformation.

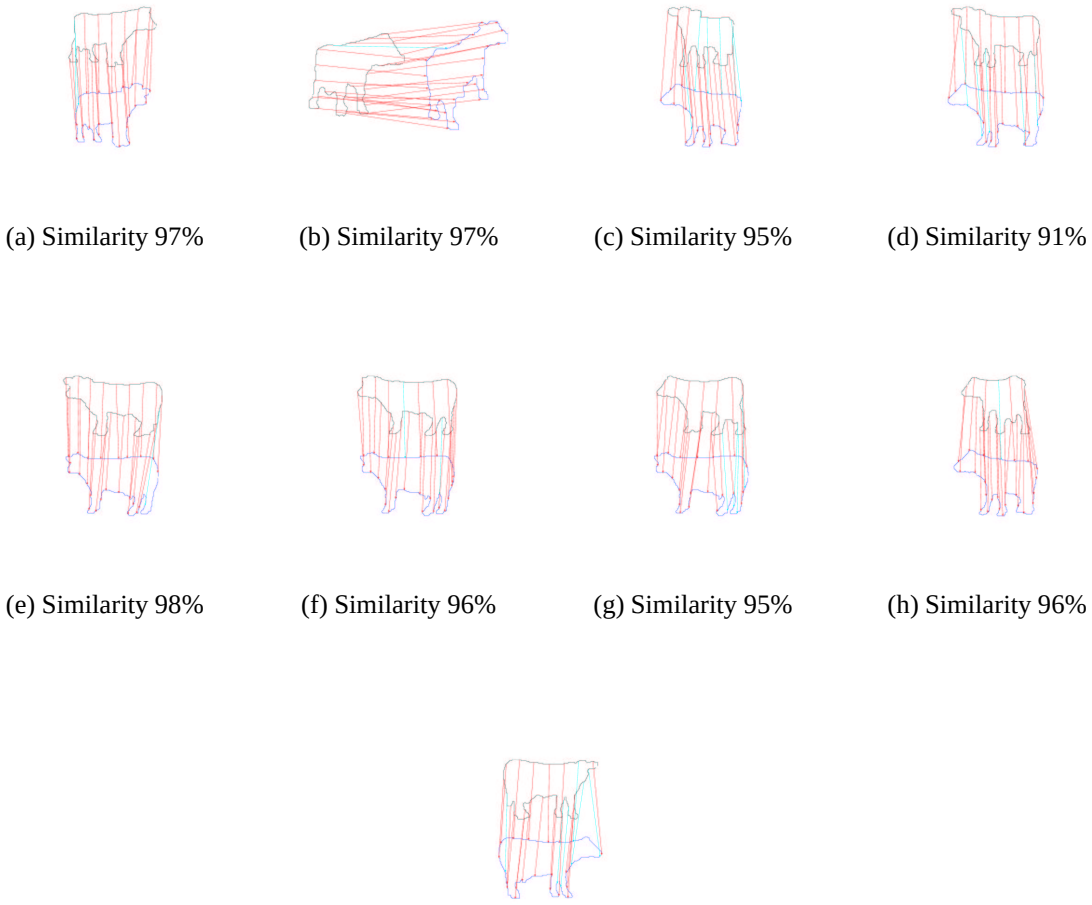
A. Further Examples of Matching Results



(a) Similarity 91% (b) Similarity 97% (c) Similarity 0% (d) Similarity 90%

(e) Similarity 90% (f) Similarity 93% (g) Similarity 94% (h) Similarity 94%

Figure A.7.: Examples for matching different 3D postures of horses. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for three representatives was used to build an object contour gallery. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated with relatively high accuracy. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself only used the 0 scale as defined in section 2.8.2. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.

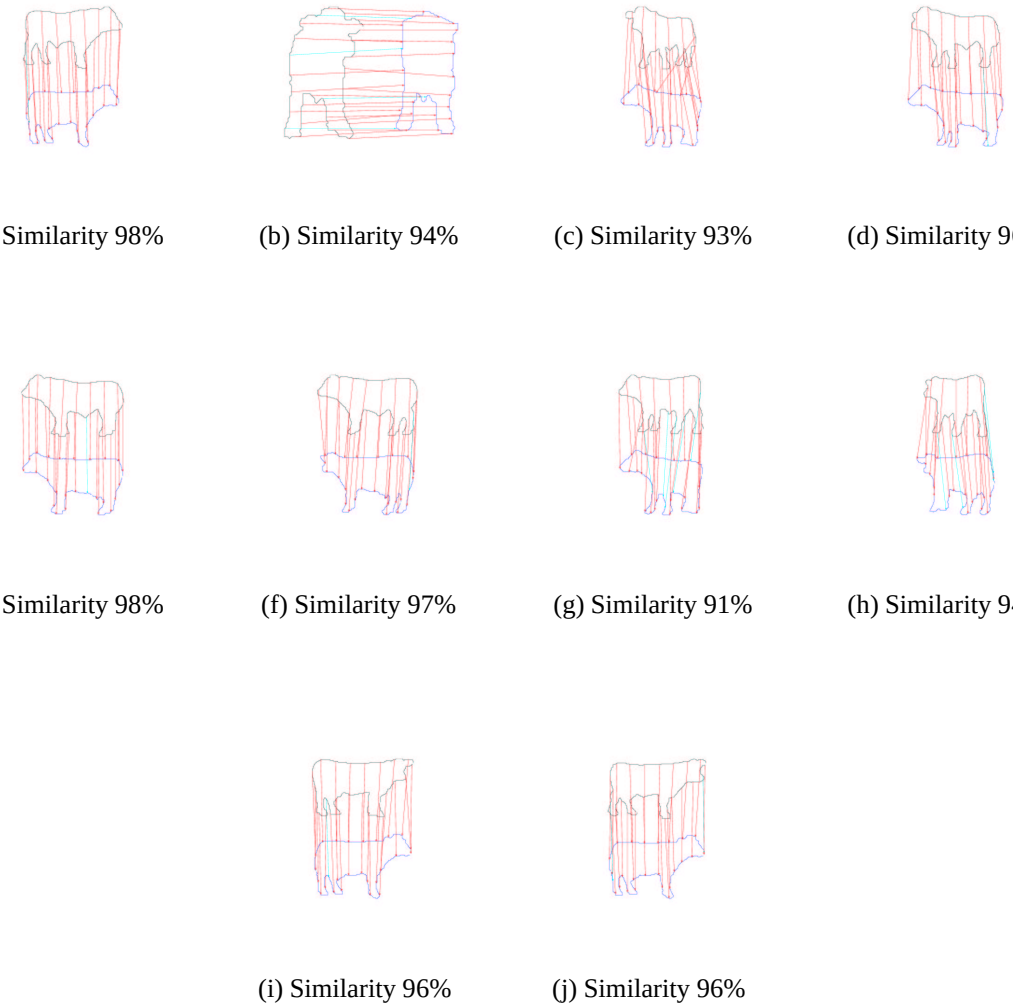


(a) Similarity 97% (b) Similarity 97% (c) Similarity 95% (d) Similarity 91%

(e) Similarity 98% (f) Similarity 96% (g) Similarity 95% (h) Similarity 96%

(i) Similarity 95%

Figure A.8.: Examples for matching different 3D postures of cows. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for three representatives was used to build an object contour gallery. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated with relatively high accuracy. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were $\sigma = 3.1414$ and $f = 1.18925$. f is the factor between two Gabor levels in the transformation.

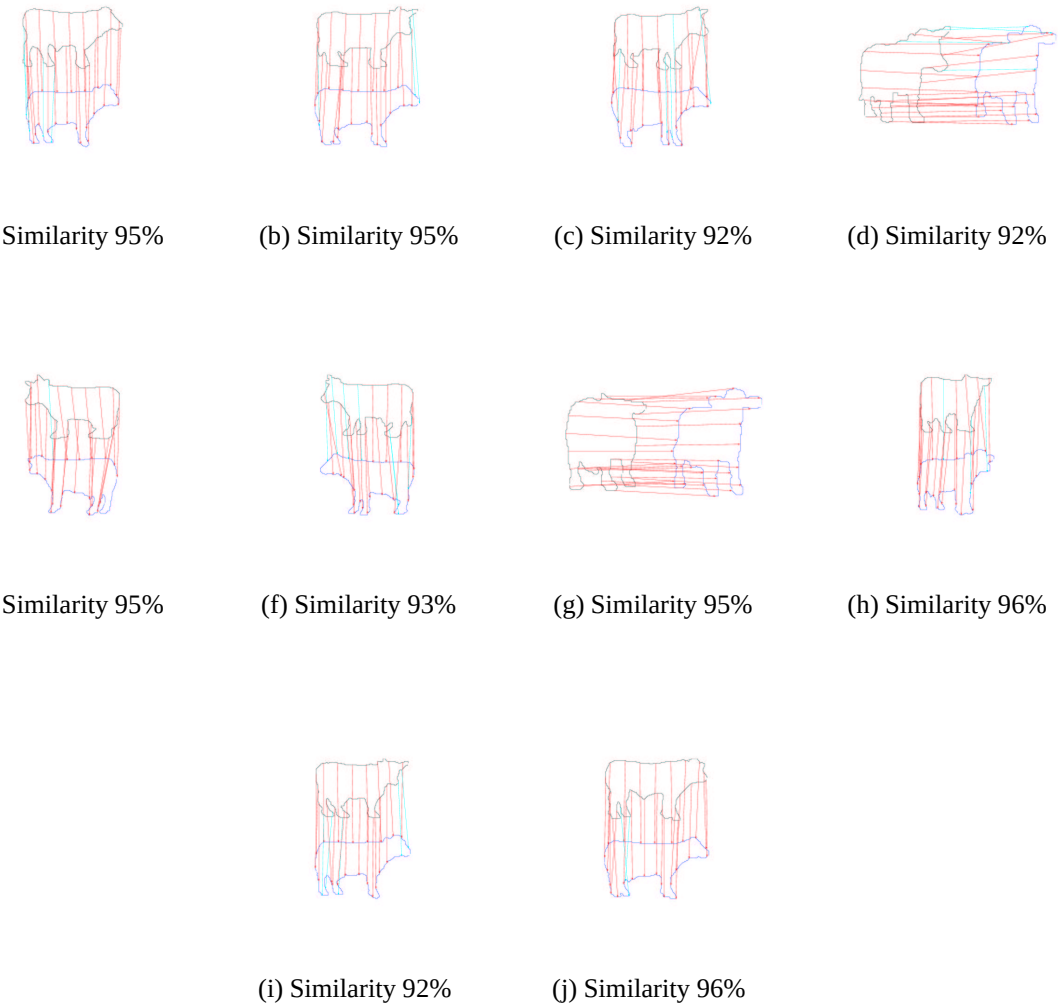


(a) Similarity 98% (b) Similarity 94% (c) Similarity 93% (d) Similarity 96%

(e) Similarity 98% (f) Similarity 97% (g) Similarity 91% (h) Similarity 94%

(i) Similarity 96% (j) Similarity 96%

Figure A.9.: Examples for matching different 3D postures of cows. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for three representatives was used to build an object contour gallery. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated with relatively high accuracy. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were $\sigma = 3.1414$ and $f = 1.18925$. f is the factor between two Gabor levels in the transformation.



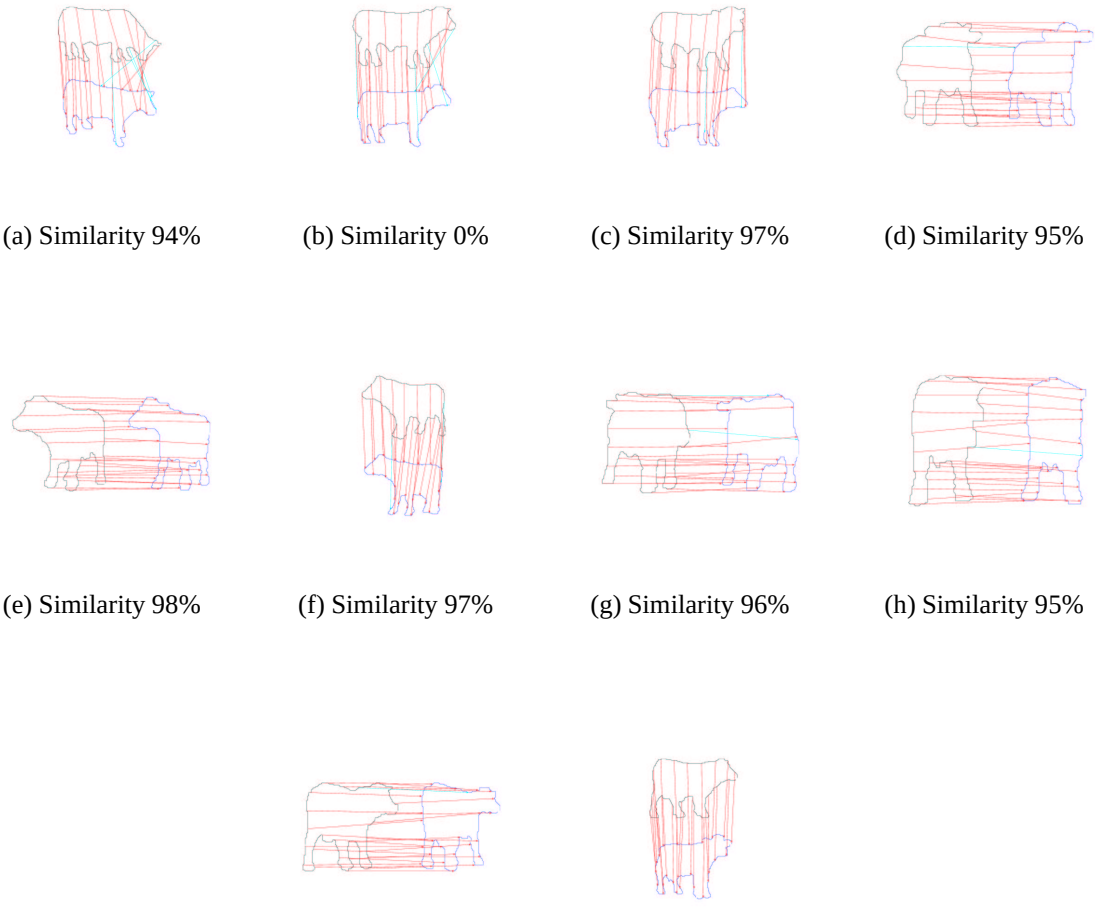
(a) Similarity 95% (b) Similarity 95% (c) Similarity 92% (d) Similarity 92%

(e) Similarity 95% (f) Similarity 93% (g) Similarity 95% (h) Similarity 96%

(i) Similarity 92% (j) Similarity 96%

Figure A.10.: Examples for matching different 3D postures of cows. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for three representatives was used to build an object contour gallery. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated with relatively high accuracy. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were $\sigma = 3.1414$ and $f = 1.18925$. f is the factor between two Gabor levels in the transformation.

A. Further Examples of Matching Results

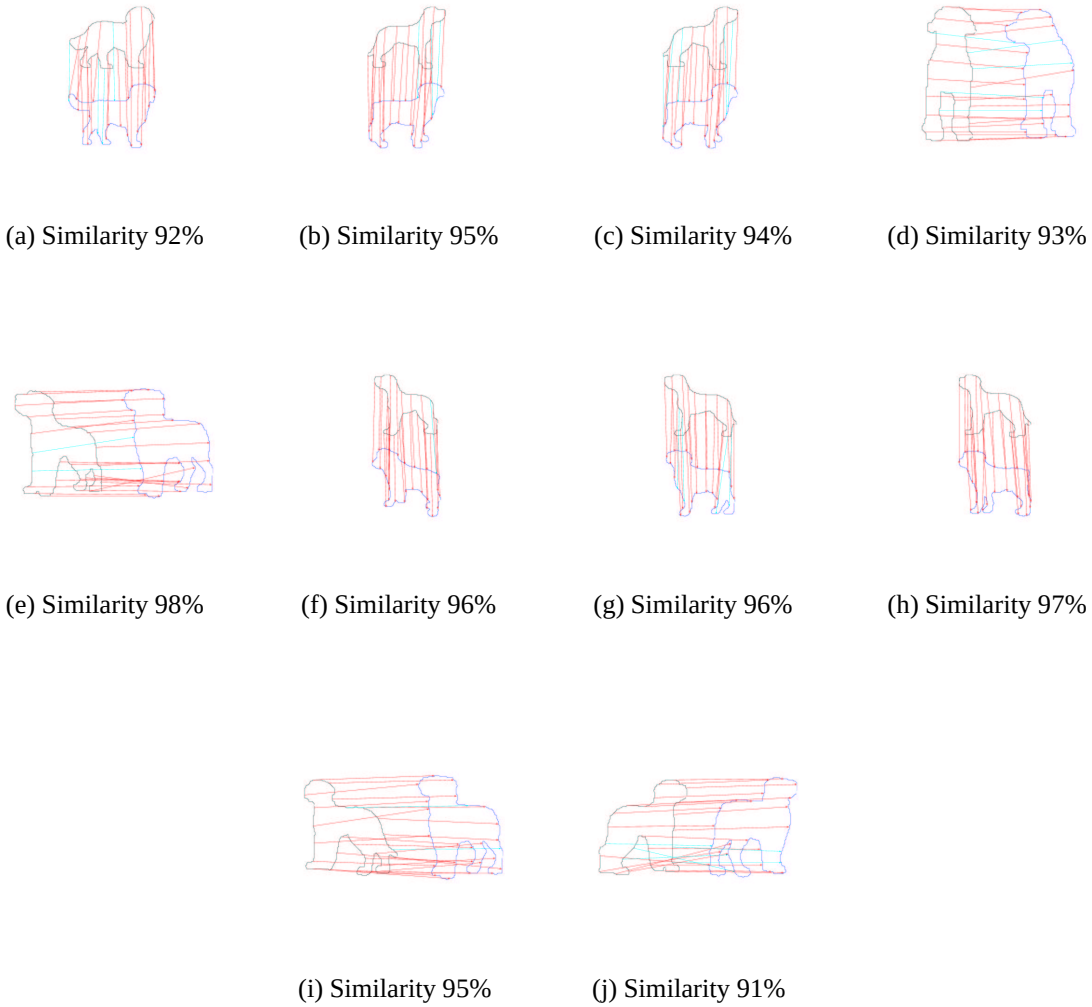


(a) Similarity 94% (b) Similarity 0% (c) Similarity 97% (d) Similarity 95%

(e) Similarity 98% (f) Similarity 97% (g) Similarity 96% (h) Similarity 95%

(i) Similarity 96% (j) Similarity 94%

Figure A.11.: Examples for matching different 3D postures of cows. The reference contour used for generating these results is shown at the top of the figures. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for three representatives was used to build an object contour gallery. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated with relatively high accuracy. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were $\sigma = 3.1414$ and $f = 1.18925$. f is the factor between two Gabor levels in the transformation.



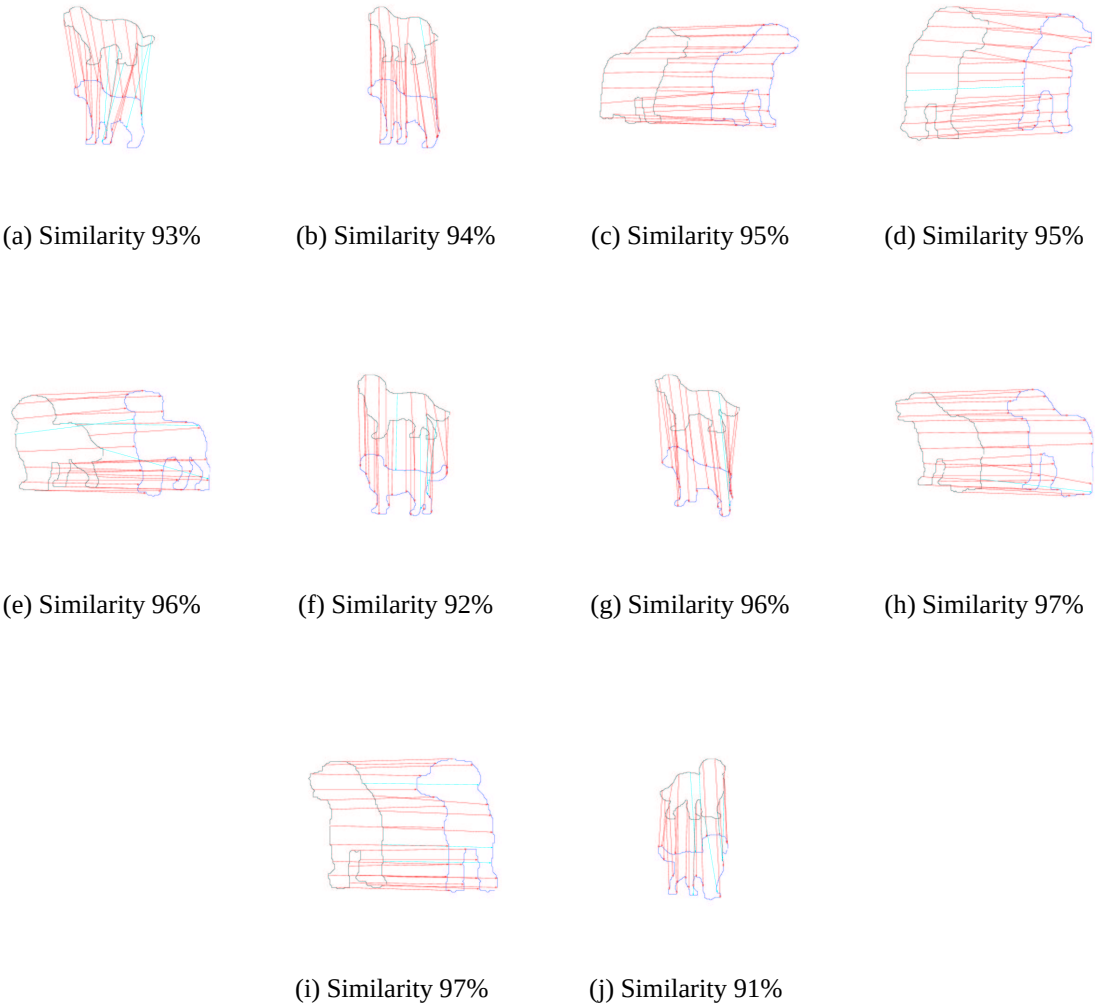
(a) Similarity 92% (b) Similarity 95% (c) Similarity 94% (d) Similarity 93%

(e) Similarity 98% (f) Similarity 96% (g) Similarity 96% (h) Similarity 97%

(i) Similarity 95% (j) Similarity 91%

Figure A.12.: Examples for matching different 3D postures of dogs. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for three representatives was used to build an object contour gallery. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated with relatively high accuracy. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were σ = 3.1414 and f = 1.18925. f is the factor between two Gabor levels in the transformation.

A. *Further Examples of Matching Results*



(a) Similarity 93% (b) Similarity 94% (c) Similarity 95% (d) Similarity 95%

(e) Similarity 96% (f) Similarity 92% (g) Similarity 96% (h) Similarity 97%

(i) Similarity 97% (j) Similarity 91%

Figure A.13.: Examples for matching different 3D postures of dogs. Red arrows indicate similarities of 90% and above. Out of ten different representatives of this object class taken from the ETH80 database the complete set of images for three representatives was used to build an object contour gallery. The other representatives were subsequently matched against this gallery and the results show that the corresponding points can be estimated fairly well and even the 3D posture can be estimated with relatively high accuracy. The results were obtained by matching complete contours that were subdivided into 25 interval ranges and the number of interval ranges is equal to the number of final feast to feast matches used for the similarity computation of the contours. The number of interpolation points used for each contour was the nearest multiple of powers of 2,3,5,7 and 11 to the length of the path of the contour in the original image where the contour was extracted from multiplied by 1.5. The matching algorithm itself was done on all possible scales, if the sizes of the two contours to match were different. The parameters for the Gabor transformations used were $\sigma = 3.1414$ and $f = 1.18925$. f is the factor between two Gabor levels in the transformation.

B. Preprocessing of the Image Data

B.1. Illustration of the Color Cue and its Limitations

The region-based color cue approach has the advantage that for larger thresholds the number of regions reduces strongly and the remaining edges — borders between regions — have a high probability of containing at least a part of an object contour see figure B.1 (d). The regions contain probably only a part because it is likely that the large regions generated by crude thresholds will contain parts that belong to different objects.

(a) Image

(b) Grouping by High Similarity Using a Low Threshold

(c) Grouping by Medium Similarity Using a Medium threshold

(d) Grouping by Low Similarity Using a High Threshold

Figure B.1.: Illustration of the threshold dependency of the color similarity cue: (a) The original image. (b)-(d) Grouping operations are performed using a low (b), medium (c), high (d) threshold which means only very high (b), medium (b) and even very low (d) similarities between the color of two neighboring pixels are sufficient to group the pixels together to one region. Please refer to B.1 on page 119 for an explanation on how the color cue has been used in this work.

B.2. Converting RGB to L*a*b* Color Values

Given a RGB color value (R, G, B) with $R, G, B \in [0, 255]$ the following equations are used to transform the RGB color values to L*a*b* color values:

$$
\begin{aligned}
var_R &= \begin{cases} (\frac{\frac{R}{255}+0.055}{1.055})^{2.4} \cdot 100 & \text{for} \quad \frac{R}{255} > 0.04045 \\ \frac{R}{255 \cdot 12.92} \cdot 100 & \text{for} \quad \frac{R}{255} \leq 0.04045 \end{cases} \\
var_G &= \begin{cases} (\frac{\frac{G}{255}+0.055}{1.055})^{2.4} \cdot 100 & \text{for} \quad \frac{G}{255} > 0.04045 \\ \frac{G}{255 \cdot 12.92} \cdot 100 & \text{for} \quad \frac{G}{255} \leq 0.04045 \end{cases} \\
var_B &= \begin{cases} (\frac{\frac{B}{255}+0.055}{1.055})^{2.4} \cdot 100 & \text{for} \quad \frac{B}{255} > 0.04045 \\ \frac{B}{255 \cdot 12.92} \cdot 100 & \text{for} \quad \frac{B}{255} \leq 0.04045 \end{cases} \\
X &= var_R \cdot 0.4124 + var_G \cdot 0.3576 + var_B \cdot 0.1805 \\
Y &= var_R \cdot 0.2126 + var_G \cdot 0.7152 + var_B \cdot 0.0722 \\
Z &= var_R \cdot 0.0193 + var_G \cdot 0.1192 + var_B \cdot 0.9505 \\
var_X &= \begin{cases} \frac{X}{95.047}^{\frac{1}{3}} & \text{for} \quad \frac{X}{95.047} > 0.008856 \\ (7.787 \cdot \frac{X}{95.047}) \ + \ (16/116) & \text{for} \quad \frac{X}{95.047} \leq 0.008856 \end{cases} \\
var_Y &= \begin{cases} \frac{Y}{100.000}^{\frac{1}{3}} & \text{for} \quad \frac{Y}{100.000} > 0.008856 \\ (7.787 \cdot \frac{Y}{100.000}) \ + \ (16/116) & \text{for} \quad \frac{Y}{100.000} \leq 0.008856 \end{cases} \\
var_Z &= \begin{cases} \frac{Z}{108.883}^{\frac{1}{3}} & \text{for} \quad \frac{Z}{108.883} > 0.008856 \\ (7.787 \cdot \frac{Z}{108.883}) \ + \ (16/116) & \text{for} \quad \frac{Z}{108.883} \leq 0.008856 \end{cases} \\
L* &= (116 \cdot var_Y) - 16 \\
a* &= 500 \cdot (var_X - var_Y) \\
b* &= 200 \cdot (var_Y - var_Z)
\end{aligned}
$$

B.3. A Psychophysical Acceptable Distance Function for Color Differences in HSI Space

In order to generate a psychophysically acceptable distance function for color differences in HSI space one has to create a distance function Dist which I name the 'shadow distance' for reasons that become apparent later in the following way: Each pixel of the image RGB_{Im} and the background image RGB_{Bg} is transformed from the discrete RGB space ($[0, 255]^3$) to the HSI space (figure B.2) first. Please note that with decreasing luminance ('I' component) a lot of 'color' values are coded that simply correspond to the psychophysical impression of black.

$$RGB_{\mathrm{Im}} \ \rightarrow \ HSI_{\mathrm{Im}} \quad RGB_{\mathrm{Bg}} \rightarrow HSI_{\mathrm{Bg}} \tag{B.1}$$

Then the shadow distance function between to pixels Dist is defined using the difference in luminance values L_{Diff} weighted by a constant factor L_{Imp} plus the difference in saturation

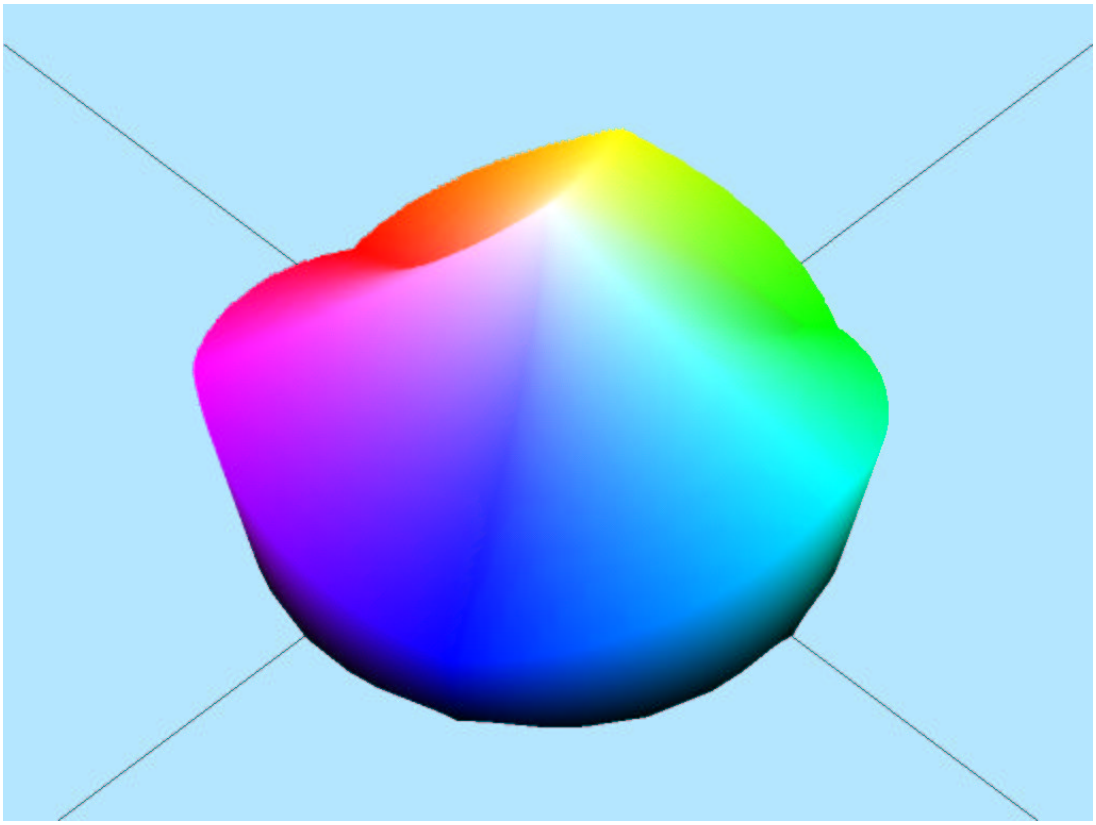


Illustration of the HSI color space

Figure B.2.: Illustration of the HSI color space. Shown is the transformation of the discrete RGB cube defined on $[0, 255]^3$ to the HSI color space. 'I' or the Luminance component corresponds to the vertical axis. 'H' (hue) is essentially an angle representing a color code and 'S' (saturation) represents the distance of the color to the luminance axis 'I'. The planar axes shown can be chosen arbitrarily. Please refer to B.3 on page 121 for explanation on how the HSI color space can be used for object segmentation.

values S_{Diff} as well weighted by a constant impact factor S_{Imp} and a non trivial distance measure H_{Con} of the color difference explained below which itself is dependent on the luminance and saturation values.

$$\begin{aligned} \text{Dist} &= L_{\text{Diff}} \cdot L_{\text{Imp}} + S_{\text{Diff}} \cdot S_{\text{Imp}} + H_{\text{Con}} && \text{(B.2)} \\ L_{\text{Diff}} &= |I_{\text{Im}} - I_{\text{Bg}}| \quad S_{\text{Diff}} = |S_{\text{Im}} - S_{\text{Bg}}| && \text{(B.3)} \end{aligned}$$

Assuming that neither of the hue values is equal to the no hue value (255 in most implementations) the distance measure H_{Con} for the color component is computed by the cyclic difference H_{Diff} of the hue values.

$$H_{\text{Con}} = \frac{H_{\text{Diff}}}{H_{\text{Space}}} \cdot H_{\text{Imp}} \qquad \text{(B.4)}$$

The cyclic difference H_{Diff} has to be used as the hue is essentially an angle (see figure B.2 for an illustration). H_{Space} will be defined below and H_{Imp} is a constant weighting the impact of the hue differences.

$$H_{\text{Diff}} = \min\left(\left|H_{\text{Im}} - H_{\text{Bg}}\right|, \left|\min\left(H_{\text{Im}}, H_{\text{Bg}}\right) + \left|255 - \left(\max H_{\text{Im}}, H_{\text{Bg}}\right)\right|\right|\right) \qquad \text{(B.5)}$$

Note that because the cyclic difference has to be used the maximal difference in hue values is 128 and not 255 as for example for luminance or saturation values. H_{Space} is computed to weight hue differences according to the current luminances L_{Fac} and saturations S_{Fac}. For very low or very high luminance values and for low saturation values hue differences are weighted less than if they had been measured at an average luminance value with a saturation above a certain constant S_{max}. L_{Fac} is designed in a way that if both of the luminance values are in an interval centered at the medium luminance value, L_{Fac} is constant one and falls off linearly from one to zero beyond that interval at very low or very high luminance values. Important is only the part considering low luminance values as one can see figure B.2 for high luminance values the color saturation diminished rapidly. For low luminance values this is not the case which has the effect that if for example a shadow of an object falls on a white wall there are suddenly huge color differences detected although no real significant change in color has occurred. The precise interval borders where the L_{Fac} starts to decrease to zero can be controlled by the constants L_{max} and L_{min}.

$$L_{Fac} = \min\left(\max\left(\left(128 - \left(\left|\max\left(L_{\text{Im}}, L_{\text{Bg}}\right) - 128\right|\right) - L_{\text{min}}\right), 0\right) / \left(L_{\text{max}} - L_{\text{min}}\right)\right), 1\right) \qquad \text{(B.6)}$$

S_{Fac} is designed to decrease linearly from one to zero if both saturation values are below a constant S_{max}. If both saturation values are below the constant S_{min} it even decreases to zero which has the effect that hue differences have to be very large to have any impact.

$$S_{\text{Fac}} = \min\left(\left(\max\left(\max\left(S_{\text{Im}}, S_{\text{Bg}}\right) - S_{\text{min}}, 0\right)\right) / \left(S_{\text{max}} - S_{\text{min}}\right), 1\right) \qquad \text{(B.7)}$$

The product $S_{\text{Fac}} * L_{\text{Fac}}$ can be viewed as a significance factor for the weighting of hue differences. To actually compute what impact a hue difference should have we have to evaluate what amount of difference in hue values H_{Space} should transform to one unit of difference taking into account the trustworthiness of the measured hue values. We therefore use the constants $H^{\text{diff}}_{\text{max}}$ and $H^{\text{diff}}_{\text{min}}$ to set limits for the amount of hue difference that is necessary to generate one unit

difference. If the hue significance factor is high (equals one) small difference like $H_{\min}^{\text{diff}}$ in hue equal one unit and if the factor is low $H_{\max}^{\text{diff}}$ huge difference in hue values are necessary to generate one unit of difference. The interpolation between these two extremes in not linear as the constant exponent $H_{\exp}$ is used to 'punish' only highly unreliable hue values.

$$H_{\text{Space}} = (1 - S_{\text{Fac}} * L_{\text{Fac}})^{H_{\exp}} * \left|H_{\max}^{\text{diff}} - H_{\min}^{\text{diff}}\right| + H_{\min}^{\text{diff}} \tag{B.8}$$

For other color spaces like RGB, YUV etc. the formulas required would have to be even more complex. These impressive set of formulas makes it clear why the L*a*b* color space and not any other color space is used throughout this work as by using the L*a*b* color space one can avoid the need to define a complex metric or distance function for color values.

C. Graph Algorithm for the Generation of a Chain of Points

This part of the appendix is only presented for readers interested in implementing some of the low level graph generation algorithms. If you are only interested in the scientific results of this work it is better to avoid reading them.

C.1. An Algorithm for Extracting the Different Components of a Graph

The algorithm `CyclesAndComponents` — based on an algorithm presented in (Breymann, 1996) — is illustrated in figure C.1. We first transform the undirected graph whose creation was described in the section 2.4 into an directed graph by inserting two opposite edges in the new graph for each undirected edge in the old graph and provide this new graph as input for the function `CyclesAndComponents`. The function `CyclesAndComponents` uses a container class N_{state} representing the states of the nodes and E_{state} for the states of all edges, respectively. The states for nodes can have these values: `for_NODE_NOTVISITED`, `for_NODE_SEEN` and `for_NODE_DONE` and the states for edges: `for_EDGE_NOTVISITED`, `for_EDGE_VISITED` and `for_EDGE_INVALID`. At the beginning the state of nodes is initialized with `for_NODE_NOTVISITED` and the edge states are initialized with: `for_EDGE_NOTVISITED`. The number of components and the number of cycles are set to zero and the container for the different components is cleared. An iteration over all nodes is done and for the next node with a state of `for_NODE_NOTVISITED` the number of components is increased by one and the following procedure is started: Push the current node on a stack and work on the nodes of this stack until the stack is empty again. If the stack is empty and there are still nodes in the state `for_NODE_NOTVISITED` found in the main loop a part of the graph could not be reached from the nodes we started with and we push this new node on the stack and start working on the stack again using the following procedure: For each node coming from the stack we put it in the current component of the component container and then check its state. If it is `for_NODE_VISITED` we can change it to `for_NODE_DONE` and ignore it further on. If it is `for_NODE_NOTVISITED` or `for_NODE_SEEN` we change its state to `for_NODE_VISITED` and look for the existence of edges starting at this node. The state of these found edges is checked and if the state of an edge is not `for_EDGE_INVALID` we continue working on it. The state of the edge is set to `for_EDGE_VISITED` and the state of the reverse edge to `for_EDGE_INVALID`. Afterwards we check the state of the target node of the current edge in question. If this node state is `for_NODE_VISITED` or `for_NODE_SEEN` the number of cyles is increased by one as the algorithm has already used this node. If the state

```
1  Function CyclesAndComponents ()
2      Convert edges of an undirected graph into two opposite directed edges
3      N_state := Note states for all nodes node initialized with N_state(node) = for_NODE_NOTVISITED
4      E_state := Edge states for all edges edge initialized with E_state(edge) = for_EDGE_NOTVISITED
5      components =0, cycles = 0, componentContainer.clear()
6      foreach node in N_state
7          if N_state(node) equals for_NODE_NOTVISITED or for_NODE_SEEN
8              ++components
9              push node on stack
10     while stack is not empty pop the last node
11         componentContainer[components].insert(node)
11         if N_state(node) equals for_NODE_VISITED
12             N_state(node) = for_NODE_DONE; continue
13         if N_state(node) equals for_NODE_NOTVISITED or for_NODE_SEEN
14             N_state(node) = for_NODE_VISITED;
15             push node on stack
16             foreach edge of node
17             if E_state(edge) equals for_EDGE_INVALID continue
18                 if N_state(targetnode(edge)) equals for_NODE_VISITED or for_NODE_SEEN
19                     ++cycles
20                 if N_state(targetnode(edge)) equals for_NODE_NOTVISITED or for_NODE_SEEN
21                     push node on stack
22                     N_state(targetnode(edge)) = for_NODE_SEEN
23     return cycles, components, componentContainer
```

Figure C.1.: Algorithm for computation of the number of cycles and components of a graph. See C.1 on page 125 for a detailed explanation.

of the node is `for_NODE_NOTVISITED` or `for_NODE_SEEN` we push this node on the stack again and set its state to `for_NODE_SEEN` and process the next subsequent edge. If there is non we continue to pop nodes from the stack and repeat the process described above. As mentioned above we continue with the first iteration over all edges in the graph if the stack is empty.

C.2. A Shortest Path Graph Algorithm With Different Preferences For Nodes To Be Used

For the first step of creating a counterclockwise graph as described in section 2.4 no problems can occur. It is always possible to find a subgraph of the original graph from the WN to the NE point. The optimal path is computed using the well known Dijkstra algorithm (Dijkstra, 1959) starting at the target node NE on the original graph with the distance between the pixels corresponding to the nodes as edge points. The Dijkstra algorithm computes the shortest path from a given node to all other nodes in the graph. Once it has been applied it is easy to extract the computed shorted path from NE to WN and insert it in the new counterclockwise oriented graph. Now a new graph has to be build each time — for the next step connecting NE with EN and all the other combinations of extreme nodes — that is equal to the old graph but does not contain the path already inserted in the counterclockwise oriented graph. The nodes already used are marked with the tag `for_NODE_VISITED` and these node states are provided to a modified Dijkstra algorithm that prefers to connect nodes that have not been used already and only uses the others if no other way can be found. The output of the modified Dijkstra algorithm is a data structure that codes for every specific node the preceding node of the path from the target node to that specific node. With this information it is possible to insert the wanted connections between a start and end node into the counterclockwise oriented graph. This complicated procedure is necessary as we must exclude that a part of an already used path will be used again, probably with the effect of corrupting the counterclockwise orientation. However for open contours or for graphs containing hairs one has to use the same nodes multiple times. The idea of alternatively marking the used edges fails for the following reasons: If one marks only the edges used so far and not the corresponding reverse edges the danger arises that for specific contours — consider for example the written letter 'E' — the shortest path from one extreme point to another has nodes that already have been used (although in a path with the opposite direction). In the example the EN (left top) extreme point can easily be connected via the other extreme points to the ES (left bottom) extreme point. However the shortest path from the ES to the EN extreme point is not the inside border of the 'E' but the outside border which already has been used. Of course this can happen when trying to connect the first extreme point with the second. Therfore we must compute the start of the counterclockwise orientation procedure using the heuristic that we start with this pair of adjacent extreme nodes that has the shortest path in the full graph and move on from their in the counterclockwise direction. If one removes the reverse edges of already used edges as well one runs into problems if *hairs* are in the graph. A hair of order 1 by definition is a node that is only connected and connects itself to the same single other node. A hair node of second order is a node that is connected and connects itself to a hair node of order one or two and to just one other node. A simple open contour consists therfore of two order one nodes and N-2 hair nodes of order two. If we remove both edges — the one running to and the one coming from a hair node of order one — with the effect of disconnecting the hair

```
1  Function DijkstraWithNodeStates(targetNode,N_states)
2      initialize two containers: precedingNodes with -1 and
           minEdgeCost with the maximum value of double numbers
3      set minEdgeCost(targetNode) = 0
4      initialize the dynamic priority queue dynPriQueue with the minEdgeCost container
5      while dynPriQueue not empty
6          pop the top node from dynPriQueue
7          set notVisitedNodesFound = false
8          foreach edge of node
9          if N_state(targetnode(edge)) equals for_NODE_NOTVISITED
10             N_state(targetnode(edge)) = for_NODE_SEEN
11             notVisitedNodesFound = true
12             if minEdgeCost(targetnode(edge)) > minEdgeCost(node) + edgeCost(edge)
13                 dynPriQueue.update(targetnode(edge), minEdgeCost(node) + edgeCost(edge))
14                 precedingNodes(targetnode(edge)) = node
15         if notVisitedNodesFound equals false
16             foreach edge of node
17             if N_state(targetnode(edge)) not equal to for_NODE_VISITED
18                 if minEdgeCost(targetnode(edge)) > minEdgeCost(node) + edgeCost(edge)
19                     dynPriQueue.update(targetnode(edge), minEdgeCost(node) + edgeCost(edge))
20                     precedingNodes(targetnode(edge)) = node
21 return precedingNodes
```

Figure C.2.: Algorithm for computation of the shortest path to a $targetNode$ from all other nodes of a graph where not already marked nodes `for_NODE_NOTVISITED` are preferred for the path computation.

node from the rest of the graph the algorithm of creating a counterclockwise oriented graph has to terminate, as we can not access any other node from the hair node anymore. Therefore the algorithm C.2 prefers to compute shortest path using nodes that have not been used up till now and only uses other nodes if this is the only way to proceed.

C.3. An Algorithm to Transform a Graph into a Sequence of Nodes

Given a graph of connected contour points the task remains to transform the graph into a sequence of nodes. By convention the first node of the graph is used as the starting point in the function `ExportReachableNodesMultiEdge()` and the sequence of stored edges for each node is preserved in a depth search in order to be able to travel along hairs or open contours. Of course the found nodes can occur multiple times in the output chain as paths of the graph can run several times through them. The function first checks if the starting node is a part of the graph and we assume that to be always true. The function `ExportReachableNodesMultiEdge()` uses a container class N_{state} representing the states of the nodes and E_{state} for the states of all edges, respectively. The states for nodes can have these values: `for_NODE_NOTVISITED`, `for_NODE_SEEN`, `for_NODE_VISITED` and `for_NODE_DONE` and the states for edges can be `for_EDGE_NOTVISITED` and `for_EDGE_VISITED`. At the beginning the state of nodes is initialized with `for_NODE_NOTVISITED` and the edges states are initialized with: `for_EDGE_NOTVISITED`. Create a container $nodeToRemember$ and a queue $nodeQueue$ for nodes and put in both containers the $startNode$. While the $nodeQueue$ is not empty pop the top $node$ of the queue and check its state. If its state is equal to `for_NODE_DONE` work on the next node of the queue. Push the $node$ at the end of a container $nodeChain$ that was created for the purpose to export the result. If the state of $node$ is `for_NODE_NOTVISITED` or `for_NODE_SEEN` change its state to `for_NODE_VISITED`. Set a boolean flag $foundFirstNotVisitedEdge$ to false. Work on every $edge$ originating at the current $node$. If the edge state equals `for_EDGE_NOTVISITED` change it to `for_EDGE_VISITED`. If the state of the target node of the current edge $N_{\text{state}}(\text{targetnode}(edge))$ equals `for_NODE_NOTVISITED` mark the node as `for_NODE_SEEN`. We now check if we have already found an edge that was not visited before.

The motivation for this is that as the main purpose is to transform the graph of points to a chain of points we prefer to perform a depth search on the edges, e.g. the first edge, of the starting node. The queue is additionally used to do a breath search after the depth search has terminated but there may have been some nodes not covered yet. So the algorithm conserve the topological ordering of the graph if it exists. In the case of multiple edges of one node we assume that a hair is present and that the first edge of the node goes to the hair. Therefore we have to mark the edges of the graph if we have seen them already and we only stop if we encounter a node that was already visited and that has no edge left to work on.

Now check if the state of the target node of the $edge$ $N_{\text{state}}(\text{targetnode}(edge))$ equals `for_NODE_NOTVISITED` or `for_NODE_SEEN`. So if the $edge$ and the target node were not worked on before we set $foundFirstNotVisitedEdge$ to true to remember that one has already found an edge and the next node for the depth search. For the same reason the

```
1  Function ExportReachableNodesMultiEdge(startPoint)
2      N_state := Note states for all nodes node initialized with N_state(node) = for_NODE_NOTVISITED
3      E_state := Edge states for all edges edge initialized with E_state(edge) = for_EDGE_NOTVISITED
4      push the startPoint in the queue nodeQueue and the container nodeToRemember
5      prepare the output result container nodeChain
6      while nodeQueue is not empty pop the next node from it
7          if N_state(node) equals for_NODE_DONE continue
8          push node at the end of nodeChain
9          if N_state(node) equals for_NODE_NOTVISITED or for_NODE_SEEN
10             N_state(node) = for_NODE_VISITED
11         set foundFirstNotVisitedEdge = false
16         foreach edge of node
17             if E_state(edge) equals for_EDGE_NOTVISITED
17                 set E_state(edge) = for_EDGE_VISITED
18                 if N_state(targetnode(edge)) equals for_NODE_NOTVISITED
19                     set N_state(targetnode(edge)) = for_NODE_SEEN
20                 if foundFirstNotVisitedEdge == false
21                     if N_state targetnode(edge)) equals for_NODE_NOTVISITED or for_NODE_SEEN
22                         foundFirstNotVisitedEdge = true
22                         push targetnode(edge)) on nodeQueue
23                 elseif foundFirstNotVisitedEdge == true
24                     if N_state targetnode(edge)) not equal to for_NODE_DONE
25                         if N_state targetnode(edge)) equals for_NODE_NOTVISITED
26                             N_state targetnode(edge)) = for_NODE_SEEN
27                         push targetnode(edge)) to back of nodeToRemember
28         if foundFirstNotVisitedEdge == false or edge found was last edge of node
29             N_state(node) =for_NODE_DONE
30         if nodeQueue empty and nodeToRemember not empty
31             insert the nodes of nodeToRemember in reverse order into the nodeQueue
32     return nodeChain
```

Figure C.3.: Algorithm for computation of the reachable nodes of a graph starting at a certain node. See C.3 on page 129 for a detailed explanation.

$\mathrm{targetnode}(edge)$) is pushed on the $nodeQueue$ so that the depth search will be continued in the next iteration. On the other hand if one had already found the first not visited edge which means $foundFirstNotVisitedEdge$ is equal to true then we check if state of the target node of the $edge$ $N_{\mathrm{state}}(\mathrm{targetnode}(edge))$ does not equal `for_NODE_DONE` else we terminate. If it equals `for_NODE_NOTVISITED` we mark it as seen (`for_NODE_SEEN`) and push it in the $nodeToRemember$ container to be able to perform a breath search after the the termination of the current depth search. If there are additional edges of $node$ repeat the described process above and if there were no not visited edges found or the first edge found is as well the last edge originating from $node$ mark node with $N_{\mathrm{state}}(node)$ =`for_NODE_DONE`. Work on the next $node$ from the $nodeQueue$ as described above and if the $nodeQueue$ is empty check if we have stored nodes in the $nodeToRemember$ container for an additional breath search. If some are present insert them in the reverse order of how they were found in the $nodeQueue$ and repeat the algorithm. The motivation of using the reverse order is that the last nodes found are probably relative close to the termination node of the last depth search, so the distance between nodes is kept relatively small. If $nodeToRemember$ was as well empty return the sequence of points using the $nodeChain$ container and terminate.

C. Graph Algorithm for the Generation of a Chain of Points

Part II.

Early Development of Biological Contour Processing

4. Early Development of Perception

4.1. Foundations of Perception

It has long been realized that perception is not a passive intake of unstructured sensory signals, but a highly selective and structured active process employing complicated and widely unknown rules to guide behavior [1]. If these rules are not hardwired at birth, and it will be argued in detail that some of them are not, this creates a vicious circle of knowledge depending on perception to arise and perception in turn depending on acquired knowledge to be possible. An attractive way to break this circle is the postulate that only some basic perceptual mechanisms are present at birth, which can be used to learn useful processing rules from the properties of the environment. These can lead to refined perception and, consequently, to the acquisition of refined knowledge about the environment. The evolutionary advantage gained from such a hierarchy would be a lower burden on the coding capacity of the genome and more flexibility to cope with such environmental changes that happen too fast for evolutionary changes.

The most prominent rules for visual perception are the *Gestalt principles* formulated by Wertheimer (1923) and Koffka (1935) . Their basic postulate is that a percept is more than the sum of the constituent parts in that these parts are linked to form a coherent whole. The Gestalt principles are rules that govern what should be linked together to form a percept and what should not. The most important ones are *proximity*, *similarity*, *closure*, *good continuation*, *common fate*, *good form*, and *global precedence*.

Recent research in computer vision has begun to recognize Gestalt principles as important for object selection and segmentation, and they have been applied in numerous models of visual perception, where they are usually taken for granted. To my knowledge, there are two models for their *development* during maturation of the visual system (Grossberg and Williamson, 2001; Choe, 2001), but I know of none that relies on short sequences of a single natural scene as the basis for learning.

I will present a detailed model of how the principles of *collinearity* and *curvilinearity* can be learned from real visual stimuli, assuming that the principle of *common fate* actively organizes perception already at birth. I will employ three assumptions, which are, to varying degree, covered by experimental data:

- Gestalt principles are implemented by the connectivity of the visual cortex.
- There is a hierarchy of Gestalt principles in the sense that some principles are already active at birth, others are developed later and influenced by visual experience. This hier-

[1] Parts of the presented work have been published in Neural Computation (Prodöhl et al., 2003).

archy is reflected in the temporal order in which the various principles become observable during development.

- The higher principles are learned from experience, while the lower ones assist in structuring the data to be learned.

Concretely, a review will be done on the literature to present evidence that the principles of collinearity and curvilinearity correspond to structured horizontal connections between simple cells in V1 and that the principle of common fate is more fundamental than collinearity and curvilinearity. Then, a neuronal model is described that shows that collinearity and curvilinearity can be learned from observing moving objects by structuring horizontal connections with a Hebbian learning rule.

The work is organized as follows. In section 4.2 a review is given on the recent psychophysical work on the course of development of various Gestalt principles in early infancy, in section 4.3 the focus is put on the role of *common fate*. In section 4.4 a look is done in depth at recent neurophysiological data about which biological pathways and computations are probably present at birth or shortly afterwards and which are structured by experience. The more technically minded reader may want to skip this introduction and move to section 2.1, where a simplified model of visual processing up to V1 and the proposed learning dynamics of the horizontal connections are described. The technical details can be found in the later sections. In section 6.1 the outcome of computational experiments using this model is presented. Finally, in the discussion the basic components which are considered crucial for the success of the model are extracted.

4.2. The Developmental Succession of Gestalt Principles

In this section psychophysical evidence that Gestalt principles are not present at birth, and that they develop one after the other is collected. Although for adults the Gestalt principles of *good form* or *good continuation* are dominant for perceiving an object as a unity children younger than one year can hardly make use of them (Spelke et al., 1993). The same holds for *texture similarity* and *color similarity*. 12-week old infants do not use these principles at all with regard to object unity and slowly start to employ them during the development in the first postnatal year. Detecting form at a very early stage seems to depend on continuous optical transformations caused by object or observer motion. At 24 weeks infants are unable to grasp 3-D object form from multiple stationary binocular views. However, if 16-week old observers are presented with a continuous geometrical transformation around a stationary 3-D object, they are able to build up a 3-D form representation (Kellman and Short, 1987).

The disparity information from the two eyes can not be used immediately after birth. It is only at the age of 4-8 weeks that the convergence accuracy of the eyes for distances between 25 cm and 200 cm reaches the accuracy of the adult (Hainline et al., 1992). Accommodation of the eyes becomes accurate at 12 weeks postnatally (Braddick and Atkinson, 1979). Stereo vision itself develops even later at the age of 16 weeks (Yonas et al., 1987). In the period between 12 and 20 weeks postnatally strong binocular interaction arises, which at 24 weeks culminates in a superior binocular acuity and the beginning of stereopsis (Birch and Salomão, 1998; Birch, 1985).

The contrast sensitivity for spatial frequency gratings increases between 4 and 9 weeks for all spatial frequencies in line with the convergence accuracy of the eyes (Norcia et al., 1990). At 9 weeks the acuity for low spatial frequency gratings reaches adult levels but for higher frequencies it is still three octaves worse than in the adult (Courage and Adams, 1996). From this time on the contrast sensitivity for high spatial frequency gratings increases systematically in line with the emergence of stereo vision. The color system develops even later: luminance contrast above 20% is reliably detectable at the age of 5 weeks, but there is still no response to isoluminant chromatic stimuli of any size or contrast. In the following weeks chromatic gratings are detectable only at low spatial frequencies with an acuity 20 times lower than that for luminance stimuli. The sensitivity to chromatic gratings increases more rapidly in the following weeks than the one for luminance stimuli (Morrone et al., 1990). Not only physical factors such as changing photoreceptor density may be responsible for the changes in contrast sensitivity but the neural noise is nine times higher in neonates than in adults and decreases to adult levels during the first eight months of development (Skoczenski and Norcia, 1998).

4.3. The Relevance of Rough Motion Information

In this section I discuss the biological relevance of motion information and collect evidence that it is available at an early stage of development. It was argued that neither form nor color nor disparity information can be processed at early postnatal stages, but luminance information at low spatial frequencies can. The latter is thought to be mediated mainly by the M-path, which is also essential for motion processing. The Gestalt principle of *common fate* realized by common motion of object parts is the dominating principle for perceiving the unity of an object in 12 week old infants (Kellman et al., 1986). This is independent of the direction of motion in three dimensional space. *Common motion* dominates *figural quality*, *substance*, *weight*, *texture* and *shape* in 16 week old infants (Streri and Spelke, 1989). At this age the infant is able to make a distinction between object and observer motion and uses only object motion for the generation of an object percept (Kellman et al., 1987).

The question arises what kind of computation is carried out regarding the visual information originating from a moving object detected by the retina. 24-week old infants can predict linear object motion in grasping tasks but have difficulties to do the same for visual tasks involving tracking of objects that are out of reach (Hofsten et al., 1998). The apparent inability to predict linear object motion together with the observation that at low spatial frequencies luminance information with sufficient contrast can be processed directly after birth, makes it very unlikely that the visual system is able to estimate accurate motion vectors at early stages of development. Nevertheless, *some* motion processing is important in early development: Piaget (1936) reported that even newborns react to high contrast stimuli moving in front of their faces. This is consistent with the findings on luminance gratings and it is concluded that at birth a system must be present that can detect changes in the visual world signaled by achromatic luminance stimuli of low spatial frequency.

I interpret these findings such that learning Gestalt principles relies on changes in low frequency luminance information at times when no egomotion occurs. Infants can actively create such a situation by gazing in a constant direction, where a moving object of sufficient size and luminance contrast is present, for a considerable amount of time. This specific behavior is in-

deed observed regularly, as has been reported, e.g., in (Piaget, 1936; Barten et al., 1971). The model on this particular situation, which from "within" the visual system is distinguished by strong transient responses in retina and cortex together with the knowledge that the observer is not moving. In order to complete the model two things remain to be specified.

1. What is the supposed neuronal substrate for the functional organization according to Gestalt principles?

2. How can this substrate be modified on the basis of the transient responses caused by object movement?

4.4. Development of the Visual Pathway

In order to motivate an answer to these questions now some facts are reviewed about the early development of connectivity in the visual pathway. Although for the retina the process of maturing is not complete after birth Tootle (1993) has shown that ganglion cells of cats show burst-like spontaneous activity and that those cells fatigue very quickly after repeated stimulation with the same stimulus in the first postnatal week. This functionally is interpreted as a kind of transient response property that detects changes in the visual input. The same author further showed that ON- and OFF-ganglion cells are already present at birth and that the proportion of light-driven ganglion cells approaches 100% in the second postnatal week. Furthermore the retinogeniculate (Snider et al., 1999) and the thalamocortical (Isaac et al., 1997) pathways develop mainly prenatally and are therefore present at birth. In visually inexperienced kittens 90% of all cells in Area 17 are of simple type and 70% of all visually active neurons of this area show a rudimentary orientation bias (Albus and Wolf, 1984), although only 11% are specifically tuned to one orientation. Most of these cells respond preferentially to contrast changes caused by decreasing light intensity as 76% of all responding neurons are activated by OFF-zones exclusively. This means that area 17 shows a sensitivity bias in favor of dark stimuli immediately after birth. The cells responding to visual stimuli are located in layers 4 and 6 of the striate cortex and there is almost no activity in layers 2/3 and 5 before three weeks postnatally. In the fourth postnatal week ON- and OFF-zones are equal in number and almost all cells in layers 4 and 6 show orientation tuning. Psychophysical experiments show that in the human infant contrast differences overrule orientation based texture differences in segmentation tasks (Atkinson and Braddick, 1992) up to the 12th week.

Now the attention is turned to the question of what the neural substrate for learning the Gestalt principles of *collinearity* or *curvilinearity* is. There is extensive evidence in the psychophysical (Field et al., 1993; Hess and Field, 1999; Kovacs, 2000), neurophysiological (Malach et al., 1993; Bosking et al., 1997; Schmidt et al., 1997; Fitzpatrick, 1997) and modeling (Grossberg and Mingolla, 1985; Li, 1998; Ross et al., 2000; Yen and Finkel, 1998; Grossberg and Williamson, 2001) literature that these principles are at least partly implemented by horizontal connections in V1. The models described in (Grossberg and Mingolla, 1985; Grossberg and Williamson, 2001; Ross et al., 2000), agree with perceptual data even to the point of reproducing illusionary contours.

Most but not all vertical interlayer local circuits in V1 of macaque monkeys develop prenatally in precise order without visual experience (Callaway, 1998). This means that axon

terminals at least find the right layer and already form a crude retinotopic projection. However, intralayer horizontal connections are present but only rudimentarily developed at birth, as most axon terminals have not yet hit their target cells. Studies of postmortem human brains show that the first horizontal connections develop 1-3 weeks before birth (37 weeks after gestation) in layers 4b and 5. Their number increases rapidly after birth and culminates in a uniform plexus at around 7 weeks after birth. The patchiness of these projections as it is found in the adult emerges after at least 8 weeks postnatally (Burkhalter et al., 1993; Katz and Callaway, 1992). The long-range connections can extend up to a maximum of four hypercolumns in each direction (Katz and Callaway, 1992). Burkhalter et al. (1993) further showed that consistent with the results of neuronal activity in kittens, layers 2/3 and 6 develop horizontal connections later than layers 4b and 5. In layer 2/3 they are not present until the 16th postnatal week and reach maturity in the 60th week. It is interesting to note that the connections in layer 2/3 are patchy from the start. This suggests that they can already benefit from the patchiness of the connections in layer 4b probably mediated by a direct vertical connection from layer 4b to layer 2/3 that develops after birth (Katz and Callaway, 1992). Furthermore, as layer 4b belongs to the M-path and provides direct input to area MT (which is strongly involved in motion processing) it is concluded that the processing of visual information related to motion precedes and probably supports the processing of form, color, precise stereoscopic depth, and their integration. This assumption is consistent with the psychophysical results mentioned earlier. The development of horizontal connections has been shown to depend on the visual input presented (Löwel and Singer, 1992).

4.5. Relation to Natural Image Statistics

A work about learning from natural stimuli is incomplete without discussing what is known in the literature about the statistics of such stimuli. The idea that the visual system is wired in a way that it provides an efficient and non-redundant representation of the incoming signals goes back to (Attneave, 1954) and (Barlow, 1961).

Based on this principle, there have been successful predictions of properties of retinal, LGN, and simple cells in V1. Examples without attempt on completeness include (Olshausen and Field, 1996; Bell and Sejnowski, 1997; van Hateren and Ruderman, 1998). A complete review of this line of work is beyond the scope of this paper but is done beautifully in (Simoncelli and Olshausen, 2001). Additional assumptions have to be employed, typically either the *sparseness* (Olshausen and Field, 1996) of a cortical representation or the statistical *independence* of the activities of the cells involved. The latter leads to properties of visual cells resulting from independent component analysis (van Hateren and Ruderman, 1998; Bell and Sejnowski, 1997). Also, *translation invariance* is usually assumed, because otherwise the required statistical basis would become intractably large.

Independent component analysis has recently been applied successfully to networks of V1 cells that support contour enhancement (Hoyer and Hyvärinen, 2002). They learn a feedforward layer of contour coding cells that take input from complex cells in V1. The underlying assumption is sparseness of coding.

With the presented work a slightly different approach is taken. I do not employ any assumption about sparseness or independence of cortical signals. Rather, the spatiotemporal properties

of the visual pathway up to V1 are modeled and a Hebbian learning to the horizontal connections between simple cells is applied. The model is more sophisticated in biological detail than others in this area. E.g., positivity of neuronal responses is always maintained. As a consequence, the stimuli are preprocessed in a nonlinear way before providing data for learning. The importance of such nonlinearities has been pointed out in (Zetzsche and Krieger, 2001). A feature that the model shares with the others is the explicit assumption of translation invariance, which leads to weight sharing during learning. This assumption is rather unbiological but hard to avoid for keeping computation times acceptable.

5. A Model for the Early Development of Horizontal Connections.

5.1. Complete Model Overview

Starting from the data just reviewed we assume that the specific connectivity pattern of long-range horizontal connections provides the neural basis for the Gestalt principles of collinearity and curvilinearity. This notion is supported by the apparent cooccurrence of the use of these principles and the maturation of the respective connections during development. This answers the first question raised at the end of section 4.3 about the neural substrate of Gestalt principles. In order to answer the second one about how these connections develop depending on object motion, we propose a quantitative model of how the transient retinal stimuli are propagated towards the cells interconnected by the axons in question (figure 5.1) and how their connection strengths are modified. We model the relevant parts of the visual pathway running from the retina via the retinogeniculate and thalamocortical connections to the simple cells of layer 4b in primary visual cortex and apply a Hebbian learning rule to shape the connectivity.

All functions we will use to describe our model are functions of two dimensional space but we omit this dependency for convenience of notation. The full details of the retina model are given in section 5.2.1 and we only summarize the important features here: The retinal photoreceptors show a strong transient response to changing stimuli. Their output is passed to bipolar cells, and finally ganglion cells perform a spatial filtering using the well known center-surround antagonism that enhances local contrast differences. As the temporal information detected by the photoreceptors is preserved the ganglion cells show a transient output as well.

Our retina model is based on the model for Y-ON-ganglion cells developed by Gaudiano (1994) , which is extended in two respects: Firstly, both ON- and OFF-ganglion cells are modeled and, secondly, the time course of the transient response pattern of each cell is represented in a simplified way by just two values (see figure 5.2): a maximal transient response v_{tr} when *new* input comes in, and the steady state activity rate v_{st} while this input is constantly present. Here, is assumed that the time scale of the ganglion cells is faster than the change in the input, which is recorded by a camera at 3 frames per second. This requirement results in an upper bound for object speed relative to the time scales used for the neuronal processing.

Neither the LGN nor the cortical layer 4c_α are explicitly modeled as it is assumed that in early ontogenesis no processing relevant for the development of Gestalt principles is performed there. Therefore, the activity of ON- and OFF-ganglion cells provides direct input to the simple cells in layer 4b. It will become clear later that for the purpose of our model it is not necessary to model all aspects of inhibitory neurons in striate cortex in detail. A detailed model of the cortical dynamics mediated by short, middle and long-range corticocortical connections is also not required. Let us explain why: There are a lot of inhibitory interneurons in the cortex that

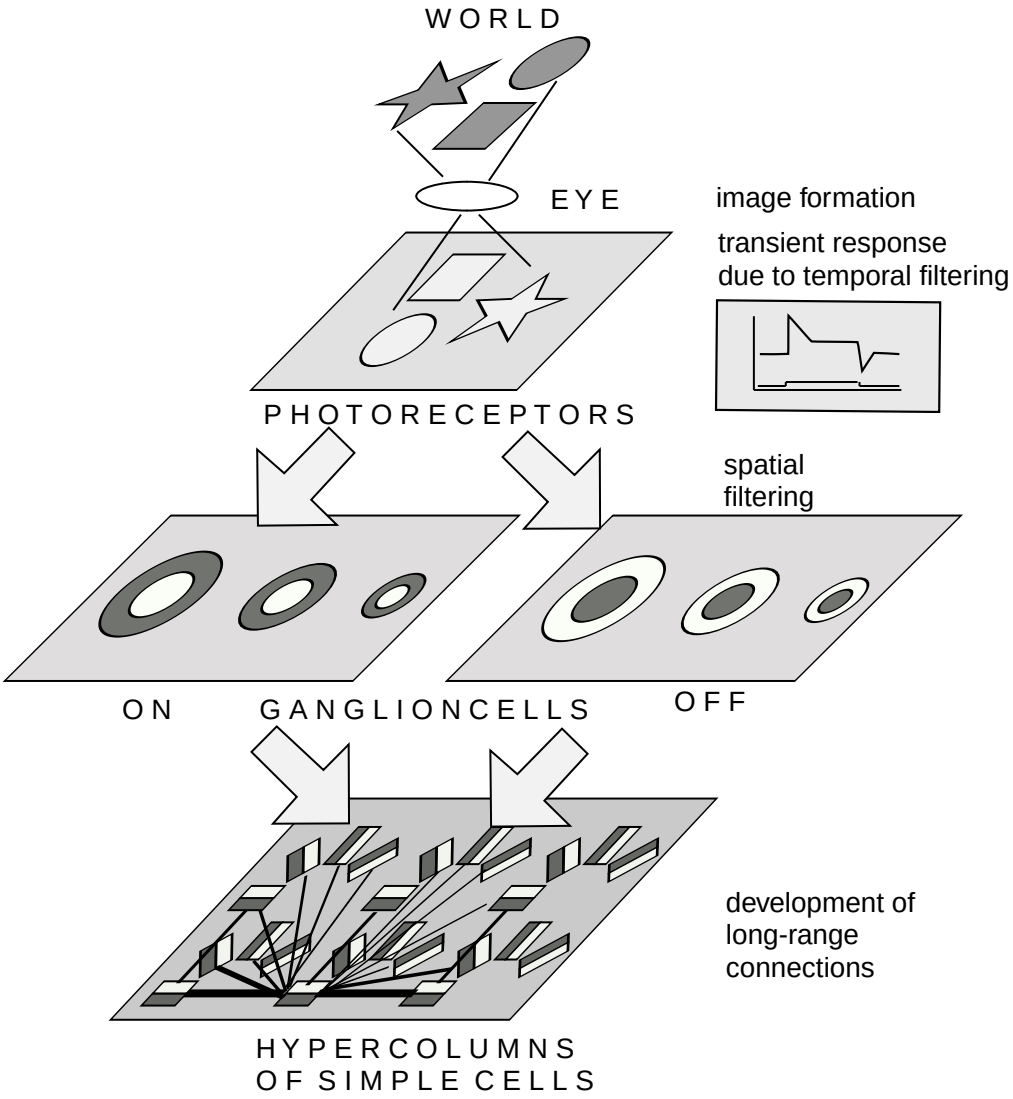


Figure 5.1.: Illustration of the complete model.

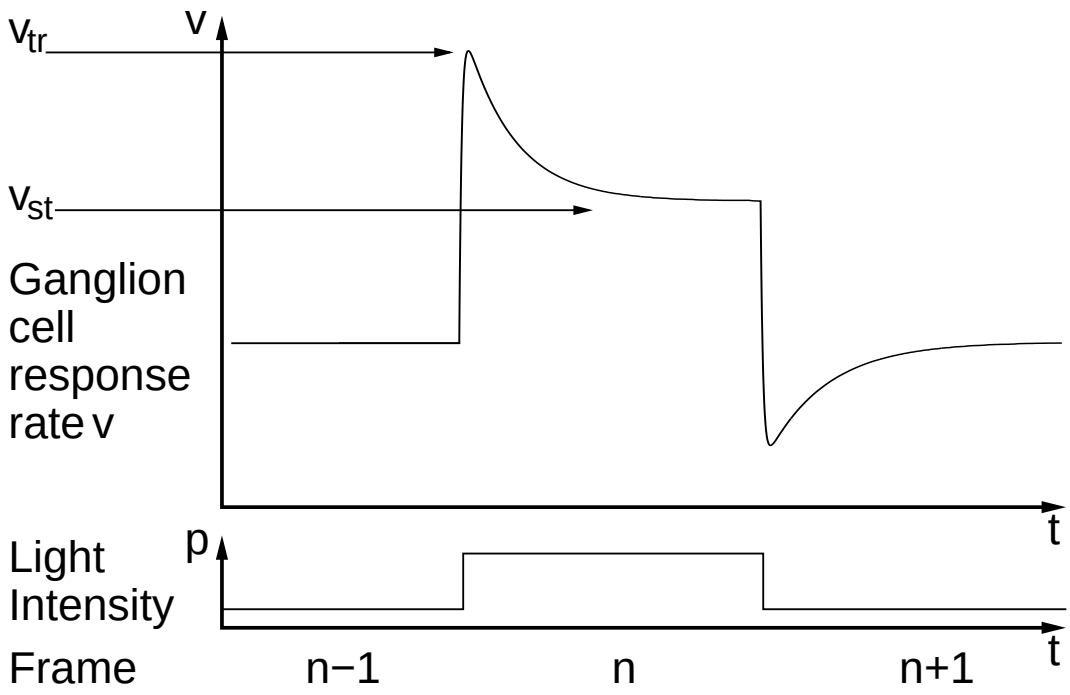


Figure 5.2.: This figure illustrates the computation of the discrete approximations v_{tr} and v_{st} to the continuous transient activity of an ON ganglion cell.

receive afferent input themselves and affect the excitatory pyramidal cells later at least by short-range lateral connections. We assume that the main effect of those inhibitory connections with regard to our model is to avoid an excitatory explosion when input arrives at the cortex, as only afferences are coming in. This assumption is realistic because the high neural noise in neonates (see section 4.4) requires strong global inhibition — stronger than the overall excitation — in the cortex to keep the whole system stable. Furthermore, activity mediated along horizontal connections alone should not be able to trigger a response in a target cell. If inhibitory neurons have no other function than the ones mentioned we can avoid modeling them explicitly by letting only those excitatory cortical cells participate in the structuring of long-range cortical connections that receive strong primary afferent input themselves. Consequently, we do not need to model the cortical dynamics further at this early stage of development as purely intracortical influences at a postsynaptic neuron without strong primary afferent input should not have significant impact on the activity. In the adult, however, there are substantial influences from intracortical long-range connections, and also the neural noise is reduced by a factor of nine.

The modeled simple cells are arranged in hypercolumns, and during the simulations a long-range connection structure between these cells emerges. One cell of each hypercolumn can be connected to any cell located in a 9×9 square surrounding its own hypercolumn. It should be noted that each simple cell in the model in fact represents a local pool of cells that all have similar properties. Therefore, it makes sense to model connections of a model cell to itself.

One of the crucial features of our model for the development of a specific long-range connection structure is that the cortical simple cells in layer 4b show a transient response pattern.

In the results section we will see that this transient cortical response pattern enables the development of an iso-orientation long-range connection structure as it is found in animals (Schmidt et al. (1997)). In the following we describe how we model the transient cortical activity starting with the transients of the ganglion cells. An interesting question, which is beyond the range of this paper is how these responses are produced biologically. For simplicity (and computational tractability) we omit all biological details that could be part of the theoretically possible mechanisms — single cell properties, sustained local inhibition etc. — that enable transient responses. These must be investigated in the animal and by models on a finer scope than ours.

We have found that the success of the model depends much more on the fact that the response is transient than on its precise time course. Therefore, for each time step n between the acquisition of successive video frames the output of an ON- or OFF-ganglion cell is discretized to two values $v_{\mathrm{tr}}(n)$ and $v_{\mathrm{st}}(n)$. Consequently, it is natural to also discretize the primary afferent response a that a simple cell would show if no other (intracortical) influences where present to a transient and a stationary value a_{tr} and a_{st}, respectively. a is computed by a 2-dimensional spatial convolution (denoted by $*$) with the kernels g^{ON} and g^{OFF} representing the synaptic couplings made by ON-afferences and OFF-afferences to the cortical cells.

$$\begin{aligned} a_{\mathrm{tr}}(n) &= v_{\mathrm{tr}}^{ON}(n) * g^{ON} + v_{\mathrm{tr}}^{OFF}(n) * g^{OFF} \\ a_{\mathrm{st}}(n) &= v_{\mathrm{st}}^{ON}(n) * g^{ON} + v_{\mathrm{st}}^{OFF}(n) * g^{OFF} \end{aligned} \tag{5.1}$$

The full details of the cortex model are given in equations (5.25)-(5.27) in section 5.3. Here it is suffices to mention that the functions g^{ON} and g^{OFF} are responsible for the orientation ϕ and the polarity $(+,-)$ of the simple cell receptive fields. The resulting receptive fields for the ϕ_+ cells are shown schematically in the top row and leftmost column of figures 6.1 and 6.2. Regardless of the detailed mechanism that causes the transient nature of the cortical cell response: if a transient response is triggered at frame n then there must be a difference in the primary afferent input of this frame $a_{\mathrm{st}}(n)$ and the primary afferent input the cell received during the previous frame $a_{\mathrm{st}}(n-1)$. Then, it follows from our retina model that there has to be a difference in the values of $a_{\mathrm{tr}}(n)$ and $a_{\mathrm{st}}(n)$ as well, as after the transient over- or undershoot a new steady state $a_{\mathrm{st}}(n)$ will be reached, which cannot be equal to the old one $a_{\mathrm{st}}(n-1)$ because otherwise there would have been no transient response at all.

The real output o of the ganglion cell — the one that can be measured experimentally by counting action potentials and linearly transforming the base rate to zero — can then be approximated as:

$$o(n) = \max(a_{\mathrm{tr}}(n) - a_{\mathrm{st}}(n), 0) \tag{5.2}$$

The response in (5.2) is quantitatively a bit too strong as the relaxation of the afferent input $a_{\mathrm{st}}(n)$ may not be completed in the time between two consecutive frames. However, the important feature for our model is that transient responses are successfully detected by this output function. To point out how crucial the transient nature of cortical responses in layer 4b is for the development of long-range connections we have done additional simulations with simple cells having sustained (o^*) instead of transient responses:

$$o^*(n) = \max(a_{\mathrm{st}}(n) - \Theta, 0)\,, \tag{5.3}$$

where Θ is a constant near the baseline primary afferent activity.

Figure 5.3.: One frame of a typical camera sequence used in the simulations. There is a single moving object in front of the observer and a structured background. The strong diagonal grey tone border in the background is used to illustrate the difference between sustained and transient responses in the learning rule. Such a biased background is a problem for static responses, which tend to learn just that bias. If transient responses are used, the distribution of orientations caused by the moving person is much broader.

A simple Hebbian learning mechanism is used for the adaption of the long-range synaptic strengths:

$$\Delta w_{ij} \quad = \quad \epsilon \, o_i o_j \tag{5.4}$$

For computational efficiency, this learning rule is not applied to single synaptic weights but to ensembles of *equivalent* connections. Two connections are equivalent if their pre- and postsynaptic cells have the same orientation and polarity, and they span the same cortical distance. This effectively leads to a system of connections that is translation invariant by design. Mathematical details and a biological motivation can be found in section 5.3.2.

If the *transient* cortical responses *o* are inserted into equation (5.4) to evaluate the correlation between pre- and postsynaptic cell, respectively, the connection structure shown in figure 6.1 emerges. We will argue in section 6.1 that this is suitable to form the anatomical basis for the Gestalt principle of collinearity.

If the simulations are done with the same visual processing but with the Hebbian learning on the basis of the *sustained* cortical responses (o^*) the resulting connection structure is qualitatively different (figure 6.2). In this case no general principles are learned but the final connection structure reflects properties of the particular visual data used for learning.

5.2. Model of Subcortical Processing

5.2.1. Retina Model

We do not model the development of the retina itself during the postnatal weeks. Instead we extend and modify an existing retina model (Gaudiano, 1994) to compute ON- and OFF-Y-ganglion cell responses. Once this is done we discretize the continuous differential equations in time by just to values for each cell type (ON and OFF). The activity rate v_{st} corresponds to the tonic or steady state part in the continuous model, and the transient rate v_{tr} to an upper bound for the maximum transient or phasic rate of the ganglion cell response (figure 5.2). These discrete approximations of the retina model are essential because their output is used in the model for the cortical layer. To understand how these equations are derived we now go into the details of the continuous retina model. Note that the spatial dependency of the functions used is omitted to improve readability.

5.2.2. The Photoreceptors

Given the light intensity value p_{n} for each pixel of the camera image recorded at time t_{n}, we define a function $p_{\mathrm{In}}(t)$ that is equal to p_{n} for all times t in the interval of $[t_{\mathrm{n}}, t_{\mathrm{n+1}})$. The values of this function $p_{\mathrm{In}}(t)$ are the interval of $\left[p_{\mathrm{In}}^{\mathrm{min}}, p_{\mathrm{In}}^{\mathrm{max}}\right]$, that is given by the camera as $[0, 255]$. We compute the nonlinear photoreceptor response $r(t)$ according to:

$$r(t) \quad = \quad z(t) \; p(t) \quad \text{with} \tag{5.5}$$

$$p(t) \quad = \quad [p_{\mathrm{max}} - p_{\mathrm{min}}] \, p_{\mathrm{In}}(t) \; / \; (p_{\mathrm{In}}^{\mathrm{max}}) \; + \; p_{\mathrm{min}} \, . \tag{5.6}$$

Here the biological fact is taken into account that at extreme light intensities a photoreceptor can perform temporal high-pass filtering. To achieve this the photoreceptor adapts its response

rate under extreme constant light intensity towards its base rate activity by multiplying the visual input $p(t)$ (5.6) with an internal state $z(t)$. The way this internal state is computed makes clear that it is something like a short term memory for light intensity. By this light adaption mechanism a sudden change to a given extreme light intensity level produces a short term activity rate substantially different from the one produced under a constant light intensity of the same extreme magnitude. The photoreceptor response $r(t)$ is computed by using the transformed light intensity $p(t)$. The simple linear rescaling in (5.6) to the interval $[p_{\min}, p_{\max}]$ is necessary, because the increased minimal value allows for dynamic photoreceptor behavior. We have chosen $p_{\min} = 30, p_{\max} = 255$. The chosen value of $p_{\min}$ is not particularly critical but should be well above zero, because otherwise low light intensities in the visual world cannot be modulated by the internal state of the photoreceptor and can therefore not trigger a dynamic photoreceptor response distinguishable from the response to constant low light intensity.

The rate of change dz/dt of the internal parameter depends on the relative intensity $p_{rel}(t)$ (5.8) of the visual stimulus that the photoreceptor receives:

$$dz/dt = F\,[M - z(t)] - H\; p_{\text{rel}}(t)\; z(t) \quad \text{with} \tag{5.7}$$

$$p_{\text{rel}}(t) = (p(t) - p_{\min}) \,/\, (p_{\max} - p_{\min})\;. \tag{5.8}$$

Here F, M and H have the values chosen by Gaudiano (1994) . M is the maximal value of $z(t)$ and F is a gain parameter controlling how fast $z(t)$ approaches M in the absence of light ($p_{\text{rel}}(t) \equiv 0$). On the other hand $Hp_{\text{rel}}(t)$ controls the decay of $z(t)$.

5.2.3. Bipolar and Ganglion Cells:

The next steps in retinal processing are the integration of photoreceptor activity by horizontal cells and the feedforward processing by bipolar cells. We summarize horizontal influences of horizontal and amacrine cells in a later step of the model (equations (5.11,5.12)), and concentrate on the straightforward modeling of bipolar cell responses $b^{+,-}$: It is assumed that one bipolar cell receives input from just one photoreceptor. The value $r_{max} = p_{\max}\, M$ is given by (5.5) as M is the maximal value of the internal state $z(t)$ of a photoreceptor and $p_{\max}$ the maximal light intensity.

$$b^+(t) = r(t) \;\text{ und }\; b^-(t) = r_{\max} - r(t) \tag{5.9}$$

The ganglion cell activity $v(t)$ itself is modeled using the *shunting* equation (5.10) first used by Grossberg and Sperling (1970) . In our particular equation the ganglion activity rate $v(t)$ is limited to the interval of $[0, 1]$ without passive decay:

$$dv(t)/dt = [1 - v(t)]\, u(t) - [v(t)]\, w(t)\,. \tag{5.10}$$

The excitatory $u(t)$ and inhibitory $w(t)$ inputs contribute to the ganglion cell response in a PUSH-PULL way, which means that each bipolar cell contributes to both center and surround mechanisms of the ganglion cell in different quantities (McGuire et al., 1986). For ON-ganglion cells these influences have been modeled by the following equations where we use the Gaussians c (central) and s (surround) with the parameters of the Gaudiano model for Y-ganglion cells extended to two dimensions:

$$u(t) = c * b^+(t) \;+\; s * b^-(t)\,, \tag{5.11}$$

$$w(t) = s * b^+(t) \;+\; c * b^-(t)\,. \tag{5.12}$$

Biologically, these influences come from lateral integration mediated by horizontal and amacrine cells. For OFF-ganglion cells the above equations have been used with $b^+(t)$ and $b^-(t)$ interchanged. In our simulations, the center Gaussian has a value of σ equal to 3.18 pixel and the surround Gaussian of 3.89 pixel.

The analytical solutions of (5.10) for ON- and OFF-cells then are:

$$v^{ON}(t) = E/p_{\max}\ [c * r(t) - s * r(t)] + F_{\mathrm{ON}}\,, \tag{5.13}$$
$$v^{OFF}(t) = -E/p_{\max}\ [c * r(t) - s * r(t)] + F_{\mathrm{OFF}}\,, \tag{5.14}$$

where the following abbreviations have been used for convenience of notation:

$$E := 1/\left(MV_s + MV_c\right)\,, \tag{5.15}$$
$$F_{\mathrm{ON}} := MV_s/\left(MV_s + MV_c\right)\,, \tag{5.16}$$
$$F_{\mathrm{OFF}} := MV_c/\left(MV_s + MV_c\right)\,. \tag{5.17}$$

The numbers used in these definitions are as follows: V_s and V_c are the integrals over the two Gaussians used for modeling center and surround influences in the PUSH-PULL model.

So far, we have presented a continuous model for ON and OFF ganglion cells and one can discuss various aspects of the model like the validity of the used simplifications, e.g., where exactly in the retina the spatial integration takes place, how the temporal filtering is probably done and more. We refer the reader to (Gaudiano, 1994) for these arguments.

As shown in figure 5.2 the continuous time course is now approximated by two values which are computed as follows.

$$r_{\mathrm{tr}} = p_{\mathrm{n}}\, z_{\mathrm{n}-1}\,, \tag{5.18}$$
$$z_{\mathrm{n}} = MF\ /\ (F + Hp_{\mathrm{n}})\,, \tag{5.19}$$
$$r_{\mathrm{st}} = p_{\mathrm{n}}\, z_{\mathrm{n}}\,. \tag{5.20}$$

With these approximations and using the continuous equations for the ganglion cell responses (equations (5.13) and (5.14)) we can approximate the ON- and OFF-ganglion cell activity discretely in time and represent each of them by two values that describe the spatiotemporal properties of the ganglion cell responses:

$$v_{\mathrm{tr}}^{ON} = E/p_{\max}\ [c * r_{\mathrm{tr}} - s * r_{\mathrm{tr}}] + F_{\mathrm{ON}} \tag{5.21}$$
$$v_{\mathrm{st}}^{ON} = E/p_{\max}\ [c * r_{\mathrm{st}} - s * r_{\mathrm{st}}] + F_{\mathrm{ON}} \tag{5.22}$$
$$v_{\mathrm{tr}}^{OFF} = -E/p_{\max}\ [c * r_{\mathrm{tr}} - s * r_{\mathrm{tr}}] + F_{\mathrm{OFF}} \tag{5.23}$$
$$v_{\mathrm{st}}^{OFF} = -E/p_{\max}\ [c * r_{\mathrm{st}} - s * r_{\mathrm{st}}] + F_{\mathrm{OFF}} \tag{5.24}$$

5.3. Cortex Model

How should the cortical layer be modeled? Recall from section 4.4 that after birth 90% of all active cells are of simple type. For that reason we will not consider complex cells here as they probably play only a minor role directly after birth and most likely develop afterwards. Additionally, the development of the long-range connection structure of different kinds of simple

cells after birth probably occur at different speeds. The development of the long-range connection structure between *edge* detector (odd symmetry) cells of all scales is expected to be more robust than the one of *bar* detector (even symmetry) cells immediately after birth. One reason for this is the increased sensitivity to low spatial frequency gratings of the cortical cells (see section 4.4) in the newborn. This phenomenon assigns a special role to the edge detector cells. An edge between two large areas of high and low luminance, respectively, remains an edge for all edge detector cells independent of their scale or spatial frequency. This is not true for example for a bar detector cell as the strength of the response is determined by the preferred spatial frequency and the size of the bar presented. Therefore we restrict our model to simple cells with odd symmetry that are functionally edge detectors.

How can edge detector cells be modeled? In (5.1) we have given an equation for the primary afferent response of a simple cell, and the terms g^{ON} and g^{OFF} are now defined in detail: The most important feature of an edge detector cell is its preferred orientation ϕ, and for each orientation there are two edge detector cells, one with positive (ϕ_+) and one with negative (ϕ_-) polarity. They can be modeled by using the sine part of a Gabor function (5.25) with different signs. In our simulations we use cells with eight different orientations ϕ and do not model other properties of simple cells like selectivity for motion direction. The equations are as follows.

$$g(x, y, \phi) = \exp(-\frac{k^2(x^2 + y^2)}{2\sigma^2}) \cdot \sin\left(kx\cos\phi + ky\sin\phi\right) , \tag{5.25}$$

$$\phi_+ : \quad g^{ON} = \max(+g, 0),\ g^{OFF} = \max(-g, 0) , \tag{5.26}$$

$$\phi_- : \quad g^{ON} = \max(-g, 0),\ g^{OFF} = \max(+g, 0) . \tag{5.27}$$

The parameters are $\sigma < 2$ and $k = 0.5$. A value of σ above two leads to receptive fields with more than two significant ON- or OFF-subfields in contrast to the data about simple cells. Note that the resting activity of a cortical neuron in (5.1) is nonzero despite of the vanishing integral of g in (5.25). The reason is that the resting activity of the retinal ganglion cells is nonzero and there are only afferences to the cortex. Biologically it is not likely that the synaptic strengths of the afferent thalamocortical connections are so finely tuned in the first postnatal weeks as the Gabor-like receptive fields imply. With the high specific connection structure given in (5.26) and (5.27) the effects of the short-range intracortical inhibitory and excitatory connections are implicitly modeled which are the probable basis for sharp orientation tuning (Somers et al., 1995). However, to use parts of a Gabor function is an easy alternative way to implement some form of orientation tuning without the computational cost of modeling the short-range corticocortical excitatory and inhibitory feedback connections.

5.3.1. Cortical Organization

We assume that the retinogeniculate and thalamocortical pathways already exist at birth and that their mapping is already retinotopic (see section 4.4). Simple cells sensitive to low spatial frequencies also exist and their theoretical primary afferent responses are modeled using equation (5.1). The receptive field center positions $\vec{p} \in \mathrm{N}^2$ for the simple cells are placed in the image at grid points with a distance of four pixels in the horizontal and vertical direction. For each of those grid positions a hypercolumn consisting of simple cells of eight different receptive field orientations is modeled. Within one hypercolumn each specific orientation is represented by two cells ϕ_+ and ϕ_- with receptive field properties defined by equations (5.26) or (5.27).

All cells in one hypercolumn have the same receptive field center $\vec{p}_0$ in retinal coordinates and neighboring hypercolumns have different receptive field centers $\vec{p}_k$. Each cell of a hypercolumn establishes connections with all neurons of its own hypercolumn and with all neurons in the neighboring four hypercolumns in each direction of the cortical plane (9×9 neighborhood). Each model cell represents biologically a pool of cells with nearly the same properties and therefore a model cell was allowed to make connections to itself. To avoid border artifacts learning of long-range connections was disabled in the first four hypercolumns at the border of the cortical plane.

5.3.2. Learning Horizontal Connections

We now shift our focus to the *organization* of the long-range cortical connections illustrated at the bottom of figure (5.1) and in more detail in figure (5.4), in which the repeated regular structure is representing a hypercolumn with eight orientations (only four are shown) and two polarities. A small segment of the cortical plane is shown and the dashed arrows in it illustrate the range of the cortical horizontal connections.

To adapt the synaptic weights the difference in primary afferent input (5.2) is used in (5.4) to select only those cells that have a transient cortical response. Equation (5.4) is a Hebbian learning rule. Δw_{ij} is the change of synaptic strength between neuron j and i and ϵ is a general learning factor that controls the impact of each learning step. The choice of ϵ is not critical. To illustrate the effect of transient responses we have also examined a Hebbian learning rule that only uses the sustained (5.3) cortical responses.

$$\Delta w_{ij} \quad = \quad \epsilon \; o_i^* o_j^* \, . \tag{5.28}$$

This corresponds to the classical interpretation of the Hebbian postulate in neural modeling because it is a correlation learning rule that operates directly on the input. All cells that have a primary afferent response above the baseline activity θ of (5.3) may participate in the structuring process.

After the increment of (5.4) or (5.28) is used to update the synaptic strengths (5.29) two additional procedures are incorporated. To avoid artificially high synaptic values a connection strength above one is reset to one by (5.30). To introduce competition between synapses the total synaptic strength for a given ensemble $\mathcal{C}$ is held constant at K by normalizing the weights after each learning step and multiplying them with K (5.31).

$$w_{ij} \quad \leftarrow \quad w_{ij} + \Delta w_{ij} \tag{5.29}$$

$$w_{ij} \quad \leftarrow \quad \min(w_{ij}, 1) \tag{5.30}$$

$$w_{ij} \quad \leftarrow \quad w_{ij} K \; / \; \sum_{\mathcal{C}} w_{ij} \tag{5.31}$$

One of the steps to make the model applicable to natural image data is the introduction of *ensembles* $\mathcal{C}$ of equivalent connections. First we will give a technical description of such an ensemble and then we explain the biological motivation for building those ensembles. All established connections can be characterized by four parameters: The receptive field center position $\vec{p} \in \mathrm{N}^2$ of the presynaptic cell, the spanned distance measured in hypercolumns $\vec{r} \in \mathrm{Z}^2$, orientation and polarity of the presynaptic simple cell $\phi^{\mathrm{pre}}_{+,-}$ and the orientation and polarity of

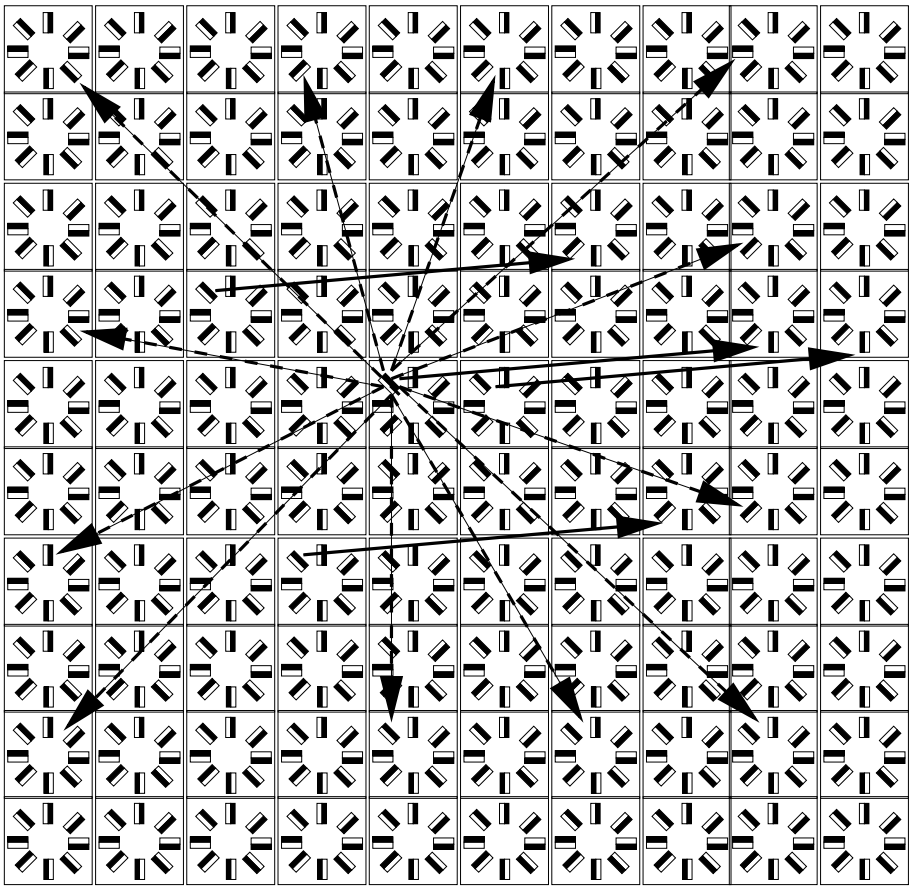


Figure 5.4.: Illustration of the cortical horizontal connection scheme. A small segment of the cortical plane is shown to illustrate the range of the long-range connections (indicated by the dashed arrows) and some connections of an ensemble of equivalent connections (indicated by the solid arrows and explained in the text). The repeated regular structure represents a hypercolumn with 8 orientations (just four are shown) and two polarities. Please refer to the section 5.3.1 for a detailed explanation.

the postsynaptic simple cell $\phi^{\text{post}}_{+,-}$. Mathematically, one can build the equivalence classes from the relation that two connections — and with them their synaptic weights — are equivalent if they have the same $\vec{r}$, $\phi^{\text{pre}}_{+,-}$ and $\phi^{\text{post}}_{+,-}$ but possibly different centers.

By doing this we have introduced translational invariance of the synapses that connect the same pre- and the same postsynaptic cell types over the same distance of hypercolumns, which is illustrated for a few connections by the solid arrows in (5.4). As we have two polarities and eight different types of cells as pre- and postsynaptic cells and nine times nine different $\vec{r}$ we have 1296 ensembles of connections for each presynaptic cell type of the reference hypercolumn and a total of 20736 ensembles for this hypercolumn as a whole. As an example, the four solid arrows in figure 5.4 are equivalent and are forced to have identical weights.

What is the biological foundation for building these ensembles? Consider a newborn with a moving object in front of it. The newborn will gaze in one direction and the image of the objects moves over the retina. Then on a larger time scale the newborn will shift its head trying to follow — not very successful because of the low tracking accuracy after birth (Hofsten et al., 1998; Piaget, 1936) — the stimulus and gaze again in the new direction. This happens several times until the newborn has lost the object. If we now think of the visual input the newborn receives in terms of object features projected on the retina we see that the same features are shifted on the retina because of the object motion and because of the more or less randomly distributed eye or head saccades of the newborn. One could argue now that this should mainly affect horizontally neighboring cells and not vertically neighboring cells because most movements are on the horizontal surface. But when the newborn is gazing in one direction and then moves his head or eyes to gaze in another direction it is very unlikely that this can be done without a vertical shift given the low accuracy of tracking movements in the newborn. The structuring of long-range connection strengths in the cortical region that corresponds to the retinal area covered by the object will of course average in time over all presented stimuli and this average should be the same for equivalent connections because the object features seen have been the same on average. The result should be connection strengths of roughly the same magnitude for equivalent connections. Therefore, we can just model one synapse for each ensemble of connections and reduce the amount of synapses drastically.

5.4. Input data

We have applied the model to two movies of a person moving and waving his arms in front of a background from a seminar room. The backgrounds showed different biases. The sequence "moving.mpg" consists of 199 frames, and the background contains a diagonal rectangle.

In the course of paper revision we also applied the model to the sequence "fw_carsten.mpg", which is in color and has a larger resolution, because it has been collected for a different experiment. This movie consists of 100 frames and shows no strong diagonals in the background. Sampling has been adjusted and color ignored. The results for the transient responses are very similar to the ones for the other sequence. The ones for the sustained responses reflect the different background bias.

To clarify the relationship to static natural images we have applied the model to movies made out of 60 frames from a texture database from MIT[1] and one with 98 frames from a scene database by British Telecom [2].

All four sequences as well as the respective learning processes can be retrieved from [3].

[1] http://www-white.media.mit.edu/vismod/imagery/VisionTexture/
[2] ftp://ftp.vislist.com/IMAGERY/BT_scenes/
[3] ftp://ftp.neuroinformatik.ruhr-uni-bochum.de/pub/pictures/gestalt/

5. A Model for the Early Development of Horizontal Connections.

6. Development of Horizontal Connections

6.1. Results for Transient and Sustained Simple Cell Responses

The simulations have been performed with sequences of camera images like the example shown in figure 5.3. A typical sequence consisted of 100 - 200 frames and showed a person moving in front of a camera and performing some arm movements. Note that in the whole sequence (one frame is show in figure 5.3) the person was the only moving object and that there is a strong diagonal grey tone border in the background.

The learned long-range connection structures for the transient (5.2) and the sustained (5.3) cortical responses are illustrated in figures 6.1 and 6.2, respectively. The shown cells all have the same polarity and vary only in orientation. The results are very similar for the cells of the other polarity ϕ_-, while cross connections between different polarities have not been examined in detail.

I now interpret the results shown in figure 6.1 that have been obtained when the transient cortical responses (5.2) have been used. In detail the results are:

- The size of the long-range connection structure far exceeds the size of the classical receptive field.
- The strongest connections are of course established from the reference hypercolumn (central array positions) to itself. They connect a pool of the identical presynaptic cell types with themselves (iso-orientation).
- There are additionally strong connections in the 3x3 neighborhood of the reference hypercolumn to iso-oriented cells.
- The connections from the (central) reference hypercolumn to other hypercolumns are made to cells with a similar orientation of pre- and postsynaptic receptive field, respectively. Looking at these connections one can see that the direction of the strongest connections corresponds to the orientation in the receptive fields (collinearity). Therefore, the receptive field is somewhat *extended* by these long-range connections.
- The connectivity pattern diminishes in strength with the difference in orientation of pre- and postsynaptic receptive fields (curvilinearity).

This shows that the use of transient responses for Hebbian learning can lead in an efficient and robust way to a connection structure which is suited to form the anatomical basis for the Gestalt

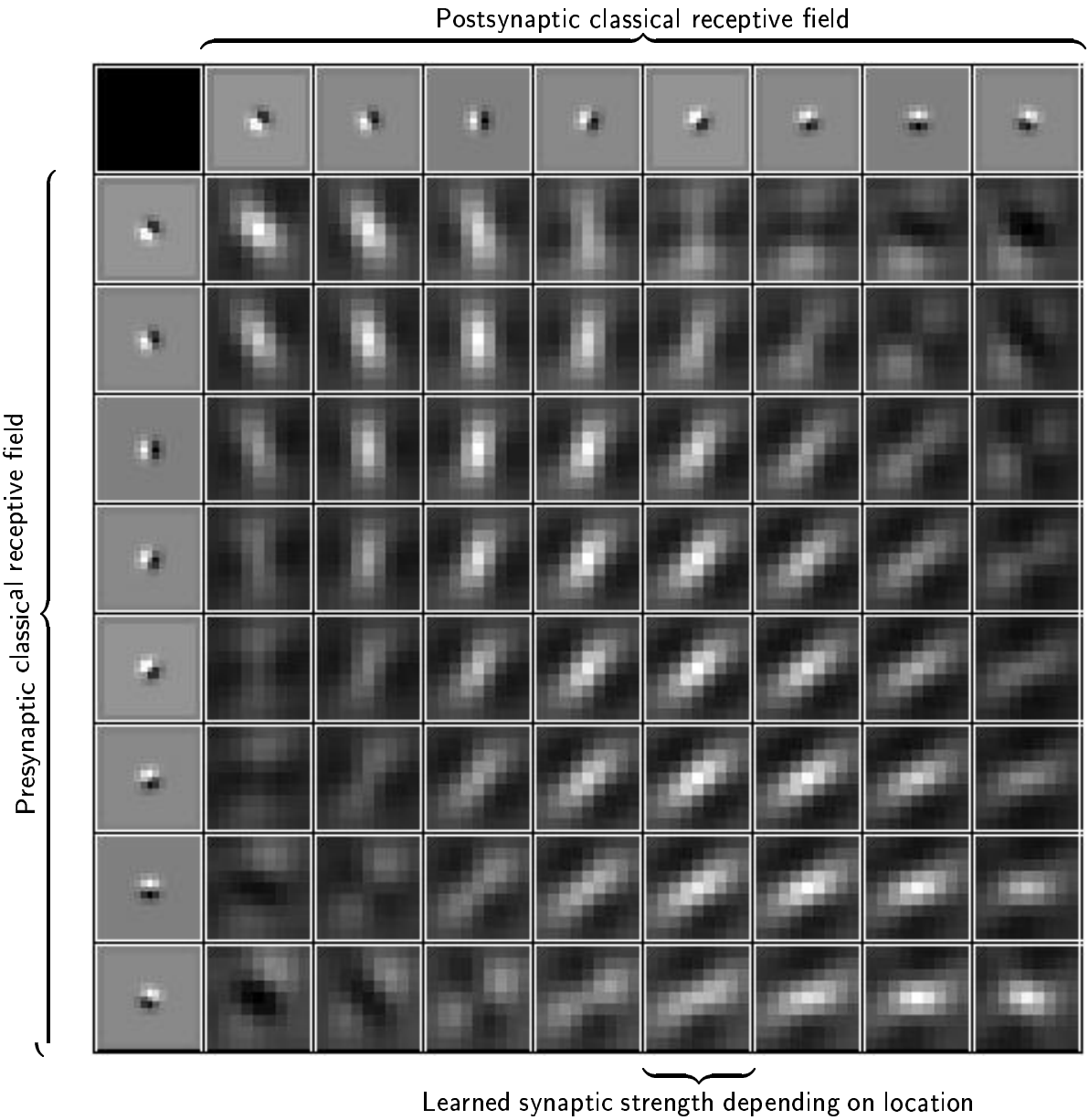


Figure 6.1.: Learned synaptic strengths. The leftmost column shows the classical receptive field shapes ϕ_+^{pre} of presynaptic simple cells, the top row those of the postsynaptic cells (ϕ_+^{post}). The other squares show the spatial distribution of synaptic strengths of one presynaptic cell to a 9×9 patch of postsynaptic cells of constant orientation selectivity. Both types of squares use the same scale in retinal coordinates. The weights are coded in grey scale with white corresponding to the highest weight. The central position of each 9×9 array corresponds to a connection between pre- and postsynaptic cells in the same hypercolumn. The weight distribution shown has been learned on the basis of the transient cortical responses defined in (5.2) with a Hebbian learning rule (5.4). The influence of cortical connections far exceeds the size of the classical receptive fields. Furthermore, the connections that are established prominently connect simple cells with nearly the same orientation in hypercolumns that in retinal coordinates refer to points lying in the direction of their preferred orientation. Thus, it supports collinearity and curvilinearity.

principle of collinearity and even curvilinearity (Field et al., 1993; Hess and Field, 1999; Guy and Medioni, 1996). The importance of transient responses is elucidated by comparison with the results from the same learning rule applied to the sustained responses shown in figure 6.2 caused by the same stimuli. The resulting horizontal connection structure is qualitatively different from the one in figure 6.1. The most significant long-range connections are established to the postsynaptic cells in column 6 of figure 6.2. These show a strong response to the tilted edge in the background of the sequence. Furthermore, almost all long-range connections are in the direction of the that edge.

To a lesser extent the same holds for the connection strengths for the neighboring four columns (4-8) of column 6 and just the presynaptic cells with an orientation difference of 90 degrees (shown in row 2 and the last row of figure 6.2) to the grey tone border-cells show almost no developed connection structure. One can say that the learned connection structure is dominated by the strong grey tone border that is an accidental property of the background. As the grey tone edge is constantly present in the image, cells with an orientation similar to the grey tone border-cells are – according to their tuning curve – active as well. Therefore, although their presynaptic cell activation may be relatively weak their connection strength to the grey tone border-cells increases with each learning step. One could argue that this is a problem of the threshold Θ used in (5.3), and that an increased threshold should filter out the weak responses to borders, so that just the presynaptic border cells connect to the postsynaptic border cells leaving the rest mainly unchanged. This procedure could work for any particular image as probably one can find a threshold that filters out the important borders and suppresses the weak responses for that particular image. However, given the variations in natural images, this threshold must be changed from image to image, because a weak response to a border in one image may be of the same magnitude as the response to the strongest border in another image. One could think that a kind of adaptive threshold that decides about the presence of a border taking its strength in relation to the maximum border strength found in the image could be a solution. Related to this is the idea to normalize the maximal responses in the image. These approaches have the additional disadvantage that even a 'noise' image would participate in the learning process to the same degree as an image with very strong borders.

The results show that the inclusion of transient responses in the model overcomes these conceptual problems by using information from two different cues, the grey tone and the occurring change in subsequent images. Therefore, only moving edges in the image take part in the learning process while static ones have no influence. As it was not possible to learn collinearity with sustained cortical responses out of the same amount of biased image data that was sufficient for transient ones, one can say that using transient responses is an efficient and robust way to do so. One could argue now that with different kinds of backgrounds presented or with a large variety of directions present in one background the disturbing influences will eventually average out, and in the end the same connection structure as with transient response could emerge. With a well balanced input or at least a very large input of different visual scenes this may be possible. It would, however, be a considerable risk for the infants if early development depends critically on this condition. It may also be argued that the advantage of transient over sustained responses is only a consequence of the orientation bias in the background.

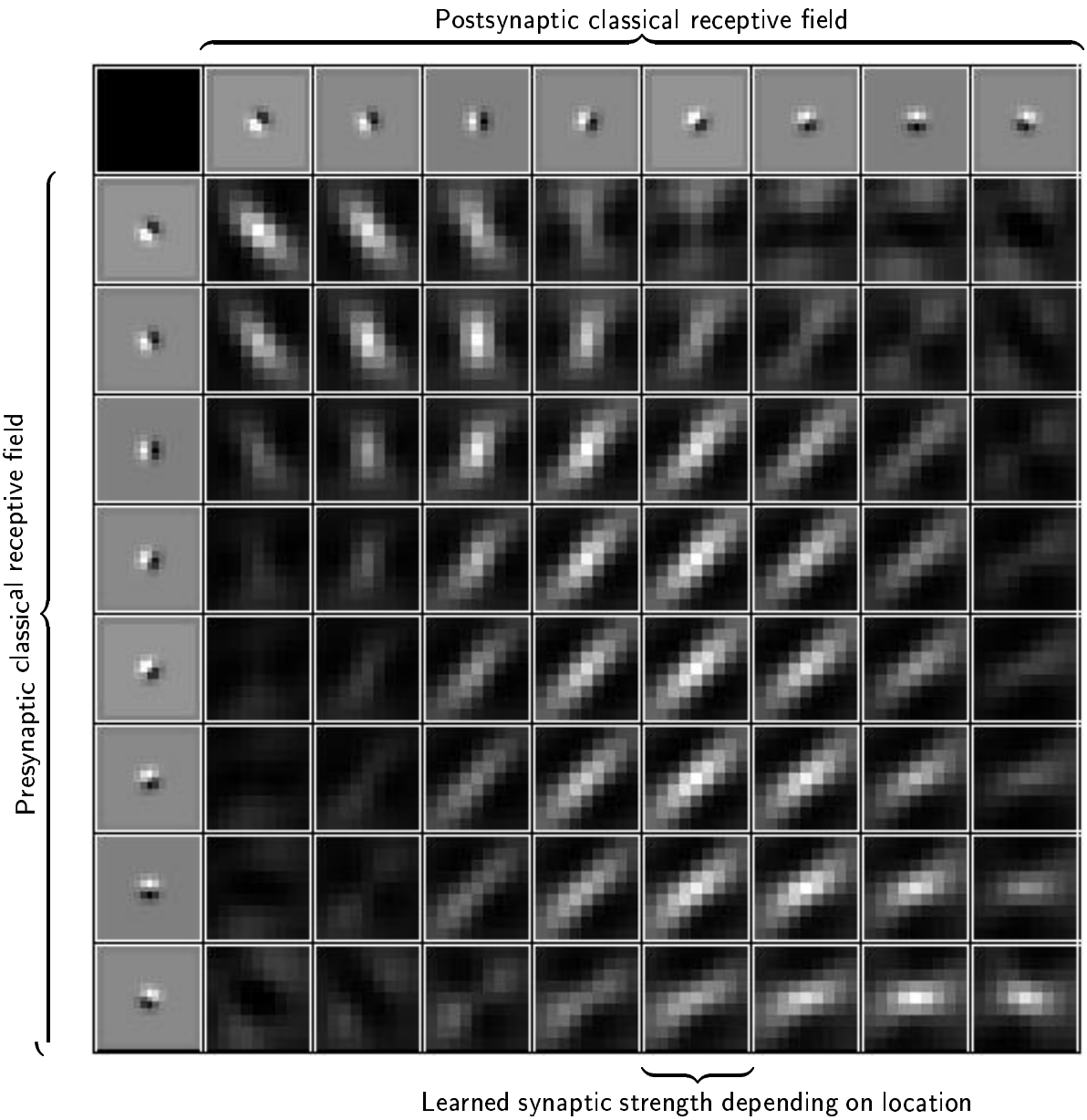


Figure 6.2.: This figure shows long-range cortical synaptic strengths which develop if *sustained* cortical responses (5.3) are used in the Hebbian learning rule (5.28). For an explanation of the display arrangement refer to figure 6.1. The influence of the strong grey tone diagonal in the background of the used images can easily be detected in most of the synaptic connection fields. This example shows how the arbitrary background dominates the learning process of long-range connections when sustained cortical responses are used.

6.2. Application of Sustained Responses to a Database of Natural Images.

In order to clarify the two previous points the model has been applied to standard image collections, which are concatenated into sequences. In this case, transient responses are pointless, because there is no real movement. Learning from the sustained responses also yields a biased connection structure, which supports collinearity in the horizontal and vertical orientations but hardly in the oblique ones. See figure 6.3 for the results on the database available from British Telecom. More stimulus sets together with the learning results are available from the web site and described in section 5.4.

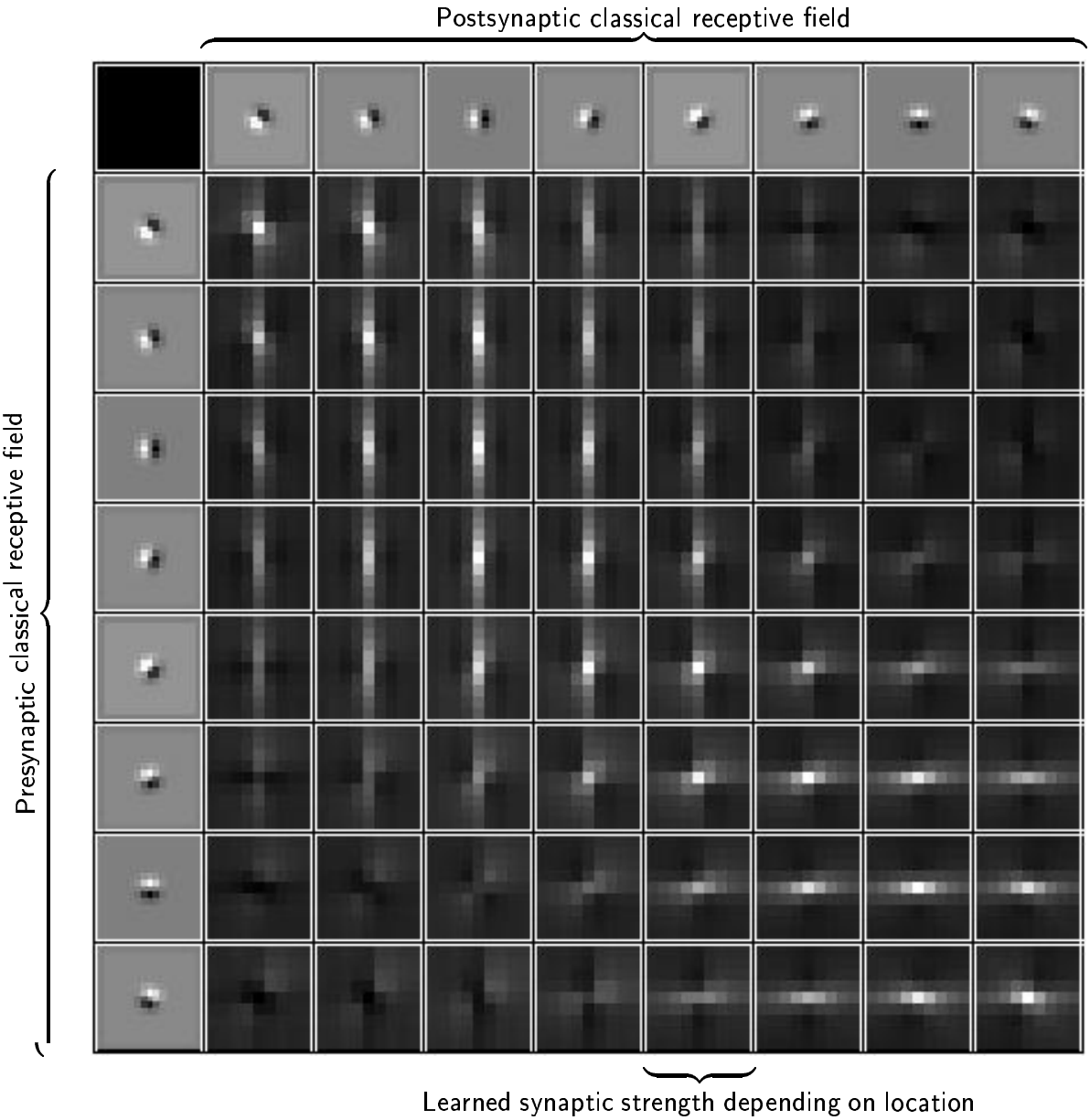


Figure 6.3.: This figure shows long-range cortical synaptic strengths learned from a sequence of static natural images. Transient responses make no sense in this case, so the *sustained* cortical responses (5.3) are used in the Hebbian learning rule (5.28). For an explanation of the display arrangement refer to figure 6.1.

7. Discussion

7.1. Interpretation of the Results

The start was the assumption that there is a hierarchy of Gestalt principles and that the principle of common fate is more fundamental than the one of good continuation. Technically speaking this can be reformulated as saying that motion is a primary cue for the task of segmenting objects from the background. Closed boundaries (as supported by collinearity and curvilinearity) provide a secondary cue that can be learned using the primary one. In this situation, an important strategy is the distinction between observer motion and object motion. This information is probably detectable by a newborn but it is very unlikely that it can already be used accurately on a cellular level, although there is evidence that this is possible even for four months old neonates (Kellman et al., 1987) and in the adult (Leopold and Logothetis, 1998). How can object motion be detected by newborns? A very easy way for the organism to achieve this distinction is to gaze in one direction for a considerable amount of time (Piaget, 1936). While the direction of gaze is fixed changes of illumination on the retina cannot be caused by eye movements or observer motion. Consequently, the structuring process in the cortical model is strongly dependent on the feature constellations resulting from moving objects. Using the latter it was shown (figure 5.3) that it is possible to learn, e.g., the Gestalt principle of collinearity, or more precisely the anatomical structure of long-range connections in primary visual cortex which probably implements this Gestalt principle and was found experimentally by Schmidt et al. (1997). They showed that cells with an orientation preference in area 17 of the cat are linked preferentially to iso-oriented cells, a result which was reproduced by the model. Furthermore, the coupling strength diminishes with the difference in preferred orientation of pre- and postsynaptic cell.

From a technical point of view the system has learned by experience something similar to an association field (Field et al., 1993; Hess and Field, 1999), projection field (Grossberg and Williamson, 2001) or extension field (Guy and Medioni, 1996). These are well known to aid the concept of *collinearity* and *curvilinearity* in technical computer vision algorithms and biological models. Actually, the learned arrangement of synaptic strengths may provide means of improving the extension field (Guy and Medioni, 1996) algorithm, as strong connections between cells with similar orientation are established in all directly neighboring hypercolumns, even if the hypercolumn lies in a spatial direction different from the pre- and postsynaptic orientation. Another lesson learned is that one presynaptic cell type should not only connect to one specific postsynaptic cell type in the target hypercolumn, but with a smaller synaptic weight to a range of types with similar orientation in the same target column.

7.2. Relation of the Results to Other Models

Recently, two models have been presented that address the ontogenesis of Gestalt principles. In (Grossberg and Williamson, 2001) a very detailed perceptual model emerges from learning. It is very remarkable how well the performance of this model matches to neurophysiological measurements. In (Choe, 2001) PGLISSOM, a variant of the self-organizing map, leads to lateral connectivities similar to the ones shown in figure 6.1 after learning from patches of elongated Gaussians. The model differs from those in the respect that here only horizontal connections are considered and it is shown that these can be learned from real camera images. Using real sensor data complicates things considerably, and consequently other details regarding, e.g., stability against contrast changes, have not been addressed. However, one could show that the complexity of data required for learning can be taken from actual motion. It is clear that these connections are only one aspect of a complete system, but it is believed to be worthwhile to study this isolated subsystem.

There are several possible reasons for the result that the desired learning effect could be achieved by using the transient rather than the sustained responses. The first is the biased background. The moving object is much more likely to provide the rich variety of oriented edges required for learning. The transient responses do not react to the static background and therefore that bias cannot influence the connection structure. This also suggests that object motion is preferable to observer motion. Pure observer motion against a static background would learn exactly the background biases, and the same holds for saccadic eye movements. Furthermore, the common motion of object edges gives strong hints that these edges belong together — this could not be achieved if the whole background would move consistently. In order to learn from sustained responses, the background would have to vary a lot in order for the biases of individual backgrounds to average out.

Using only one sequence may be regarded as a weakness of the system, and of course it would be much too little data to cover the environmental properties. This problem is greatly alleviated by assuming translation invariance and therefore employing massive weight sharing. Keeping this in mind, the results indicate that concentrating on the moving parts of a scene provides excellent preprocessing for learning of collinearity. Actually, the fact that one sequence is sufficient clearly demonstrates the power of the preprocessing to select the data relevant for learning. Also, the number of images in the collections is clearly rather small, but I tried to keep about as many as I had movie frames for a fair comparison.

It may be argued that a biased background is an unrealistic assumption in the visual world of ambulatory system. However, there is considerable orientation bias in collections of natural images, at least in the environment given by the campus of Duke University (Coppola et al., 1998). The results indicate that moving objects yield a more even distribution of orientations, although we have not studied that systematically. The assumption that persons moving around can provide a major source of data for visual learning of young infants seems safe. It may also be argued that 100 static images are too few to learn. However, even large image sets would show the bias described in (Coppola et al., 1998). Learning from static images imprints this bias into the connection structure, as can be seen in figure 6.3. At least the experiments show that learning is faster when based on the transient responses to image sequences showing real motion.

Regarding the biological relevance of the model, many details have been omitted and many

simplifications been made in order to achieve a computationally tractable size. However, good results have been reached with a combination of basic mechanisms, namely spatiotemporal retinal filtering, topology preservation in the cortical map, the transient nature of cortical cell responses, and Hebbian learning. It is a well established fact that all those single building blocks of the model exist in the brain, so the complete model should be regarded as a valid approximation to one of the processes that organize perception during early development.

7.3. Learning From Natural Images

The problem to learn relevant feature constellations from natural images is known at least since the models of Marr (1982) were introduced. I think that the main reason for the difficulties faced by the approach is the concentration on feature constellations that are present in the *whole* image. Due to the nature of Hebbian learning (or other second order correlation rules) the useful second order feature constellations are much harder to detect in the whole image than in the part of the image representing a single object. Of course, usefulness is not a physical entity, but arises from the natural desire of living creatures to be able to distinguish objects for various purposes like grabbing an object, escape, ingestion etc., which in turn yields considerable evolutionary advantage.

The problem probably gets harder the more complex the features become, because of their diminishing statistical significance in whole images of natural scenes. If one considers features like collinearity, vertices, or closed boundaries, which define a geometric object, or combinations of closed boundaries, which are listed here in order of their assumed complexity, then the statistical significance to find those features in whole images probably not only diminishes with rising complexity but the more complex features are probably not encountered often enough to make a difference statistically. And even if there is a small significance for those complex features it would take a long time and a lot of different scenes to learn them using a statistical algorithm. For the case of collinearity Krüger (1998) was able to show that collinearity and short-range parallelism are statistically significant features of natural images if the set of images examined is large and varied enough. Geisler et al. (2001) link this to the psychophysical performance of contour grouping and conclude that this must be due to an underlying neuronal structure. The system shows that the necessary variation can be derived from a single scene with one object moving across it for long enough that the movement covers all image locations. Both studies are opposite extremes, the reality for a newborn neither consists of snapshots of many possible scenes nor a single moving object. Both together show that the neural circuits underlying the collinearity Gestalt principle can be learned from natural input.

An important aspect pointed out by Geisler et al. (2001) and Simoncelli and Olshausen (2001) is that simple linear correlations are not sufficient to extract interesting statistics from natural data. Hebbian learning, however, relies on linear correlations. In the case, it has been applied to nonlinearly preprocessed data, so there is no contradiction here. Geisler et al. (2001) also find that curvilinearity cannot be learned from simple cooccurrence but require Bayesian co-occurrence statistics. Again, this is not a contradiction due to the nonlinearities in the model, which are motivated biologically rather than statistically.

Comparison of the results to the ones from Hoyer and Hyvärinen (2002) is made difficult by the fact that the assumptions about the underlying network are different. Their model relies

on a feedforward structure for contour coding, while the emphasis is on horizontal connections. It seems that horizontal connections would hurt the statistical independence of the cells they connect, so the model does not map naturally onto the ICA concept. The biological evidence is certainly too sparse to make a decision in favor of one of these assumptions. This is clearly a point which requires further analysis on the modeling as well as on the biological side.

Further technical studies (Pötzsch, 1999) indicate that feature constellations of higher complexity, e.g., vertices that seem to play an important part in object recognition cannot be learned by simple correlation learning rules that operate on the features of the whole image. As indicated above these useful features are probably not statistically significant features of natural images. If this is the case complex features can only be learned from natural images if there is purposeful behavior that carefully selects the data worthwhile to be learned. Concentrating on moving objects seems to be a good strategy even in the absence of a tracking mechanism. It can be conjectured that more sophisticated mechanisms like head and eye saccades can boost learning further. Reinagel and Zador (1999) show that effect on learning image statistics, therefore a positive effect on learning Gestalt rules may be expected.

List of Figures

LIST OF FIGURES

8. Bibliography

Abbasi, S. and Mokhtarian, F. (2001). Affine-similar shape retrieval: Application to multi-view 3-d object recognition. *IEEE TPAMI*, 10(1):131–139.

Albus, K. and Wolf, W. (1984). Early post-natal development of neuronal function in the kitten's visual cortex: a laminar analysis. *Journal of Physiology*, 348:153–85.

Alferez, R. and Wang, Y.-F. (1999). Geometric and illumination invariants for object recognition. *IEEE TPAMI*, 21(6):505–536.

Arbter, K., Snyder, W., Burkhardt, H., and Hirzinger, G. (1990). Application of affine-invariant fourier descriptors to recognition of 3-d objects. *IEEE TPAMI*, 12(7):640–647.

Arkin, E. M., Chew, L. P., Huttenlocher, D. P., Kedem, K., and Mitchell, J. S. B. (1991). An efficiently computable metric for comparing polygonal shapes. *IEEE TPAMI*, 13(3):209–216.

Atkinson, J. and Braddick, O. (1992). Visual segmentation of oriented textures by infants. *Behav Brain Res*, 49(1):123–131.

Attneave, F. (1954). Some informational aspects of visual perception. *Psychological Review*, 61:183–193.

Avrithis, Y., Xirouhakis, Y., and Kollias, S. (2001). Affine-invariant curve normalization for object shape representation, classification, and retrieval. *Machine Vision and Applications*, 13:80–94.

Barlow, H. (1961). Possible principles underlying the transformation of sensory messages. In Rosenblith, W., editor, *Sensory Communication*, pages 217–234. MIT Press.

Barten, S. et al. (1971). Individual differences in visual pursuit behavior of neonates. *Child Development*, 42:313–319.

Bell, A. J. and Sejnowski, T. J. (1997). The "independent component" of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338.

Bell, B. and Pau, L. F. (1990). Contour tracking and corner detection in a logic programming environment. *IEEE TPAMI*, 12(9):913–917.

Birch, E. (1985). Infant interocular acuity differences and binocular vision. *Vision Research*, 25(4):571–576.

Birch, E. and Salomão, S. (1998). Infant random dot stereoacuity cards. *J Pediatr Ophthalmol Strabismus*, 35(2):86–90.

Born, C. and Völpel, B. (1991). Grouping bits to objects. Technical report, Institut für Neuroinformatik, Ruhruniversität Bochum.

Bosking, W., Zhang, Y., Schofield, B., and Fitzpatrick, D. (1997). Orientation selectivity and the arrangement of horizontal connections in tree shrew striate cortex. *Journal of Neuroscience*, 17(6):2112–2127.

Braddick, O. and Atkinson, J. (1979). A photorefractive study of infant accommodation. *Vision Research*, 19:1319–1330.

Breymann, U. (1996). *Die C++ Standard Template Library*. Addison Wesley.

Burkhalter, A., Bernardo, K. L., and Chrles, V. (1993). Development of local circuits in human visual cortex. *The Journal of Neuroscience*, 13(5):1916–1931.

Callaway, E. M. (1998). Prenatal development of layer-specific local circuits in primary visual cortex of the macaque monkey. *The Journal of Neuroscience*, 18(4):1505–1527.

Chien, C.-H. and Aggarwal, J. K. (1989). Model construction and shape recognition from occluding contours. *IEEE-TPAMI*, 11(4):372–389.

Choe, Y. (2001). *Perceptual Grouping in a Self-Organizing Map of Spiking Neurons*. PhD thesis, University of Texas at Austin. http://www.cs.tamu.edu/faculty/choe/ftp/choe.diss.pdf.

Chuang, G. C.-H. and Kuo, J. C.-C. (1996). Wavelet descriptor of planar curves: Theory and applications. *IEEE Trans Image Proc.*, 5(1):56–70.

Cohen, F. and Wang, J.-Y. (1994). Part i: Modeling image curves using invariant 3-d object curve models — a path to 3-d recognition and shape estimation from image contours. *IEEE TPAMI*, 16(1):1–12.

Coppola, D. M., Purves, H. R., McCoy, A. N., and Purves, D. (1998). The distribution of oriented contours in the real world. *PNAS*, 95:4002–4006.

Courage, M. and Adams, R. (1996). Infant peripheral vision: the development of monocular visual acuity in the first 3 months of postnatal life. *Vision Research*, 36(8):1207–15.

De Boor, C. (1972). On calculating with b-splines. *J. Approximation Theory*, 6:50–62.

De Campos, T. E., Feris, R., and Cesar, R. (2000). Improved face vs. non-face discrimination using fourier descriptors through feature selection. In *Proceedings of the XIII Brazilan Symposium on Computer Graphics and Image Processing (SIBGRAP00)*.

de Monasterio, F. M. and Schein, S. J. (1982). Spectral bandwidths of color-opponent cells of geniculocortical pathway of macaque monkeys. *J Neurophysiol*, 47:214–224.

Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numerische Math.*, 1:269–271.

Field, D. J., Hayes, A., and Hess, R. F. (1993). Contour integration by the human visual system: Evidence for local "association field". *Vision Res.*, 33(2):173–193.

Fitzpatrick, D. (1997). The functional organization of local circuits in visual cortex: Insights from the study of tree shrew striate cortex. *Cerbral Cortex*, 385(6):535–538.

Gaudiano, P. (1994). Simulations of x and y retinal ganglion cell behavior with a nonlinear push-pull model of spatiotemporal retinal processing. *Vision Research*, 34:1767–1784.

Geisler, W., Perry, J., Super, B., and Gallogly, D. (2001). Edge co-occurrence in natural images predicts contour grouping performance. *Vision Research*, 41(6):711–724.

Gorman, J. W., Mitchell, R. O., and Kuhl, F. P. (1988). Partial shape recognition using dynamic programming. *IEEE TPAMI*, 10(2):257 – 266.

Grossberg, S. (1970). Neural pattern discrimination. *Journal of Theoretical Biology*, 27:291–337.

Grossberg, S. and Mingolla, E. (1985). Neural dynamics for perceptual grouping: textures, boundaries, and emergent segmentation. *Perception and Psychophysics*, 38:141–171.

Grossberg, S. and Williamson, J. (2001). A neural model of how horizontal and interlaminar connections of visual cortex develop into adult circuits that carry out perceptual grouping and learning. *Cerebral Cortex*, 11(1):37–58.

Guy, M. and Medioni, G. (1996). Inferring global perceptual contours from local features. *International Journal of Computer Vision*, 20:113–133.

Hainline, L. et al. (1992). Development of accomodation and convergence in infancy. *Behavioural Brain Research*, 49:33–50. Special Issue: Normal and Abnormal Visual Development in Infants and Children.

Heijmanns, H. and Tuzikov, A. (1998). Similarity and symmetry measures for convex shapes using minkowski addition. *IEEE TPAMI*, 20(9):980–993.

Hess, R. and Field, D. (1999). Integration of contours: new insights. *Trends in Cognitive Sciences*, 3(12):480–486.

Hofsten, C., Vishton, P., Spelke, E., Feng, Q., and Rosander, K. (1998). Predictive action in infancy: tracking and reaching for moving objects. *Cognition*, 67(3):255–285.

Hoyer, P. O. and Hyvärinen, A. (2002). A multilayer sparse coding network learns contour coding from natural images. *Vision Research*, 42(12):1593–1605.

http://www.organic computing.com/ (2000).

Isaac, J., Crair, M., Nicoll, R., and Malenka, R. (1997). Silent synapses during development of thalamocortical inputs. *Neuron*, 18(2):269–80.

Jeannin, S. and Bober, M. (Mar. 1999). Description of core experiments for mpeg-7 motion/shape. Technical report, Technical Report ISO/IEC JTC 1/SC 29/WG 11 MPEG99/N2690, MPEG-7,Seoul.

Kameda, Y. (1993). Three dimentional pose estimation of an articulated object from its silhouette image. In *Proc. of Asian Conference on Computer Vision*, pages 612–615.

Kartikeyan, B. and Sarkar, A. (1989). Shape description by time series. *IEEE TPAMI*, 11(9):977–984.

Kastrup, D. (1997). Grouping bits to objects revisted. Technical report, Institut für Neuroinformatik, Ruhruniversität Bochum.

Katz, L. C. and Callaway, E. M. (1992). Development of local circuits in mammalian visual cortex. *Annual Review of Neuroscience*, 15:31–56.

Kellman, P. J., Gleitmann, H., and Spelke, E. (1987). Object and observer motion in the perception of objects by infants. *J Exp Psychol Hum Percept Perform*, 13(4):586–593.

Kellman, P. J. and Short, K. (1987). Development of three-dimensional form perception. *J Exp Psychol Hum Percept Perform*, 13(4):545–557.

Kellman, P. J., Spelke, E., and Short, K. (1986). Infant perception of object unity from translatory motion in depth and vertical translation. *Child Development*, 57(1):72–86.

Kindratenko, V. and Van Espen, P. J. M. (1996). Classification of irregularly shaped microobjects using complex fourier descriptors. In *Proceedings of the 1996 Conference on Pattern Recognition (ICPR '96)*, pages 285–289.

Koffka, K. (1935). *Principles of Gestalt Psychology*. , London.

Korn, P., Sidiropoulos, N., Faloutsos, C., Siegel, E., and Protopapas, Z. (1998). Fast and effective retrieval of medical tumor shapes. *IEEE Trans. On Knowledge and Data Engeneering*, 10(6):889–904.

Kovacs, I. (2000). Human development of perceptual organization. *Vision Research*, 40(12):1301–1310.

Krüger, N. (1998). Collinearity and parallelism are statistically significant second-order relations of complex cell responses. *Neural Processing Letters*, 8:117–129.

Lades, M., Vorbrüggen, J. C., Buhmann, J., Lange, J., von der Malsburg, C., Würtz, R. P., and Konen, W. (1993). Distortion Invariant Object Recognition in the Dynamic Link Architecture. *IEEE Transactions on Computers*, 42(3):300 – 311.

Leibe, B. and Schiele, B. (2003). Analyzing appearance and contour based methods for object categorization. In *International Conference on Computer Vision and Pattern Recognition (CVPR'03)*, Madison, Wisconsin,.

Leopold, D. and Logothetis, N. (1998). Microsaccades differentially modulate neural activity in the striate and extrastriate visual cortex. *Experimental Brain Research*, 123(3):341–345.

Li, Z. (1998). A neural model of contour integration in the primary visual cortex. *Neural Computation*, 10:903–940.

Lindeberg, T. (1994). *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers.

Loos, H. S. (2002). *User-Assisted Learning of Visual Object Recognition*. PhD thesis, Technical Faculty, University of Bielefeld, Germany.

Löwel, S. and Singer, W. (1992). Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity. *Science*, 255:209–212.

Lyche, T. and Schumaker, L. (1973). Computation of smoothing and interpolating natural splines via local bases. *SIAM J. Numer. Anal.*, 10:1027–1038.

Malach, R., Amir, Y., Harel, M., and Grinvald, A. (1993). Relationship between intrinsic connections and functional architecture revealed by optical imaging and in-vivo targeted biocytin injections in primate striate cortex. *PNAS*, 90(22):10469–10473.

Mallat, S. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 11(7):674–693.

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. San Fransisco: Freeman.

McGuire, B. A., Stevens, J., and Sterling, P. (1986). Microcircuitry of beta ganglion cells in cat retina. *Journal of Neuroscience*, 6(4):907–918.

Mokhtarian, F. (1997). Silhouette-based occluded object recognition through curvature scale space. *Machine Vision and Applications*, 10(3):87–97.

Mokhtarian, F. and Bober, M. (2003). *Curvature Scale Space Representation: Theory, Applications and MPEG-7 Standardization*. Kluwer Academic.

Mokhtarian, F. and Mackworth, A. K. (1992). A theory of multiscale, curvature-based shape representation for planar curves. *IEEE TPAMI*, 14(8):789–805.

Mokhtarian, F.and Abbasi, S. and Kittler, J. (1996a). Efficient and robust retrieval by shape content through curvature scale space. In *Proc. International Workshop on Image DataBases and MultiMedia Search*, pages 35–42, Amsterdam, The Netherlands.

Mokhtarian, F.and Abbasi, S. and Kittler, J. (1996b). Robust and efficient shape indexing through curvature scale space. In *Proc. British Machine Vision Conference*, pages 53–62, Edinburgh, UK.

Morrone, M., Burr, D., and Fiorentini, A. (1990). Development of contrast sensitivity and acuity of the infant colour system. *Proc R Soc Lond B Biol Sci*, 242(1304):134–139.

Norcia, A., Tyler, C., and Hamer, R. (1990). Development of contrast sensitivity in the human infant. *Vision Research*, 30(10):1475–1486.

Olshausen, B. A. and Field, D. J. (1996). Wavelet-like receptive fields emerge from a network that learns sparse codes for natural images. *Nature*, 381:607–609.

Ooyama, K. (1987). Scale-controlled objective analysis. *Montly Weather Review*, 115:2479–2506.

Pavlovic, V. I., Sharma, R., and Huang, T. S. (1997). Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE TPAMI*, 19(7):677–695.

Petrakis, E. G., Diplaros, A., and Milios, E. (2002). Matching and retrieval of distorted and occluded shapes using dynamic programming. *IEEE TPAMI*, 24(11):1501–1516.

Piaget, J. (1936). *La naissance de l'intelligence chez l'enfant*. reprinted by International Universities Press.

Pötzsch, M. (1999). *Object-Contour Statistics Extracted from Natural Image Sequences*. PhD thesis, Ruhr-Universiät-Bochum.

Prodöhl, C., Würtz, R., and von der Malsburg, C. (2003). Learning the gestalt rule of collinearity from object motion. *Neural Computation*, 15:1865–96.

Reinagel, P. and Zador, A. M. (1999). Natural scene statistics at the center of gaze. *Network: Computation in Neural Systems*, 10:341–350.

Ross, W., Grossberg, S., and Mingolla, E. (2000). Visual cortical mechanisms of perceptual grouping: interacting layers, networks, columns, and maps. *Neural Networks*, 13(6):571–588.

Rothe, I., Süsse, H., and Voss, K. (1996). The method of normalization to determine invariants. *IEEE TPAMI*, 18(4):366–376.

Schmidt, K., Goebel, R., Löwel, S., and Singer, W. (1997). The perceptual grouping criterion of colinearity is reflected by anisotropies of connections in the primary visual cortex. *European Journal of Neuroscience*, 5(9):1083–9.

Sekita, I., Kurita, T., and Otsu, N. (1992). Complex autoregressive model for shape recognition. *IEEE TPAMI*, 14(4):489–496.

Simoncelli, E. and Olshausen, B. (2001). Natural image statistics and neural representation. *Annual Review of Neuroscience*, 24:1193–1216.

Skoczenski, A. and Norcia, A. (1998). Neural noise limitations on infant visual sensitivity. *Nature*, 391(6668):697–700.

Snider, C., Dehay, C., Berland, M., Kennedy, H., and Chalupa, L. (1999). Prenatal development of retinogeniculate axons in the macaque monkey during segregation of binocular inputs. *Journal of Neuroscience*, 19(1):220–8.

Somers, D. C., Nelson, S. B., and Sur, M. (1995). An emergent model of orientation selectivity in cat visual cortical simple cells. *The Journal of Neuroscience*, 15(8):5448–5465.

Spelke, E., Breinlinger, K., Jacobson, K., and Phillips, A. (1993). Gestalt relations and object perception: A developmental study. *Perception*, 22:1483–1501.

Sperling, G. (1970). Models of visual perception and contrast detection. *Perception & Psychophysics*, 8:143–157.

Streri, A. and Spelke, E. (1989). Effects of motion and figural goodness on haptic object perception in infancy. *Child Development*, 60(5):1111–1125.

Tarjan, R. E. (1972). Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1:146–160.

Tieng, Q. M. and Boles, W. (1997). Recognition of 2d object contours using the wavelet transform zero-crossing representation. *IEEE TPAMI*, 19(8):911–916.

Tootle, J. (1993). Early postnatal development of visual function in ganglion cells of the cat retina. *Journal of Neurophysiology*, 69(5):1645–1660.

Ueda, N. and Suzuki, S. (1993). Learning visual models from shape contours using multiscale convex/concave structure matching. *IEEE PAMI*, 15(4):337–352.

van Hateren, J. and Ruderman, D. (1998). Independent component analysis of natural image sequences yields spatiotemporal filters similar to simple cells in primary visual cortex. *Proceedings of the Royal Society London B*, 265:2315–2320.

von der Malsburg, C. (1981). The correlation theory of brain function. Internal report, Max-Planck-Institut für Biophysikalische Chemie, Postfach 2841, D3400 Göttingen, Germany.

Wang, J.-Y. and Cohen, F. (1994). Part ii: 3-d object recognition and shape estimation from image contours using b-splines, shape invariant matching, and neural network. *IEEE TPAMI*, 16(1):13–23.

Weiss, I. (1993). Geometric invariants and object recognition. *Int J. Computer Vision*, 10(3):207–231.

Wertheimer, M. (1923). Untersuchungen zur Lehre von der Gestalt II. *Psychologische Forschung*, 4:301–350.

Wiskott, L. (1996). Segmentation from motion: Combining gabor- and mallat-wavelets to overcome aperture and correspondence problem. Internal Report IRINI 96-10, Institut für Neuroinformatik, Ruhr-Universität D-44780 Bochum, Germany. 12 pages.

Wiskott, L., Fellous, J.-M., Krüger, N., and von der Malsburg, C. (1997). Face recognition by elastic bunch graph matching (ps, pdf). *IEEE Transactions on Pattern Analysis and Machine, Intelligence*, 19(7):775–779.

Wiskott, L. and Malsburg, C. (1993). A neural system for the recognition of partially occluded objects in cluttered scenes (ps, pdf). *Int. J. of Pattern Recognition and Artificial Intelligence*, 7(4):935–948.

Yang, Z. and Cohen, F. S. (1999a). Cross-weighted moments and affine invariants for image registration and matching. *IEEE TPAMI*, 21(8):804–814.

Yang, Z. and Cohen, F. S. (1999b). Image registration and object recognition using affine invariants and convex hulls. *IEEE-Trans. on Image Proc.*, 8(7):934 – 946.

Yen, S. and Finkel, L. (1998). Extraction of perceptually salient contours by striate cortical networks. *Vision Research*, 38(5):719–741.

Yonas, A. et al. (1987). Four-month-old infants' sensitivity to binocular and kinetic infomation for three-dimensional-object shape. *Child Development*, 84(4):910–917.

Zetzsche, C. and Krieger, G. (2001). Nonlinear mechanisms and higher-order statistics in biological vision and electronic image processing: review and perspectives. *Journal of Electronic Imaging*, 10(1):56–99.

Zhu, S.-C. (1999). Embedding gestalt laws in markov random fields. *IEEE TPAMI*, 21(11):1170–1187.