Otto-von-Guericke-Universität Magdeburg

Diplomarbeit

Structural Deformable Models for Robust Object Recognition

Verfasser:

Steven Bergner

sbergner@cs.uni-magdeburg.de

10. Dezember 2003

Betreuer:

Stephan Al-Zubi

Prof. Klaus Tönnies

Otto-von-Guericke Universität, Magdeburg (Germany)
{stephan,klaus}@isg.cs.uni-magdeburg.de

Abstract

A hierarchical framework for the recognition of compound deformable shapes is developed. In extension to traditional approaches an additional layer of control is introduced to guide the local search for shapes. This is realized by incorporating knowledge about their spatial relationships. A new technique of expectation maps is applied to allow parallel shape searches to inspire each other. Furthermore, these maps are used to assess spatial coherence among shapes. Thus, the occurrence of well formed shapes at some places in the image may suggest searches for related shapes at according positions. The shape searches are driven by a dynamic physical model. Both, structural model and physical shape model may be refined by adding more training data.

The framework is applied to the detection of images from ant databases. For that purpose a sensor to detect ants by their typical color is developed. The capability of the system to extract semantic information through the matching of structural knowledge may become an important ingredient of a more powerful system for content-based image retrieval. First results of 84% recognition rate on a subset of 75 images from the ant database indicate its usability to recognize compound shapes.

Zusammenfassung

In dieser Arbeit wird ein hierarchisches System zur Erkennung zusammengesetzter verformbarer Objekte entwickelt. Das Problem traditioneller Verfahren, aufgrund der Ablenkung durch lokale Minima das richtige Objekt nicht zu finden, wird durch die Einbeziehung einer übergeordneten Kontrollschicht vermieden. Diese verwendet Wissen über strukturelle Lagebeziehungen zwischen Teilformen, um mehrere parallele Suchen zu koordinieren. Dazu wird eine neue Technik sogenannter Erwartungskarten ('expectation maps') verwendet, die aus statistischen Informationen über die Transformationen zwischen den Teilformen gewonnen werden. Der Austausch von Erwartungskarten hilft verschiedenen Formsuchen, sich gegenseitig anzuregen. Weiterhin werden diese Karten verwendet, um Lagebeziehungen neuer gefundener Teile zu weiteren Nachbarteilen zu bewerten. Anhand einer Bewertungsfunktion wird für jede mögliche Interpretation der Gesamtstruktur die optimale Kombination der Teilformkanditaten ermittelt. Die bestbewertete Interpretation repräsentiert das gefundene zusammengesetze Objekt. Die lokale Suche nach Einzelformen geschieht durch ein dynamisches physikalisches Modell, bestehend aus einem Feder-Masse System. Nach Bestätigung erster Ergebnisse ist es möglich, die Parameter dieses Modells durch statistische Analyse zu verfeinern.

Das Programm findet Anwendung bei der Erkennung von Ameisen in Bilddatenbanken mit Makroaufnahmen dieser Tiere. Dazu wurde ein Sensor entwickelt, der eine Klassifikation der Ameisen anhand ihrer Körperfarbe vornimmt. Weitere Sensoren zur Erkennung von Kanten und Ecken, sowie zur Generierung einer Multiskalenrepräsentation sind ebenfalls Teil des Systems. Eine Erkennungsrate von 84% auf einer Teilmenge von 75 ausgewählten Bildern der Datenbank ist eine erste Indikation für die Funktionstüchtigkeit der entwickelten Methode.

Danksagung

An dieser Stelle möchte ich einigen Personen danken, die mich in der zurückliegenden Zeit unterstützt haben. Im Zusammenhang mit dieser Arbeit möchte ich meinen Betreuer Stephan Al-Zubi nennen, der mir in zahlreichen Diskussionen immer eine wichtige Quelle der Inspiration war. Letztlich ist es seine Forschung, die dieser Arbeit den ersten entscheidenden Impuls gab. Bei der Korrektur war mir Martin Spindler eine grosse Hilfe. Das gleiche gilt für Niklas Röber, der darüber hinaus seinen Druckaccount zur Erstellung eines wichtigen Exemplars für mich geplündert hat. Tiefsten Dank spreche meinen Eltern aus, die mit Ihrer Unterstützung so viele Dinge auf meinem Weg erst möglich gemacht haben. Abschliessend gilt mein inniger Dank Jenny, die ich liebe und die mir in den letzten Monaten in allen Situationen eng zur Seite stand.

Contents

Abstract i

Zusammenfassung iii

Danksagung v

Contents ix

List of Figures xiv

List of Tables xv

List of Abbreviations xvii

1 Introduction 1
- 1.1 Ant databases . . . 2
- 1.2 Modeling the problem . . . 3
  - 1.2.1 Bottom-up and top-down processing . . . 3
- 1.3 Structure . . . 5
- 1.4 Notation . . . 5

2 Related Work 7
- 2.1 Feature Extraction . . . 7
  - 2.1.1 Low-level segmentation . . . 8
  - 2.1.2 Skin color detection . . . 8
- 2.2 Shape and Structure . . . 9
  - 2.2.1 Dynamic models . . . 9
  - 2.2.2 Statistical models . . . 12
  - 2.2.3 Dynamic vs. Statistical Model . . . 13
  - 2.2.4 Search Strategies . . . 15
  - 2.2.5 Structural models . . . 16
- 2.3 Content-based Image Retrieval (CBIR) . . . 17

2.4 Summary 18

3 Basics **21**
3.1 Feature extraction 21
3.1.1 Types of sensors 23
3.1.2 Multi-scale pyramid 24
3.2 A physical shape model 25
3.2.1 Differential equation of Newtonian motion 26
3.2.2 Computing forces 26
3.2.3 Solving the ordinary differential equation 27

4 Concept **29**
4.1 Architecture of the system 30
4.2 A sensor to detect ants 31
4.2.1 Sensor Normalization 34
4.3 Finding shapes 35
4.3.1 Balancing sensor forces 35
4.3.2 Comparing solutions – Quality of fit function 38
4.3.3 Improving Geometry 39
4.4 Guiding the Search 40
4.4.1 Reducing the search space – the property vector 41
4.4.2 Stochastic search with memory 42
4.4.3 Efficient clustering algorithm 43
4.4.4 Expectation Maps 46
4.5 Structural Models 47
4.5.1 Representing structure 48
4.5.2 Inspiring each other: Generating expectation maps 50
4.5.3 Matching structure 51
4.6 Image retrieval 53

5 Implementation **55**
5.1 Goals and Design 55
5.2 The parts of the framework 56
5.2.1 Modular Organisation 56
5.3 The sensor framework 56
5.3.1 Geometry and physical model 57
5.3.2 Creating shapes 57
5.3.3 Intelligent search 59
5.3.4 Managing structure 59

6 Results **63**
6.1 Feature Extraction 63

6.1.1 Shape features from intensity distributions 63
6.1.2 Detecting bodies of ants 64
6.2 Matching Shapes 66
6.3 Finding Structure 68

7 Summary **75**
7.1 Discussion 75
7.2 Future Work 77
7.3 Conclusions 79

Bibliography **81**

List of Figures

1.1 **Different levels of proccessing** in pattern recognition. 4

3.1 **Sensor processing pipeline** showing the constituents of the feature extraction framework. The lower left corner illustrates the interface of a sensor. 22

3.2 **Geometry of a deformable model** showing the model of a leave while moving its last node to the corresponding corner feature. The edges are acting as springs preserving 'natural' smoothness constraints on the shape. 25

4.1 **Overview of the framework.** The system consists of a hierarchical organization of different layers of processing. The image data as the bottom most level is analyzed by sensors. The extracted features are passed on to the matching of deformable shapes. Searchers control the evolution of shapes to avoid influence of local minima. Structural knowledge is connecting searchers that are inspiring each others. In addition, the searcher provides input to form structural hypotheses. 31

4.2 **Acquisition of color samples from ants and background.** A similarly sized set of colors is taken from each class (ant body/background). The sample lines are deliberately chosen to equally pass typical and less typical places. 32

4.3 **Multivariate Gaussian distribution.** This is an example distribution of colors. Its mean and covariance matrix have been used to construct a Gaussian distribution that is overlaid as transparent isosurface at a constant distance of 1σ from the mean. 33

4.4 **Problem of collapsing shape.** This problem is inherent to explicit shape modeling. (left) original shape, (left middle) degenerate shape in a locally stable state, (right middle) stabilization by introducing supporting edges, (right) stabilization through torque forces. 37

4.5 **A regular grid of bins** is used to determine local winners. The grid size is chosen to be less than the radius of an average shape (using the standard deviation σ of its point distribution). 43

4.6 **Expectation Maps** are passed to the searcher to indicate areas of higher anticipation. The image is showing *Pheidole carribaea sloanei*. Based on the position of the head that has been found successfully (red), a search area (blue) for the back can be estimated. The dotted yellow line shows the resulting match for the back. Inversely, expectation maps are as well used to rate spatial relationship between shapes. 46

4.7 **Ant graph** showing all important parts of the structural representation. 48

5.1 **Design mode.** A special mode of the user interface allows to interactively construct geometry along with its set of sensors. 58

6.1 **Different types of sensors**, (a) original image, the blue quad shows the matched shape, (b) Gaussian smoothing, (c) gradient magnitude (edge detection), (d) corner detection. 64

6.2 **Corner sensor at different scales of resolution**, (a) original image, the blue contour is overlaid for better comparison, (b) scale 2^6, (c) scale 2^7, (d) scale 2^8. 64

6.3 **Distribution of ant body and background color**, (a) scatter plot in RGB space (blue crosses indicate ant body, red is background), (b) Gaussian model of the ant body color (iso-surface at $P(ant|c) = 0.05$), (c) $P(background|c) = 0.05$, (d) both. (true positive ant is 96.62%, false negative background is 6.52%, number of ant samples 7067, number of big samples 3403) 65

6.4 **Results of the ant body classifier.** (a) *Pheidole aenescens*, (b) clamped probability of ant, (c) multi-channel gradient magnitude, (d) edges of (b) 67

6.5 **Examples of ant classifier.** (a,b) *Pheidole albipes*, original and classification (c,d) *Cerapachys vitiensis* original and classification. The small inner holes are closed on a coarser scale. 67

6.6 **Dynamic model in action.** The image sequence (read in a row-wise order) shows the convergence behavior of the dynamic model. After being placed close to the structure, the forces automatically drag it towards the final position where it reaches equilibrium. The depicted process takes about 5 seconds if real time steps are taken. . 68

6.7 **Adjustment of spring constants** using the adjustment factor α of Eq. 4.10 (a) difference between template and reference shapes for different adjustments, (b) average of distribution show in (a) indicates a global optimum at 1. 69

6.8 **Structural guidance**, *Pheidole fervens* (a) the single thorax-template is globally applied to the image. Green shapes are local winners after clustering. The purple shape has highest quality. (b) The search is guided using the structural relationship to the head. . 69

6.9 **Multi-structure search**, *Pheidole fervens* (a) the ant is correctly classified with a probability of 81% (anochetus 77%, cerapachys 55%). (b) Only the correct match is shown. 70

6.10 **Multi-structure search**, *Pheidole carribaea* Three different interpretations are applied to the image (showing the extracted edge features) (a) The output of all searches is blended on top. The green shapes are possible candidates. (b) The interpretation as pheidole is emphasized as structure of highest probability shown in front of the output of the ant color sensor. (pheidole 75.4%, anochetus 72.0%, cerapachys 61.6%) 70
6.11 **Multi-structure search**, *Pheidole subarmata* (a) all shape candidates with most probable interpretation (orange). (b) The head is slightly displaced, the small peak in the lower thorax has been detected although being partly occluded by the leg. (probabilities of different interpretations: pheidole 82%, anochetus 76%, cerapachys 62%) 71
6.12 **Multi-structure search**, *Anochetus cato* (a) all shape candidates with most probable interpretation (orange). (b) The head is slightly displaced, the small peak in the lower thorax has been detected although being partly occluded by the leg. (probabilities of different interpretations: pheidole 74%, anochetus 81%, cerapachys 65%), searching time: 50 seconds 71
6.13 **Multi-structure search**, *Anochetus haytianus* (a) the ant is correctly classified with a probability of 79% (pheidole 50%, cerapachys 0%). (b) The background results in an insufficient edge signal after the ant sensor. The remaining weak signal is supported by the expectation from structural knowledge. 72
6.14 **Multi-structure search**, *Cerapachys vitiensis* An important property of the relative structural rating is that it can compare structures of different lengths. 72
6.15 **Ambiguous interpretations.**, *Pheidole amazonica* (a) mistaken interpretation of a pheidole and as Anochetus, (b) the characteristic round shape of the back is lost at a coarser scale of resolution, which has been used to increase the range of convergence. 73

List of Tables

1.1 Summary of notation used. 6

5.1 Overview of important classes used in the implementation. 61

List of Abbreviations

AAM	Active appearance model
ASM	Active shape model
ASSM	Active shape structural model
CBIR	Content-based image retrieval
FEM	Finite element model
LOD	Level of detail
PCA	Principle component analysis
PDM	Point distribution model
GA	Genetic algorithm

Chapter 1

Introduction

The field of automated object recognition is a fascinating and challenging area of research. The subject – images – may have many different levels of meaning. Thus, it is difficult for an automated recognition system to capture all information expressed by the image. If used wisely, images may transport very complex messages – brought to the observers mind in a blink. Beginning with the impressions and moods that basic structures and colors may pose, we start to develop further interest for their visual message. At closer look, shapes are being identified as known objects. Their spatial arrangement may reveal more complex connections and relationships. Finally, the scene is reflected by our mind and our thoughts start to play around with it.

Computational Visualistics is an interdisciplinary science centered around producing and analyzing images with the aid of computers. It is possible to approach the topic from two different sides. The generation of images is subject of computer graphics yielding ever more realistic renditions as algorithms and hardware evolve. By artificially producing images and studying their effect on human observers, we learn a lot about the effect of different techniques to our mind.

The other direction is to develop own methods of image processing and interpretation that may be performed by a computer. The architecture of modern image recognition systems can learn a lot from investigations into human physiology. In turn, the success of computer systems that have been developed along certain biological archetypes may help to support new theories.

Aside from the purpose of investigating relationships between images and observers, the goal of teaching the computer to 'see' is fascinating and motivating on its own. A desirable result could be the development of more natural ways of human-computer interaction. Furthermore, the computer itself becomes a more useful tool if it can process visual information without needing our help.

The goal of this thesis is based on the idea of improving the usefulness of the computer by making it recognize variable shapes and their spatial relationships

that are inherent to complex objects. Several layers of processing are involved in this task. At the topmost layer prior knowledge about local features, shape, and structure is used to construct a system that can cope with distorted or cluttered data. The incorporation of knowledge makes the recognition process more robust. On the other hand it might limit the applicability to varying situations and problems. Thus, special care has to be taken to keep the system flexible and adaptable to different tasks.

1.1 Ant databases

Apart from theoretical motivation to develop a more versatile method of shape detection, the focus is on an application accompanying the development. The discipline of *systematics* studies the diversity of biological beings and their classification. An important tool for such studies are catalogs of specimens. Recent development in multimedia technology and telecommunication – in particular the development of image databases within large networks – are becoming an increasingly powerful tool for researchers. Species are added to these databases on a daily basis. To cope with the large amounts of information, it is desirable to have powerful automatic support by computing technology.

In the following, a short introduction into the basic vocabulary of this area is provided. The naming scheme that is used is called *taxonomy*. It is a hierarchical naming system that goes back to Carl Linnaeus' "Systema Naturae" published in 1758. According to this system, all 'living things' on earth are classified into five *kingdoms*: monera, protista, fungi, plant, and animal. The next lower levels of classification are denoted as *phylum*, *class*, *order*, *family*, *genus*, and finally *species*. Genus and species together form a unique *scientific name*. This is called the system of *binomial nomenclature*. The overall number of species is estimated from about 2 millions to over 100 millions. Due to this large number, scientists have not yet been able to catalogue all of them[1].

The goal of *Cladistic analysis* is to determine relationships of taxa by finding evolutionary branches. The key to determine such relationships are common characteristics, such as morphological similarities.

This is where the connection this thesis becomes apparent. The extraction of characteristics of species is a crucial step when deriving insights from the catalogs. The combination of structural knowledge and deformable shapes is a very promising approach to model the morphological properties of complex life-forms. Along these lines, the model developed in this thesis might be a first step towards a more sophisticated extraction of features. The automatic detection of semantically distinct characteristics could be of great benefit for a productive utilization of the image catalogs of species.

[1] More details may be found in introductory literature on biology, such as [12], pages 19-20.

The focus in this thesis is on ant recognition. Placed in the taxonomic hierarchy, ants are found in the kingdom Animalia, phylum Arthropoda, class Uniramia (Insect), order Hymenoptera, family Formicidae. The underlying example image used to construct Fig. 1.1 is a profile view of Genus *Pheidole*, Species *yaqui*. The particular database we are operating on, has been obtained from two sites on the internet. The first is the *MCZ Entomology primary type specimen database* of the Museum of Comparative Zoology at Harvard University[2]. The second one is AntWeb by the Californian Academy of Sciences[3].

1.2 Modeling the problem

The important difference of image processing in contrast to plain data processing is that the data operated on is based on visual information. Thus, when developing algorithms for automatic image processing, it is possible to incorporate assumptions about:

1. the physical conditions of the scene,
2. the properties of the image representation, and
3. the mechanisms of the human visual system that is perceiving the image.

The image data consists of a spatial arrangement of image points – so-called pixels. We can identify objects by looking at pixels of specific colors. Information about the image representation (the grid and the physical meaning of its pixel values) can be used to identify this specific color. Furthermore, it is possible to group pixels together and compute local features, such as the gradient, or global features, like histograms. It is already possible to draw basic conclusions by extracting these features. If it comes to higher level recognition tasks, automatic algorithms are still surpassed by the capabilities of the human perception. Especially when it comes to the point of grouping features together to form a contour, it can be inspiring to look at the way humans cope with visual information. To interpret an image it is necessary to make assumptions about the data and the objects being analyzed. These assumptions *form a model*. Using such a model, it is possible to develop an algorithm that searches for patterns in the data.

1.2.1 Bottom-up and top-down processing

Similar to the processes in the human perception, information from images is extracted at different levels[4]. At the lowest level one just looks at single pixels or, on

[2]to be found at `http://mcz-28168.oeb.harvard.edu/mcztypedb.htm`

[3]located at `www.antweb.org`

[4]Details about the pattern recognition processes in the human visual system are, e.g., given in a textbook on sensation and perception by Matlin [27] on pages 161–176.

a coarser level, local neighborhoods of pixels are inspected. Using these techniques, it is possible to extract local features of the data, such as edges or homogeneous regions. On the next level of representation the observed features are grouped together to shapes. Recognizing these shapes as known objects of given properties, elevates the representation of the image contents to a semantic level. This allows one to conceptually group things together and to draw conclusions from the depicted scene. The process described here is a *data-driven* or *bottom-up* view of recognition. The opposite methods to these mechanisms are *top-down* processes. This means to have certain expectations or *hypotheses* about the content and the properties of an image. These expectations are used to focus on specific information contained in the image. Output from the corresponding stages in our framework is illustrated in Fig. 1.1. The image is constructed from a typical example from the ant database. In the lower left corner the result of an *edge feature* detection is shown. These features are grouped using a *shape model* of the head. The *structural relationships* between body parts provide *hypotheses* to narrow the search range for possible backs.

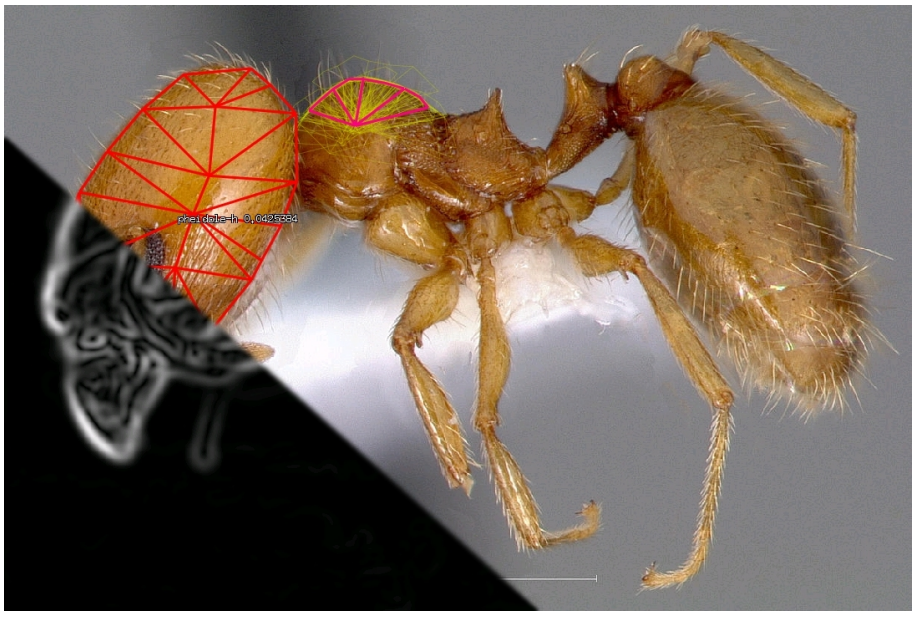


Figure 1.1: **Different levels of proccessing** in pattern recognition.

The actual viewing process in the human perception system is utilizing both mechanisms, top-down and bottom-up, assisting each other. The system presented here will make use of similar mechanisms. Different layers of the framework will cooperate to combine both, impulses that arise directly from the data and directions for further searches that are derived from higher level hypotheses.

1.3 Structure

The thesis is structured as follows:

Chapter 1 provides a short *introduction* to this thesis, including motivation, goals, and the mathematical notations used.

Chapter 2 is reviewing the state of the art in object recognition deriving consequences for the presented concepts.

Chapter 3 is describing *basic theories* and details of existing approaches used in the thesis.

Chapter 4 is the *conceptual part*. Here essential issues are discussed along with ideas and solutions that have been expanded into the system.

Chapter 5 provides details about the *implementation*.

Chapter 6 shows the system at work, providing a presentation of the *results and their analysis*.

Chapter 7 draws conclusions and summarizes the whole work by discussing the actual achievements giving an outlook to future extensions.

1.4 Notation

Table 1.1 provides an overview of frequently used symbols. Please note that **bold font** indicates a vector valued quantity (e.g., $\mathbf{s}$) and *italic font* symbolizes scalar values (e.g., s).

Notation	Meaning
$\mathbf{x}_i$	position of node i
$\mathbf{d}_{ij}$	distance between nodes i and j
$\mathbf{a}_{ij}$	adjacency between two nodes (1 if edge is defined 0 if disconnected)
s_{ij}	rest length of the spring between nodes i and j
$k_{s_{ij}}$	spring constant of the spring between nodes i and j
$\mathsf{E}(x_i)$	expected value calculated over set of all x_i
$\mathsf{S}(x_i)$	standard deviation calculated over set of all x_i

Table 1.1: Summary of notation used.

Chapter 2

Related Work

As mentioned in the introduction, image processing is mainly divided into bottom-up and top-down approaches. The algorithm that is presented in this thesis is operating on different levels of representation. At the lowest level are local image features. These features are grouped to form shapes and structures. On each of these levels several approaches exist to solve specific problems that are relevant to this work. In the following these related works are being analyzed in more detail. This chapter is structured along the different levels of representation beginning at the lowest – the extraction of features.

2.1 Feature Extraction

Knowing how the image is constructed and what kind of information it contains is the first building block for a model. It is possible to find objects in images by only looking at the values of pixels. The most basic method is classifying all pixels that are above a given *threshold* as an object. This allows to *segment* an image into object and background information.

As a pre-computation *transformations* can be applied to an image to obtain a more meaningful representation of its contents. Two different methods can be distinguished: 1. mappings of *isolated pixel values*, and 2. operations on pixels based on their *local neighborhood*. Methods of the first type are for instance histogram-based analyses, such as contrast enhancement. Our technique of ant skin detection, discussed in §4.2, falls into this class. The other class of techniques covers a large range of methods, such as the use of local histograms for illumination correction, ranking filter, or linear operations. A more detailed discussion can be found in Jähne [20]. The general goal of image transformation is the *extraction of features* that are more significant for the recognition problem.

It is important to note that the occurrence of features changes among different *scales of resolution* [25]. Besides choosing an appropriate type of feature extraction,

a decision on scale to look for it, has to be made as well.

Approaches exist to automatically choose an appropriate scale. For a given feature, Lindeberg [26] proposes to maximize its scale-normalized Gaussian derivative among different resolutions. For our approach, a manual selection of scale is sufficient, because the smoothing is only applied to improve convergence of shape matching. Further details about our method of feature extraction are given in §3.1.

2.1.1 Low-level segmentation

In previous work we have dealt with the problem of insufficient data (noise or low contrast) without making use of high level knowledge [4]. A watershed transform has been applied to multi-modal data to quantify micro-organisms. To overcome the inherent problem of over-segmentation, a merge criterion was introduced. For that, the relation between the depth of adjacent basins and the height of their connecting rims are used to hierarchically decide the merging of segments.

By only utilizing very little assumptions – without the need for smoothing the data – it is possible to suppress the effects of noise. Furthermore, the incorporation of very few model information is making the method more applicable to different kinds of shapes as well. Unfortunately, there is still a considerable number of situations where shape knowledge would have helped to divide clustered objects more precisely. This became an initial motivation to investigate more elaborate shape based segmentation techniques.

2.1.2 Skin color detection

The detection of human skin in images is a densely researched area[1]. This is a related field to our problem of detecting bodies of ants in images. Thus, it seemed promising to consider the application of existing approaches to our problem. A major motivation for color based segmentation is its simplicity. Typically, these methods are fast, because they do not need any neighborhood information. The large quantity of images available on the web plays an important role when statistics are needed to obtain a model of skin.

An elaborate model is presented by Jones and Rehg [21]. They are obtaining histogram-based Bayesian color models from skin and non-skin labeled images. The method achieves satisfactory results by relying on color information only.

The counterpart to histogram-based models are parametric Gaussian models. Hsu et al. [18] are using statistical Gaussian clustering in YCbCr color space with white tone correction. These models produce fairly reliable results and require far less amount of training data than the Bayesian method. Hence, we have chosen to develop an approach to detect ant skin along the lines of a Gaussian model. More details about that are given in § 4.2 and § 6.1.2.

[1] A survey for pixel-based detection of skin tones is given by Vezhnevets et al. [43].

A connection of coarse shape and color features is used by Fleck et al. for finding naked people on the web [14]. Their color detector is a manually setup model that is assisted by secondary features of texture and geometric properties, such as the attachment and posture of limbs to the human body.

An advantage of methods that are only relying on local features or color values is their flexibility and adaptability to various applications. This is an important advantage that should be well considered before looking at more complicated methods. Of course, the lack of context information is not beneficial in general. Thus, if local methods tend to become unreliable or are not expressive enough, we need to step up to a higher level of representation.

2.2 Shape and Structure

In most cases, the images presented to automatic detection algorithms are suffering from ambiguities or distortions. Especially in the field of medical imaging, data sets are problematic due to poor resolution containing aliasing artifacts. Or, if working on real photographs – as this is the case for the insect databases – objects are mostly inferring with each other. This poses problems of occlusion of unexpected objects. Only using local image properties, algorithms often fail to connect broken edges or occluded object parts. Thus, they are unable to cope with ambiguous or incomplete data. At this point high-level techniques provide additional knowledge and assumptions that may help to overcome limitations of local image analysis. Such techniques can be divided into three different classes[2]:

1. **dynamic models** using physically-based dynamics to constrain shape deformation,
2. **statistical models** that obtain typical deformations by evaluating training data, and
3. **structural models** that generate a desired object by assembling a set of sub-shapes and matching them to the data.

In the following sections, these three distinct approaches are discussed in more detail. Close attention will be payed to how they relate to each other and what are their specific qualities and drawbacks.

2.2.1 Dynamic models

Dynamic models are inspired by the real-world behavior of flexible objects such as rubber masks. They have been introduced to image processing by Terzopoulos and Fleischer in 1988 [40]. In addition to the shape geometry, smooth deformations

[2]The classification corresponds to the one chosen by Al-Zubi[1].

are enabled and constrained within a dynamic physical system, such as a spring-mass model. The imbalance between *internal* and *external* energy is setting the model in motion. The internal energy represents the degree of deformation, while the external energy reflects the correspondence to the chosen image features. The forces resulting from these energies are acting on the model and attract it. As the forces balance, the motion comes to rest and the system reaches equilibrium.

One of the first examples of such models are active contours or so called *snakes* introduced by Kass et al. in 1988 [24]. The *snake* is winding its way through the image between two user defined end points being attracted by strong edges[3] and simultaneously adhering to internal constraints of rigidity and stiffness. Through its internal constraints the snake is capable to keep its direction for a certain distance, bridging over disconnected edges. An important part of the system are the user applied constraints. In addition to the end points, places of attraction or repulsion can be defined to incorporate user knowledge. Since the model is, only incorporating very basic assumptions, this interaction is necessary to enable a general usability of the system.

An extension to this method are topologically adaptable snakes by McInerney et al. [29]. Here, the image data is re-sampled to a triangular grid allowing more sensible decisions as the topology of evolving snakes is changing. Snakes are powerful tools to trace contours. Nevertheless, because of the very basic constraints of smooth contours, they still need a lot of input through user interaction to produce usable results.

Yuille et al. use an analytical description of deformable templates [45]. Arc lengths and complete circles are combined to model eyes and the mouth for a face recognition system. Features, such as intensity peaks or valleys and edges are precomputed and smoothed out to a coarser scale. The features are integrated over specific paths or areas within the template. A quality function is computed by a (negatively) weighted sum of the features and shape deformation estimators. The interesting idea about the approach is that the search is guided by a schedule. Different epochs of exploration are defined by putting different emphasize on the weights in the quality function. A problem about the approach is that the parameters of the model are set empirically and are not further justified by the algorithm. As well, a spatial relationship of different parts of the face is not exploited. Thus, as shown in their results, it is possible that a mouth is successfully fitted to the place of an eye. The advantage of their work over the *snakes* approach is the incorporation of domain specific knowledge into the developing process of the model, enabling the trained model to work autonomously within a specialized task.

Another common choice of geometry are finite element models (FEM) as discussed by Pentland and Sclaroff [33]. They apply modal analysis to extract *free*

[3] A common feature for edge extraction is the computation of intensity gradient magnitudes of a smoothed image.

vibration modes of a given deformable template. The deformation can then be described by only employing the most significant (low-frequency) modes. Besides being more compact, this description facilitates comparison of deformations at multiple levels of detail. The problem inherent to this technique is one that applies to most physics-based approaches. The correct treatment of deformation is based on an appropriate adjustment of the physical parameters of the model. Unfortunately, a solution on how to obtain these parameters from images is not provided.

Metaxas et al. [30] describe a method for estimating a set of physical parameters from a set of images. They are assuming a model of known deformation parameters. Based on that, they develop a method for estimating external forces that are governing the movement in the image. This technique is successfully applied to improve the tracking of physically based movements in an image sequence.

The aspect of tracking objects using dynamic models is quite promising because their physical properties are a good estimate to the real-world behavior of depicted objects. Furthermore, the matching is significantly simplified because an initial position for the following frames can be estimated from previous matches.

A more recent approach are *active blobs* by Sclaroff and Isidoro [36]. Their technique supports real-time deformation detection and tracking in image sequences. An object is represented by a triangulated mesh working as a spring-mass system. The image appearance (texture) inside of the mesh is captured for the matching. Using the approach of difference decomposition [15], residuals (difference images) are precomputed for slight changes of shape parameters of the mesh on the original image. Thus, statistics are obtained that help to suggest a certain shift of the mesh points given the difference image of the current matching process. Robustness to specular highlights is achieved by using an error mapping that tolerates outliers. Additionally, the system is accelerated to real-time tracking, utilizing graphics hardware capabilities.

Three-dimensional models

The extension of deformable models to 3D is not straightforward for two reasons. Firstly, in 2D it is easier to make the geometry robust against collapsing by introducing additional supporting edges or torque forces to maintain angular dependence between adjacent edges. Additionally, the search space increases, which makes the matching more complicated and probably more depending on the initialization of the search.

A 3D finite element model has been introduced by McInerney et al. [28]. A deformable sphere (*balloon*) is stabilized and extended by an inflation force. By restricting further movement to just the inflation from a given point, the above problems are avoided. The inflation stops if the forces from attracting surfaces are surpassing the previously set inflation force. Thus, a physically based region growing is realized that can pass beyond weak surfaces and as well bridge between

disrupted contours. The result is a useful tool for 3D segmentation, still requiring user interaction to correctly find the objects in a volume.

A more profound solution to the problem of instability is the use of implicit geometry. The surface of an object is not formed by explicit points. Instead, it is obtained by tracing a contour at a given distance along a medial axis representation (so-called skeleton).

Such models naturally facilitate a multi-resolution or level-of-detail (LOD) description [23]. These models have successfully been applied to medical image segmentation [34] and statistical evaluation [22] to justify deformation. LOD representation is also of great use for real-time graphics [41].

In general, the advantage of deformable models is that it is relatively easy to define a set of parameters allowing for flexible adaption to the data. On the other side this setup is quite arbitrary. A proper setup of physical parameters to describe *typical deformations* for an object is fairly complicated. A very costly way to achieve that would be the use of measured real-world material constants. Unfortunately, for most image contents there is no such material information available. Making use of several training images to construct a more descriptive model is leading to a different class of deformable models that will be discussed in the next section.

2.2.2 Statistical models

Modeling deformation requires two kinds of information: the shape of an object and the information how the shape can be modified. The solution presented in the last section is looking at the physical properties of the object itself. As mentioned earlier, it is difficult to set up these parameters correctly. Usually, no further information describing the image contents exists. Furthermore, the shapes and their variation in the image are not always subject to physical laws. The approach of statistical models is taking this fact into account. The only information used to create a model is a set of training images showing a clear shape of the desired object labeled by a set of landmarks. The statistics obtained from the training data can be used to fit the physical model that is applied later during the matching. If we want to tune the physics to behave like the statistics obtained – why not use the statistics directly for the matching? This is the underlying idea behind *statistical models*.

The most influential representative among this class of methods is the *active shape model* (ASM) developed by Cootes et al. [8]. The following explanations are more detailed because this algorithm exemplifies the method behind all statistical shape matching approaches. The training data is a number of images that have been labeled using a fixed set of topologically related landmarks that are forming the contour. Such models of loose points are commonly referred to as point distribution models (PDM). Prior to the analysis the shapes are aligned to compensate for varying transformation (position, orientation, scaling). Now, the coordinates of the

points of each contour are concatenated forming one large shape vector $\mathbf{x}$. From the set of shape vectors, the mean shape vector $\bar{\mathbf{x}}$ is extracted and the variation around this mean shape is determined using principal component analysis (PCA). Only using the first few modes of variation (combined in a matrix $\mathbf{P}$), it is possible to describe most of the occurring shape variations. A shape can then be described in terms of a linear model

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}\mathbf{b}. \tag{2.1}$$

The weights for variational modes are the *shape parameters* $\mathbf{b}$. This yields a low-dimensional representation of the characteristics of the shapes in the training set. To incorporate information about the image content, grey-level profiles orthogonal to the contour through each shape are taken. Another PCA on these profiles results in the *grey-level parameters* $\mathbf{g}$. The search for a match is performed in the space of shape parameters. At each iteration the contour points are shifted to optimally correlate the distribution of trained grey-level profiles[4]. The resulting contour is projected to the shape parameter space and then back to the coordinate space for the next iteration until convergence. The search is further sped up by operating on multiple scales of resolution.

A further extension to this method are *active appearance models* (AAM) presented by Taylor, Edwards, and Cootes in 1998 [7][5]. Here, the texture within the contour is additionally taken into account. Statistics about the shape variation correlated to the textural variation are used to produce a low dimensional representation of appearance for efficient matching. This approach relates to the technique of *active blobs* that have been discussed earlier. Both approaches use difference decomposition to speed up the matching of the entire texture area. The major difference is that the dynamic model of active blobs is more intended to match one specific object, where the AAM is capable to employ characteristics of a whole class of objects (e.g. faces) for matching.

2.2.3 Dynamic vs. Statistical Model

Statistical methods overcome the heuristics commonly used in dynamic models by thoroughly analyzing given training data. This results in a more compact description of the model and a better justification of its parameters. The extensive use of training data is an advantage on the one hand. On the other, the strong dependence from the training set is a disadvantage as well. To produce satisfactory results, the variation of the training data has to precisely reflect the characteristics of new images. If the training set is too small or not representative the resulting model will fail. Especially, if only one example is given, no variation can be determined and thus no matching can take place. Here lies the important advantage of dynamic

[4] This correlation is determined using the Mahalanobis distance.

[5] A nice comparison of AAM and ASM can be found in [6].

models: They are ready to use because of their reasonable default characteristics.

Work on combining the two approaches was done by Cootes and Taylor in 1995 [9]. They treat one geometry as FEM and PDM extracting free vibration modes and variation modes. The combined linear model tends to use the vibration modes as only one or very few training shapes are given. The representation converges towards the statistical modes as more training shapes are provided.

A more recent work by the same authors is as well combining elastic and statistical models [10]. Here a global AAM is combined with local smoothness constraints and local AAMs around each node. This means the grey-level profiles used in ASM are replaced by a Gaussian windowed neighborhood around each node that is subject to an AAM analysis. The local AAMs suggest displacements for the nodes of the current model. These displacements are smoothed out over the geometry. This approach is capable to perform shape matching from just one example, becoming more reliable as more training data is added.

The two methods above address the problem of rare training data quite well. Another problem for statistical analysis is the quality of the landmarking within the training data. The question which landmarks to choose and how to place them reliably is still an area of active research. Currently hand-crafted models are the "gold standard" since a human operator is still most capable to decide which points are the most expressive ones. A work by Davies et al. [11] is analyzing different contour parameterizations to automatically find the most descriptive landmarks along a given contour. Their solution is an important step towards a less heuristic landmarking system.

Another work of interest that is dealing with statistical evaluation of shape descriptions is by Joshi et al. [22]. They are considering M-reps that have been shortly described above. A statistical evaluation of medial representations using spherical harmonics is done by Styner et al. [39]. This technique is related to Fourier analysis but using wrapped (spherical) basis functions to describe the progression of closed surfaces.

A quintessence of the review is that dynamic models are useful for their default behavior. Statistical models are more deliberately choosing model characteristics but suffer from a need for extensive and reliable training data. Approaches to combine both strategies exist, though they are more founded on the statistical side.

A different approach worthwhile to consider would be use a dynamic model because of its default behavior, guiding it as new datasets are being explored. As new successful matches are confirmed, these can be used as training data for the dynamic model allowing less interaction for subsequent trials. A statistically refined dynamic model is, unlike statistically based approaches, capable to represent non-linear behavior. Such a hybrid system overcomes the problem of acquiring reliable training data by incorporating this task online into its working process.

2.2.4 Search Strategies

An important issue with the approaches described above is that their outcome depends on the initialization of the model prior to the search. This means if the model is placed too far off the correct solution it will get stuck in a local minimum. A common view to the problem of shape matching is that of a high-dimensional search. The parameters characterizing the deformable model and its placement in the data are making up the search space.

A famous approach to perform global optimization in high-dimensional search spaces are evolutionary strategies (GS), as introduced by Ingo Rechenberg 1960's, or genetic algorithms (GA), as introduced by John Holland in the 1960's. Both techniques are following the basic principle of adapting mechanisms of biological evolution to computer simulation. Evolution can be understood as an optimization process of a population adapting itself to the conditions of its environment. A member of the population is represented by its *genes* that can be a tuple of discrete or real numbers. It is possible to evaluate the goodness of a given combination of genes using a *fitness function*. Based on the fitness, winners are *selected* to spawn the next generation by recombining their genes. In addition to the crossover, genes are mutated randomly by a certain rate.

Hill et al. [17] have successfully employed GA to match a deformable model to medical data. Their geometric model is a spline characterized by ten parameters, which are bitwise mapped to the genes of the population. The objective function seeks to minimize the distance between boundary points and edges in the data. Additionally, it tries to place these points at strong edges with an additional preference for minimal variation of edge strength among the boundary. Their results show convergence independent of the random initialization.

Another method to search the space of possible shape deformations is by placing the nodes in a systematic manner. A recent approach by Felsenszwalb [13] employs *dynamic programming* to efficiently perform a full search obtaining a globally optimal solution for a given energy function. Their idea is to split up the polygon that is representing the object. The use of a constrained Delaunay triangulation allows to eliminate single triangles along a medial graph through the shape. This elimination scheme is exploited to reuse sub searches in a dynamic programming approach. Experimental results show robustness against distortion and occlusion. Inherent to the method is that no initialization is needed.

Based on the deformable models by Terzopoulos, Hamarneh et al. introduce a *brain* to control the behavior of their deformable *organism* [16]. They establish a hierarchy of processing levels beginning at the sensation of data that is part of a physical model influencing the deformable shape. The brain is perceiving information extracted from the sensors. As a reaction to the observations it is actuating deformations by a system of muscles. The high-level control layer involves a plan or schedule as well as prior knowledge about the search process to make

sensible decisions.

The design of their system reflects the fact that deformable models need guidance. This thesis is developed along this assumption drawing major inspiration from the idea of incorporating an *intelligent* control layer. The important difference to the approach by Hamarneh et al. is that our goal is to keep the brain control less abstract. Being interested in the question of how to actually manage a sensible guidance of shape searches we have considered another class of approaches – the structural models.

2.2.5 Structural models

The two major methods for deformable shape matching described so far are both looking at a shape in its entirety. But often it is possible to decompose a shape into distinct functional parts. Imagining the appearance of a human, many different postures are possible. If analyzed statistically this would result in a large number of variation modes possibly suffering from a lot of non-linear displacements, such as rotations of body parts around joints. Such problems can be solved using *structural models*. A complex shape or a scene is interpreted as a composition of structural constituents according to given construction rules. Following the example of human shape, the visual impression could be split up into arms, legs, head, etc. Each of these shapes could be subject to a more specific shape analysis because it is treated separate from the spatial relationships to its neighboring shapes.

Structural models have first been applied to problems in graphics. A famous approach modeling plants are *L-systems* introduced by Lindenmayer in 1968 (see [35]). Another early work by Stiny and Gips (1971) is introducing a *shape grammar* to generate aesthetic paintings [38].

A more recent approach in image analysis is the *shock grammar* introduced by Siddiqi et al. [37] in 1996. They are applying grammatical rules to the analysis of shapes presented in binary images. A shape is decomposed into a medial graph consisting of atomic units – so-called shocks. The set of possible combination of shocks is expressed as a grammar allowing to prune invalid structures in favor of more likely ones. The resulting graph is representing the object and can be used as a similarity measure. A drawback of this model is that the statistical variability of the shape atoms is not taken into account.

Zhu and Yuille [46] have developed *FORMS*, a framework structurally linking deformable models to form more complex objects. Deformed worms and circles are arranged to form the contour of an object. If such an arrangement has been extracted from the image it is matched against a database of shapes. Possible matches are then propagated in a top-down manner to adjust the deformable subshapes. The model employs no hierarchy among parts and only two deformable templates (worm and circle) are used. Furthermore, the parts are not influencing each others appearance. The structural representation is limited to relations along

the medial axis. Thus, the characteristics of the shapes are described, but their spatial relationships are restricted to connections points along the medial axis. The major drawback is that the shape matching is only working on readily extracted silhouettes.

A recent development by Al-Zubi and Toennies [2] is addressing this problem of structural covariation, which is not covered in the framework above. Their approach of *active shape structural models* (ASSM) combines the advantages of structural and statistical shape representations. A single shape is statistically evaluated in terms of other shapes it relates to in the structure. This *shape context* is used to narrow statistical variation of shape constituents based on the role they play in the composition. The achievement of this approach is that the incorporation of structural knowledge into statistical evaluation allows to extract more descriptive local characteristics of single shapes. Additionally, specific deformations are taken into account that occur if the shape appears in a different context. Thus, the model is capable of comparing and characterizing shapes based on structural and shape variability, which makes it a very promising approach for many domains.

2.3 Content-based Image Retrieval (CBIR)

An interesting application for a general system detecting complex deformable shapes in images is content based image retrieval (CBIR) [6]. An important part of a CBIR system is the feature extraction. Any query that is made later on is operating on the features, instead of looking at the actual images in the database. The significance of the features is a crucial point for the quality of the query results.

An example for a general system is BlobWorld by Carson et al. [5]. They extract spatially connected regions of similar appearance (based on coarse color and detail texture features). This results in a partition of an image into so-called blobs. The features of each blob (location, texture, and color) are extracted separately. The query is done searching for one or more specified blobs without using their interrelationships.

Depending on the purpose of the system, it is possible to tailor more or less specific feature extraction methods. An important driving force behind CBIR research is the huge number of images available on the Internet. The huge variety of its image content is making it difficult to apply shape or even structural models for the retrieval. This and the fact that an automatic and reliable extraction of shape information from images is a complex problem, probably makes global image features more attractive and more successful on a general level.

The situation changes as image databases of more constrained content are taken into consideration. A good example of shape based retrieval is presented

[6] A survey of CBIR systems is given by Veltkamp and Tanase [42].

by Mokhtarian et al. [31] for their database of fish silhouettes[7]. Their *curvature scale space* is encoding the one-dimensional run of a closed contour represented by zero crossings of the deflection on multiple scales. The matching is done by first aligning the deflection maxima and then computing a difference norm at the finest scale. The system is shown to be robust against scaling and rotation and provides a promising similarity measure. Two problems are the restriction to the specific domain and the fact that the system can only handle images containing objects having one single closed contour that is easy to extract.

In general the direct comparison of contour points is too slow when matching a query shape to a large database of reference shapes. This problem is addressed by Belongie et al. suggesting a method called *shape context* [32]. Distance histograms of landmarks are used as coarse descriptor for a quick pruning of candidate shapes.

As shown, the extraction of features is a central ingredient of a CBIR system. Our goal is not to construct a fully working system of that kind. Rather, the results of thesis are intended to be a contribution towards obtaining more significant characterizations of image contents.

2.4 Summary

Current approaches to shape detection can be split into dynamic and statistical models. Dynamic models are often relying on heuristics being constructed for specific applications or having *inadequate justification* (e.g., by statistical evaluation) or need to be guided through high-level interaction. Statistical models need *training data of high reliability and great quantity* to be able to start their productive work. Our approach tries to combine the benefits of both approaches: the ability of dynamic models to deliver immediate results and the justification of the model through online refinement of physical parameters as successfully found objects are confirmed by the user. Thus, one goal of this thesis is to incorporate statistics into the physical model.

As stated earlier in the review of deformable shape matching methods, a very restrictive problem is their tendency to get trapped in *local minima*. Thus, it would be beneficial to study the use of a higher-level control mechanism. Approaches to this issue exist but only few of them consider structural relationships between deformable shapes. Furthermore, our goal of employing structural knowledge for the control of several dynamic models has hardly been addressed so far. A decomposition of shapes would allow to narrow the variation of their sub shapes and thus increase specifity. Generality is recovered by composing the structure of multiple different shapes.

According to the survey discussed in § 2.3, shapes or even structures are not

[7] A demo of the *SQUID* system can be found at
`http://www.ee.surrey.ac.uk/Research/VSSP/imagedb/demo.html`

used often in CBIR. Here lies the relationship to this thesis. A possible application would be to establish a structural representation of the content of the images. A more semantical feature extraction might allow to distinguish images that vary in only very *small but important* details. This becomes an especially interesting issue when retrieving images from insect data bases.

Chapter 3

Basics

The problem set out for this thesis is to recognize a structural composition of deformable shapes in 2D images. The previous chapter has shown that this is a quite complex problem ranging over several different research areas. Some of the approaches that have been discussed are providing useful bits for the system that is to be developed here. This chapter is going to take a closer look at the existing solutions that have proven to be useful in our system. Here, no *new* methods will be introduced. Instead, this chapter pays closer attention to some specific aspects or solutions to detail questions arising when actually realizing the existing approaches. By the end of these explanations the foundations behind this work should have been illuminated in enough detail. Based on that it will be possible to discuss the improvements and new solutions proposed in chapter 4.

3.1 Feature extraction

Whatever architecture our recognition system is constructed of, the first step is to take a look at the data. Among several methods to analyze the data the *extraction of features* is a very natural choice. This means local regions of pixels are grouped together and are inspected for possible occurrence of edges, corners, or simply homogeneous areas. Similar grouping operations are carried out by the receptive fields of the ganglion cells in the human eye[1]. These cells respond to local frequencies in the visual impression the photo receptors are exposed to. In the visual cortex even more complex groupings of signals from ganglion cells are performed to detect lines, corners, specific orientations or certain kinds of movement.

The *sensor framework* we are developing here consists of feature extractors that are mostly common in use[2]. The sensors are chosen to capture all low-level local

[1] An in-depth description of the anatomy of the human eye and the subsequent processing stages is given by Matlin [27], pages 47–71.

[2] The organization of the framework is not described in literature and reflects an own contri-

properties of the data that may become relevant in subsequent processing steps. Because of this profound importance, the sensor design needs careful attention.

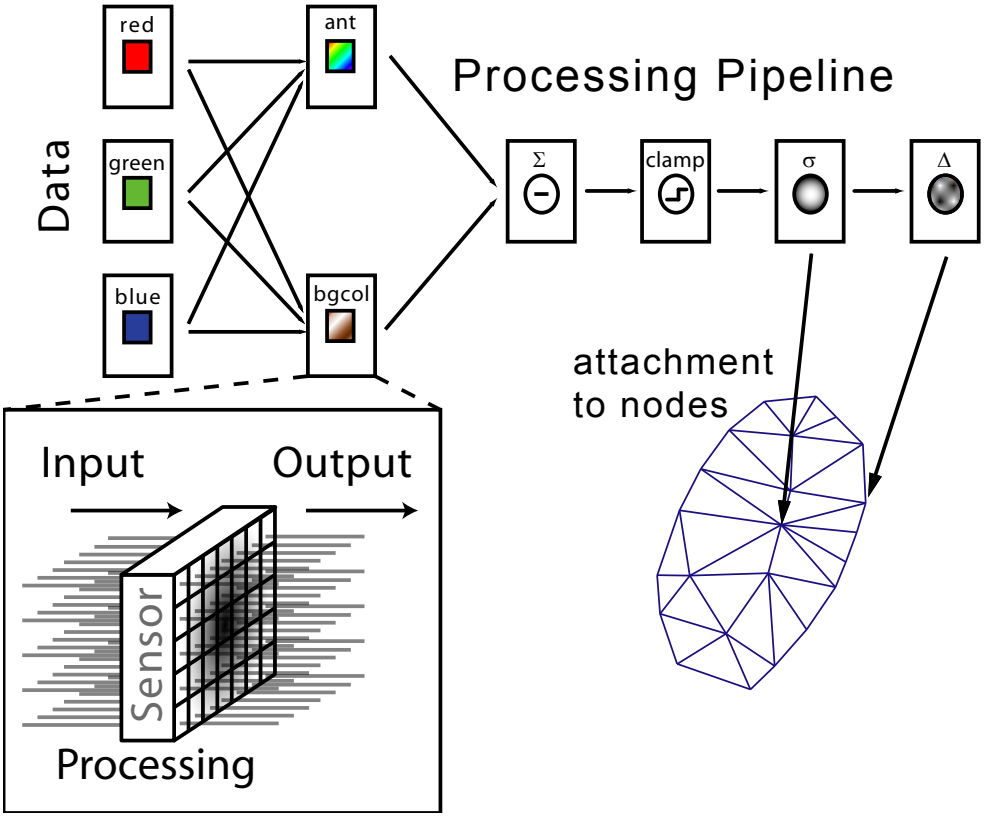


Figure 3.1: **Sensor processing pipeline** showing the constituents of the feature extraction framework. The lower left corner illustrates the interface of a sensor.

The important parts of the sensor framework are depicted in Fig. 3.1. The interface of a sensor is fairly simple. Scalar or vectorial values are passed into the sensor. The processed information is then passed on for further analysis. The sensors are the "eyes" of the deformable model. They feed in the information from the image that the model is trying to adapt to. Different sensors are attached to different nodes. Depending on what role a node plays in the object its sensor is designed for a specific task. For instance, in most images it makes sense to use a gradient magnitude sensors for nodes along the contour of an object.

The sensors are implemented as filters that return scalar values at each point

bution of the author. Despite of this fact this section has not been moved to the concept chapter because it covers a number of well known feature extraction methods that are not actually novel and thus belong more to the 'basics' of image processing.

of the grid. To make the system more flexible each sensor can be the source for an other one. The image itself is a sensor too, returning the unfiltered values at full resolution. It is the default source for new sensors. Plugging sensors one behind the other allows one to flexibly create a large range of feature detectors. To speed up the processing the sensor fields over the entire image are pre-computed. This allows a quick look-up as the model is moving over the image.

3.1.1 Types of sensors

The following set of sensors is chosen to extracting basic features while looking at different scales:

- Data sensor – basic sensor, just passing samples from the image into the preprocessing framework,
- Gaussian smooth of varying scale – using Fourier convolution theorem (see § 3.1.2),
- Gradient magnitude sensor – used to detect edges,
- Corner detector – uses Jacobian determinant to extract local curvature (see below),
- Multi-channel combination sensor – produces weighted sum of input,
- Saturation sensor – extraction of scalar color features from 3D input vector,
- Multi-channel gradient sensor – edges in color images (see below),
- Mahalanobis distance sensor – computes distance to a statistical distribution (see § 4.2),
- Mapping sensor – performs various kinds of mappings, e.g. min/max interval normalization or Gaussian normalization (see § 4.2.1).
- Preprocessing sensor – simply cache values and derivatives from input sensor to speed up processing,

This set of sensors may be used in arbitrary combinations just as illustrated in Fig. 3.1. This can be used to produce fairly complex feature extractors. More sensors for texture features or other elaborate preprocessing steps are imaginable and may be plugged into the framework if needed. The setup is controlled by an external script file.

The corner sensor makes use of the Jacobian of the intensity distribution. This is the matrix of second partial derivatives:

$$\mathbf{J} = \begin{bmatrix} \Delta_{xx} & \Delta_{xy} \\ \Delta_{yx} & \Delta_{yy} \end{bmatrix} \tag{3.1}$$

Sometimes the determinant of this matrix is referred to as *Jacobian* as well. The name denotes the matrix of derivatives of a multi-variate function. If it is the derivative of the gradient of a scalar function (as in our case) it is also called the *Hessian*. We are using the *Jacobian determinant* as an indication for *local curvature*. See § 6.1.1 for illustrations to the effects of this method.

The multi-channel gradient sensor computes Euclidean norm between adjacent vectorial (color) values and uses the resulting finite difference in x- and y-direction as 'gradient'. The magnitude of this gradient is then used as an indication of local edge strength. An illustration of the effect of this method is provided in § 6.1.1.

The sensors may be used in different ways. The idea that has to be kept in mind is that the extracted features may have some specific spatial relationships to each other (e.g. forming a contour) that may be exploited later on when matching a deformable shape to the data. Thus, the sensors serve different purposes. Some of them are performing gradual (or 'fuzzy') classification based on specific properties of the desired objects. Others, such as edge and corner detectors, help to extract geometric properties from the input intensity distribution.

3.1.2 Multi-scale pyramid

For many of the above features and especially for the geometric ones it makes an important difference at which scale of resolution they are extracted. The data may be transformed to different resolutions by applying a convolution with a low-pass filter kernel, such as the Gaussian. The extension of the filter kernel (or width of the Gaussian) decides about the coarseness of the scale. The larger the kernel, the lower are the remaining frequencies in the image signal, which yields a coarser scale.

The problem with large kernels is that the convolution in spatial domain is expensive to compute. To reduce the computational costs of this operation the Fourier convolution theorem is commonly used:

$$\begin{aligned} F\{g(x) \otimes h(x)\} &= F\{f(x)\} \cdot F\{g(x)\} \\ F\{g(x) \cdot h(x)\} &= F\{f(x)\} \otimes F\{g(x)\}, \end{aligned} \tag{3.2}$$

here, the $\cdot$ operator indicates multiplication and $\otimes$ stands for convolution of the two operand functions. This equation states that a convolution of two functions in the one domain directly corresponds to a multiplication in the other. The complexity of convolution in spatial domain is $\mathcal{O}(nm)$ for n being the number of pixels in the image and m being the (smaller) number of elements in the convolution kernel. As the costs for the Fast Fourier transform are $\mathcal{O}(n \log(n))$ and multiplication is $\mathcal{O}(n)$ the overall operation takes $\mathcal{O}(n \log(n))$. If the input image has around

1000×1000 pixels the method outperforms spatial convolution for kernels of edge lengths greater or equal to 5. This is especially often the case when computing various levels of the Gaussian multi-scale pyramid.

Another important application for smoothing is to extend the spatial radius of the sensors. This is of particular interest for the following discussion of matching shape to images.

3.2 A physical shape model

An introduction to physically based modeling can be found in various sources ranging from physics textbooks over articles in particle-based computer graphics [3] to online course notes [44]. This section is going to outline the theory and the driving equations behind the physical model. As there are no extreme velocities involved it is sufficient to stick with classical Newtonian mechanics. There are different ways to formulate the principles of motion. Two common approaches used for deformable models for object recognition or computer graphics (e.g., cloth modeling) are the Newtonian and the Lagrangian. The latter is a scalar energy based approach that has been used in [33].

For simplicity the derivations in this thesis will be kept to the classical force based method of Newton. The geometry information we are working on consists

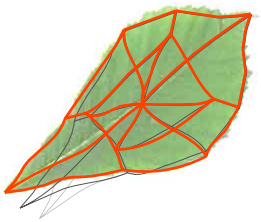


Figure 3.2: **Geometry of a deformable model** showing the model of a leave while moving its last node to the corresponding corner feature. The edges are acting as springs preserving 'natural' smoothness constraints on the shape.

of *nodes* and *edges* connecting the nodes. As the illustration in Fig. 3.2 shows, the nodes explicitly model the contour of the objects and the position of features in the interior of the objects. The entire model is actuated using a particle system approach. The nodes are treaded as particles – point of position $\mathbf{x}_i$, velocity $\mathbf{v}_i$ and a mass m_i. The index i indicates that there is a number of nodes and these properties are stored for each node separately. Mass points of no extension are a common method of abstraction in physical modeling. It is possible to use this approach for real world objects using their center of mass as position and assuming all relevant forces to work directly onto the center.

The edges are acting as springs. Static parameters, describing an edge between the nodes i and j, are its rest length s_{ij} and its spring constant $k_{s_{ij}}$. Its dynamic properties are the stretch $\mathbf{d}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ and its current length $\|\mathbf{d}_{ij}\|$.

Because of the springs, each node does not just exist and move isolated in the data set. Instead, it effects its adjacent nodes resulting in a complex dynamic system. The springs maintain a certain spatial relationship between the nodes. This is the key to model shape information. A more detailed description on how to sensibly construct a set of nodes and edges will be given later. The question discussed now is how to put the whole system to life.

3.2.1 Differential equation of Newtonian motion

The important parameters to compute the further trace of a particle from a given time step have already been introduced. The following equation is known as Newton's first law

$$\mathbf{f} = m\mathbf{a} \tag{3.3}$$

$$\ddot{\mathbf{x}} = \dot{\mathbf{v}} = \mathbf{a} = \mathbf{f}/m \tag{3.4}$$

$$\dot{\mathbf{v}} = \mathbf{f}/m, \text{ and } \mathbf{v} = \dot{\mathbf{x}} \tag{3.5}$$

The number of dots above a symbol indicate it to be the first or second derivative of the variable with respect to time. This has been introduced by Newton and is commonly being used in physics. The two subsequent equations are just conversions from Eq. 3.3 that are closer to the way the computation can be realized in a computer program. These steps transfer the second order differential equation into two first order equations in Eq. 3.5. How to solve these is described in § 3.2.3.

By applying Eq. 3.5, the method of modeling particle dynamics consists of the following steps:

1. compute forces $\mathbf{f}_i$
2. update $\mathbf{x}_i$ and its derivatives
3. perform one step in solving the differential equation

3.2.2 Computing forces

As apparent from Eq. 3.3 the driving quantity behind each movement is the force. In a spring-mass model the force acting on a node i can be split up in the following components

$$\mathbf{f}_i = \mathbf{f}_{internal_i} + \mathbf{f}_{external_i}. \tag{3.6}$$

If internal and external force balance out the model comes to rest[3]. Then the system is said to have reached equilibrium. The internal force comprise all forces

[3]Actually, it would continue to move uniformly if no data is found. To prevent this and the effect of oscillation a damping force is introduced.

resulting from the spring model

$$\mathbf{f}_{internal_i} = \sum_j \mathbf{f}_{s_{ij}} + \mathbf{f}_{c_{ij}} + \mathbf{f}_{t_{ij}}, \tag{3.7}$$

consisting of the spring force $\mathbf{f}_s$, the damping force $\mathbf{f}_c$ and the torque force $\mathbf{f}_t$. All forces are related to a certain spring connecting nodes i and j indicated by the double indexing. They are defined like this:

$$\mathbf{f}_{s_{ij}} = -k_{s_{ij}}(\|\mathbf{d}_{ij}\| - s_{ij})\mathbf{d}^0_{ij}, \tag{3.8}$$

$$\mathbf{f}_{c_{ij}} = -k_{c_{ij}}\left((\mathbf{v}_i - \mathbf{v}_j)\cdot \mathbf{d}^0_{ij}\right)\mathbf{d}^0_{ij}, \text{ and} \tag{3.9}$$

$$\mathbf{f}_{t_{ij}} = -k_{s_{ij}}(\Delta\phi)\cdot {\mathbf{d}^0_{ij}}^{\perp}. \tag{3.10}$$

The spring constant k_s fulfills the conditions $k_{s_{ij}} = k_{s_{ji}}, k_{s_{ii}} = 0$. The analogue applies to the damping constant $k_{c_{ij}}$. Using Eq. 3.7-3.10 it is possible to calculate the internal forces of the model. Examples for external forces are gravity, wind, attraction or any other kind of force field. A discussion of how to set up the external force field is given in § 4.3.1. To get a first impression of the dynamics in the system it is sufficient to compute $\mathbf{f}_s$ leaving the other forces zero.

3.2.3 Solving the ordinary differential equation

At this point we know the equations governing the movement of the particles (Eq. 3.5). From the previous section we additionally know how to compute the spring force and its companions to make the particles interact with each other. Together with the parameters of the system that have already been introduced, this is all the input needed for evaluating the equation. Now, the problem is restricted to compute its output $\mathbf{x}$. Due to the complexity and the dynamic behavior of the forces it is not possible to explicitly compute the trajectory of a particle (the trace of its position over time). Instead, the computation is done iteratively integrating over small time steps. The method used here is the Gaussian method:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t\, \dot{\mathbf{x}}(t). \tag{3.11}$$

Applying this to Eq. 3.5 we end up performing the following two steps:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t\, \mathbf{v}(t), \tag{3.12}$$

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t \frac{\mathbf{f}(t)}{m} \tag{3.13}$$

It is possible to implement the equations exactly as they are stated here. The only thing to remind is that besides the position of the particles we need to store their velocity as well. Alternatives to the described method are so-called midpoint and Runge-Kutta to only name a few. The midpoint has also been implemented in the

framework. These methods have less error compared to the Gaussian method and thus tend to be more stable. The advantage of the method we use is, besides its simplicity, the fact that we have to evaluate the forces only once. If instability or imprecision tends to become a problem it is possible to replace this simple method by more sophisticated ones.

Chapter 4

Concept

After having pointed out the issues with existing approaches in chapter 2 we are now going to discuss consequences for our approach. Since the incorporation of structural knowledge with shape detection is a rather unexplored area, we had to come up with new solutions at several different points. They key points of these novel ideas are the subject of this chapter.

The recognition system is developed along the unsolved issues with existing approaches as summarized in § 2.4. Most important problems addressed are:

1. the data needs to be prepared for subsequent matching of shape information,
2. shape matching gets trapped in *local minima*,
3. *multiple shapes* are required to allow for more general shape distinctions,
4. shapes have to be composed to flexibly describe *structure of complex objects*.

The major goal is to establish an *intelligent layer* that sensibly controls the process of shape matching. Besides forming more complex structures, *synergy effects* of shape searches shall be exploited.

The whole system is developed along the application to ant databases. Insects have distinct anatomy with certain commonness among one family and distinct structural variation between the families. Most images in these databases are taken from standardized perspectives. Nevertheless, the data is a challenging segmentation task because of the different quality of the data, changing posture of the insects and occlusion of body parts.

Inspired by the insect databases a structural shape matching model could greatly improve abilities of content-based image retrieval in general. We have started with a rather theoretical motivation of extending the abilities of existing object recognition approaches. The databases provide a convenient area of application for the new system. In the following, a model is developed considering the following aspects:

1. incorporate *existing approaches* as far as possible and overcome their limita-

tions by

2. following the idea of sub-shapes and their *structural relationships*.

3. Making use of the *characteristics of the data sets* from the application on the one hand and on the other to

4. keep the approach *general* and applicable to further domains.

Despite of these rather ambiguous goals the final system will have its limitations. This raises an important question that should guide the design of the system: What assumptions have to be made for the model? In what situations is it going to work? When is it going to fail and why? In the following, the idea behind the system will be elaborated in detail – ever looking at the assumptions made at each stage and the limitations arising from them.

4.1 Architecture of the system

As pointed out in the previous chapter the automated processing of information from images can take place at different levels. Our system (shown in Fig. 4.1) is taking this into consideration. The following list describes the basic constituents of the system that are depicted in Fig. 4.1 in more detail:

- **Feature extraction** prepares the data by detecting places of interest for subsequent processing steps. The result of this process will be the only information about the data passed on to the higher levels.

- **Shape matching** deforms the shape to adapt to the local conditions in the data set. The basic idea is that the algorithm is looking for similarities between feature constellations in the data and shape properties of the model.

- **Structural knowledge** in form of spatial relationships of sub-shapes is predefined by the user and subject to further automatic analysis by the algorithm. The results of this analysis are applied to *guide the search*.

- **A search strategy** is needed at different stages of processing. A local search carries out the fitting of shape information to the data. A more global search directs this local search to potential locations by taking structural knowledge into account.

The following sections take a closer look at the issues and possible solutions that occur at the different stages of processing. This discussion of possible ideas results in decisions for the implementation.

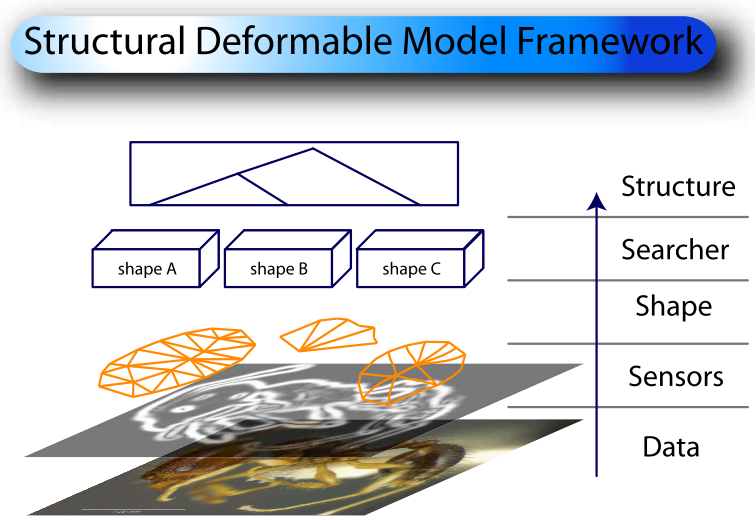


Figure 4.1: **Overview of the framework.** The system consists of a hierarchical organization of different layers of processing. The image data as the bottom most level is analyzed by sensors. The extracted features are passed on to the matching of deformable shapes. Searchers control the evolution of shapes to avoid influence of local minima. Structural knowledge is connecting searchers that are inspiring each others. In addition, the searcher provides input to form structural hypotheses.

4.2 A sensor to detect ants

The general sensor framework has already been introduced in § 3.1. There, we have already made a distinction between sensors that are extracting the object itself and the ones that are emphasizing geometric features. The remaining question is, what is the object?

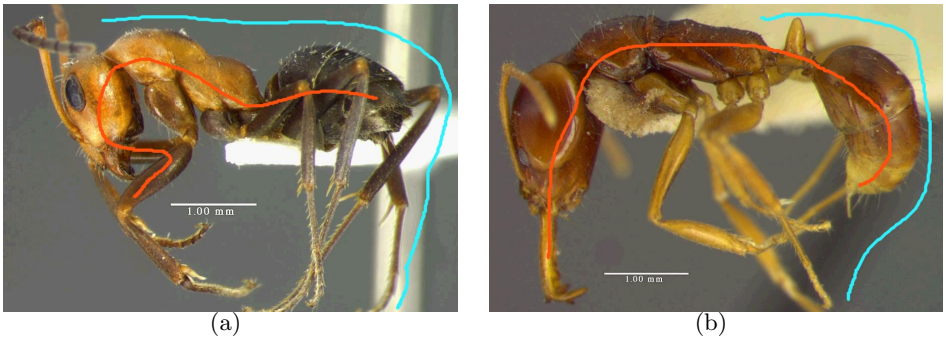
A profound idea behind the method of deformable models is to operate on scalar data without the need for pre-segmented silhouettes. The only prerequisite for the models to work is that they are getting *significant* features as input. This means that their spatial arrangement is revealing at least some recognizable shape properties of the object. If we have a choice of extracting all possible edges from a color image or only the ones that are relevant for our recognition task, we will surely decide to reduce irrelevant input as much as possible.

Considering color information as a method to locate ants in color images, we found that there is no discriminant variation in body color among classes. Beyond that, there is considerable variation in color within families of ants. Nevertheless,

ants exhibit a certain similarity in color that may be exploited for segmentation. Due to the acquisition method, most background in the images is grey (with variations towards black and white). This suggests to further employ a background model.

There is no previous work on detecting bodies of ants by color. Fortunately, a considerable amount of research has been done on the detection of human skin in color images (see § 2.1.2). The adaption of human skin recognition methods is fairly straightforward. The important difference is that one major problem in general skin recognition algorithms is the compensation for varying illumination. This is not the case for the images of the ants since they have mostly been taken in a standardized environment. The other point is that the variation of ant skin color is way larger than the one of humans, because evolution has created a great variety of species.

The observations in the images indicate some kind of *statistical relationship* between the color and the classification of an object as belonging to an ant or to the background. As already considered in § 2.1.2 we have decided to make use of a *parametric model of Gaussians*.



(a) (b)

Figure 4.2: **Acquisition of color samples from ants and background.** A similarly sized set of colors is taken from each class (ant body/background). The sample lines are deliberately chosen to equally pass typical and less typical places.

To build a statistical model, a representative distribution of colors of the two classes has to be acquired from the database. Therefore, we have used a method of guided sampling as depicted in Fig. 4.2. This approach has been chosen instead of an exhaustive tagging of the entire region in the images. To reduce the insufficiencies of the selective manual sampling we have tried to similarly cover all 'normal' regions and peculiarities as well. Specular highlights on the bodies of some ants have been avoided in the acquisition. This will lead to a misclassification later on. This is not a profound problem for our approach because the shape model later on is capable to cope with insufficient data. In our case we have used over 7000 samples of body colors and over 3000 background samples from about 40 different

images chosen to span a wide range of differently looking ants. For the *statistical*

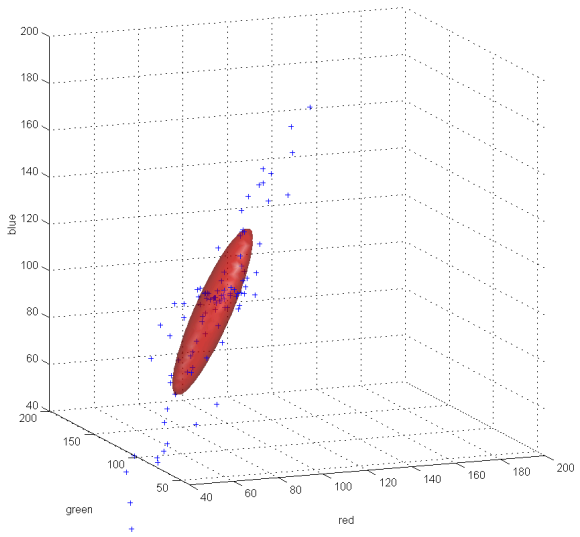


Figure 4.3: **Multivariate Gaussian distribution.** This is an example distribution of colors. Its mean and covariance matrix have been used to construct a Gaussian distribution that is overlaid as transparent isosurface at a constant distance of 1σ from the mean.

analysis an average color and a covariance matrix around the average needs to be extracted. This is all information that is needed to characterize a class by an ellipsoidal multivariate Gaussian distribution. An example illustration is provided in Fig. 4.3. The equation forming such a distribution is

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^N \|\mathbf{C}\|}} e^{-\frac{1}{2}\left[(\mathbf{x}-\mu)\mathbf{C}^{-1}(\mathbf{x}-\mu)\right]}, \tag{4.1}$$

where $\mathbf{C}$ is the covariance matrix of the statistical distribution, μ is the mean or so-called model vector of the distribution. The expression in brackets in the exponent is the *Mahalanobis distance* of the distribution. A classification based on the Mahalanobis distance is more efficient. It is linearly related to the log probability of a sample to belong to the Gaussian distribution.

Fig. 4.3 shows a single Gaussian distribution. The model that we use consists of two separate models for ant body and background. The classifier for ants is obtained by computing the Mahalanobis distance towards the two distributions. If the distance towards the body colors is less, a pixel is being classified as belonging to an ant. This corresponds to clamping the distribution of ant colors at the point of *equal error ratio*[1].

[1]This is the point (or area) where two overlapping distributions have equal probability.

As outlined in the introduction, the method we use does not rely on correctly segmented shapes presented in binary images. Instead, the goal is to deal with insufficient and inhomogeneous data. To reflect the uncertainty involved with the classification, the scalar probabilities are passed on to subsequent processing. The classification is achieved implicitly by thresholding all probabilities of the ant colors at the point or equal error ratio towards the background probability. The result is an image of scalar probabilities that is passed on to further processing.

4.2.1 Sensor Normalization

Different features not only show up at different scales of resolution but in very different dynamic ranges. For instance, depending on the filters involved, the pure intensity feature can range from 0 to 255 and the corner detection in the same image results in values from -0.01 to $+0.01$. As the values of these sensors are used later on to guide the model (see § 4.3.1) the corner sensors would be totally unimportant in comparison to the intensity sensors. Another issue with different sensors is that each type of sensor has a distinct meaning, so they are not always comparable. These two reasons demand to apply *normalization* and *weighting* of the sensors.

One way to achieve this is called Gaussian normalization, as used by Iqbal et al. [19]. This means to compute statistics for a feature e_k over the entire image range observing the expected value $\mathsf{E}(e_k)$ and the standard deviation $\mathsf{S}(e_k)$. The normalized sensor e_k^{GN01} is computed by

$$e_k^{GN01}(\mathbf{x}) = \mathsf{max}\left(\mathsf{min}(\frac{e_k(\mathbf{x}) - \mathsf{E}(e_k)}{3 \cdot \mathsf{S}(e_k)}, 1), 0\right). \tag{4.2}$$

The min and max operators are providing a non-linear mapping of the transformed values. This clamping makes sure the sensor values range within $[0, 1]$. In our implementation another kind of normalization has been considered as well:

$$e_k^{01} = \frac{e_k - \mathsf{min}(e_k)}{\mathsf{max}(e_k) - \mathsf{min}(e_k)}. \tag{4.3}$$

This normalization adjusts sensors to have equal influence on the model relative to each other. It is now possible to perform another weighting reflecting the semantical importance of each feature.

The drawback of both approaches is that they require a statistic to be computed over the entire image. An alternative would be to just incorporate an analysis of all sensor values that have occured so far, re-adjusting the mapping every time a new maximum or minimum is encountered. As we are only working on static images this idea is not relevant. Therefore, all sensor values are pre-computed and stored to allow for faster lookup. Nevertheless, it might be a useful idea for video data, where a computation narrowed to the places of interest is necessary

to save resources. The adjustment of sensor responses is applied to make them comparable to each other. This is of particular importance as they are posing forces to the nodes they are attached to.

4.3 Finding shapes

When discussing existing approaches to modeling shape information in the previous chapter, a distinction between *statistical* vs. *dynamical* modeling and *implicit* vs. *explicit* shape construction has been made. We have decided to use dynamic models of explicitly modeled geometry. The method that has been realized, has been outlined in § 3.2. Certain advantages and drawbacks shall shortly be highlighted here. The simplicity of the design speaks for the explicit models. To create such a model the user simply has to place the nodes directly to the features one considers to be relevant. This involves an intuitive understanding and handling of the model. Since its behavior is based on physical rules we are able to apply some of our real-world experience when judging about the processes seen on the image. This point is important for the later incorporation of user interaction for further adjustments since modifications work directly on the representation. Against explicit geometry speaks the fact that it tends to be instable and might collapse. Possible solutions to this problem are described in § 4.3.1.

The situation is reversed for implicit models. Other implicit parametric representations can be made robust by only allowing parameter combinations that produce valid instances. The problem with designing a specific parametric model is that it might not be general to fit to other problems. A very flexible representative of implicit models are medial axis representations. These are actually worthwhile to consider as a possible improvement of the method we propose. So far we have decided to stay with explicit models for the reasons listed above. They can be made stable by a number of workarounds that are discussed later on.

4.3.1 Balancing sensor forces

At this point we have almost all information that is needed to get the deformable shape detection started. In § 3.2 the task of fitting a shape model to the data has been formulated as a problem of balancing *internal* and *external* energy. This is realized by making the model react to the according forces as shown in Eq. 3.6. Earlier it was explained how to calculate the updates for the model at each time step. What remains to be discussed, is the setup of the external forces that are used for two different purposes:

1. to attach the model to the image and make it adapt to the feature characteristics, and
2. to externally control and guide the model if it has to be guided to a different

area as decided by a higher processing layer.

A detailed discussion of suitable sensors is given in § 3.1. They are constructed to result in high responses at locations of high relevance and low or zero output at less interesting places in the data. An adjustment that makes the sensors comparable among each other is described in § 4.2.1 (indicated by the 01 exponent in e^{01}). Based on this information that is input to each node, a force dragging the node towards the relevant features can be computed:

$$\mathbf{f}_{external_i} = b_{vs}\Delta e^{01}(\mathbf{x}_i) \tag{4.4}$$

$$b_{vs} = b_{vitality} \sum_i \|\Delta e^{01}(\mathbf{x}_i)\|. \tag{4.5}$$

Here, the scaling factor b_{vs} is used to control the influence of the gradient of the normalized sensor of the node. It introduces a fixed balancing of image forces towards internal forces. For that purpose, the sum of the sensor gradient magnitudes over all nodes is scaled to be $b_{vitality}$. As b_{vs} is being re-calculated at every time step it is like a local weighting of sensor influence, which is comparable to local thresholding or similar techniques. The effect of this step is to make the model act with the same kind of *vitality* everywhere in the data set.

Imagine the model is residing in a low-feature area and one node sensor is receiving a blurred weak signal from a close feature. As this information is the only thing the entire model is sensing, the reaction to it (according to the force derived from it) will be more vivid. The model is being dragged towards this position hopefully getting in contact with more features at other nodes. Conversely, in an area of high valued features, shape information would become less important as the movement of most nodes will entirely be governed by the force from the sensor gradients making the spring forces negligibly small. Thus, the described method stabilizes the behavior of the model over the data set.

Another purpose of applying external forces is to influence the model behavior from outside. It is possible to control the geometry explicitly by simply placing the nodes at their new positions. This manipulation has an immediate effect and is used, for instance, to spawn a new model at a given position in the data. The other opportunity of interacting with the model is to apply additional external forces – either as a force field existing over a longer period of time or as a short impulse. The basic set of supported model transformations includes

- noise – shifting nodes randomly (Here is a little trick involved: the random force applied to a node is weighted by the length of its shortest adjacent edge to prevent distorting the model too much.),
- rotate – apply force tangentially to the concentric circles around the center of rotation,
- move – all nodes are pushed into one direction,

- attract towards a center – using negative attraction (repulsion) as well.

For the impulse-like transformations it is important to scale them according to the time step taken by the update in the moment the transformation is applied. An interesting aspect of the force-based model manipulation is that sensors that have a good feature match will encounter a stronger counterpoise. This makes them keep in place while other nodes at less satisfactory positions will start moving to new places.

Preventing the collapse

One problem of the explicit formulation of geometry is that it might collapse under the influence of external forces. Nevertheless, we have decided to use explicit shapes because workarounds exist. Fig. 4.4 shows the problem of collapsing along

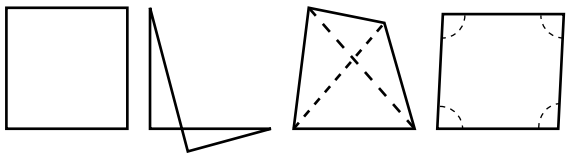


Figure 4.4: **Problem of collapsing shape.** This problem is inherent to explicit shape modeling. (left) original shape, (left middle) degenerate shape in a locally stable state, (right middle) stabilization by introducing supporting edges, (right) stabilization through torque forces.

with two possible solutions. The explanation for this event is that there are other states where the internal forces are balanced but the resulting shape might be very different. All four objects of the illustration are meant to be instances of the same spring mass model. The quad on the left shows the initial position. Since there is no influence keeping up the orthogonal edges the shape is easily disrupted as shown in the second figure from the left. The third indicates a possible solution by adding additional springs in at least one diagonal of the quad. Thus, the number of invalid balanced states of the spring model is reduced or they become less likely. This means whatever contour run our shape will have – we have to construct additional internal edges to preserve the characteristic run of the contour. To reduce the need for internal edges we have included torque forces at the edges. Their computation has been described in Eq. 3.10. The basic idea is to have rest-angle. In accordance to the rest lengths of the springs, a torque force is imposed on the adjacent nodes if the angle of the corresponding edges is deviating from the rest angle.

Adding torque forces leads to more stability for long edges. For short edges the situation is a little more complicated. Imagine having a sequence of short edges.

A single angular deviation resulting from local bending has to be distributed over a lot of small joints. This distribution of the deformation will take some time. Additionally, to reduce deformations of short edged contours, the angular spring would have to be made very stiff. Unfortunately, especially for short edges this leads to numerical instability. For that reason neither supporting diagonals nor torque force provide a complete solution to the problem of collapsing. Fortunately, in combination they yield pretty stable structures.

4.3.2 Comparing solutions – Quality of fit function

So far one important problem of dynamic models has not been addressed. As discussed in the review of related work, one major drawback of most deformable models is their tendency to get stuck in locally optimal solutions. In the description of the framework, we have already mentioned a higher-level control mechanism that prevents the shapes from getting stuck. An important input any of these higher level search or control mechanisms will need is an indication for the *quality of a shape*. This means, if multiple different solutions for a given deformable model exist, we will need some measure that indicates, which one has done a better job in matching to the data.

The introduction of a *quality of fit* function is a profound difference to the brain controlled deformable organism by Hamarneh et al. [16]. They follow one approach and incorporate domain specific knowledge to guide its exploration. Their model most likely needs a measure for current confidence as well. The information about the quality somehow provides a termination criterion. Nevertheless, an explicit statement of how such a measure may be constructed is not given.

The following derivations seeks to evaluate two different properties: the degree of the deformation of a shape and its success in managing to adapt to the features of the data. The first important internal shape descriptor is the length ratio r_l. It characterizes a global stretching or squeezing of the model that is defined as follows:

$$r_l = \frac{\mathsf{E}\left(\{\|\mathbf{d}_{ij}\| : i,j \in \mathbb{N}, a_{ij} \neq 0\}\right)}{\mathsf{E}\left(\{s_{ij} : i,j \in \mathbb{N}, a_{ij} \neq 0\}\right)}, \tag{4.6}$$

where a_{ij} is indicating adjacency between two nodes being 1 if an edge is defined or 0 if not. The result can be understood as the ratio of the average edge length is divided by the average rest length of the springs. If it is 1 all compressed springs balance out with other stretched ones. A value smaller or greater than 1 indicates that the model is respectively squashed or stretched by the conditions in the data. Based on that, a measure for the deformation q_d of a model is defined as:

$$q_d = \frac{\sqrt{\mathsf{E}\left(\{(\|\mathbf{d}_{ij}\|/r_l - s_{ij})^2 : i,j \in \mathbb{N}, a_{ij} \neq 0\}\right)}}{\mathsf{E}\left(\{s_{ij} : i,j \in \mathbb{N}, a_{ij} \neq 0\}\right)} \tag{4.7}$$

To understand this equation omit r_l at first. The counter of the fraction is the

root mean squared difference of the actual edge lengths to their rest length. This is essentially what we use to indicate deformation. The rest of the equation is just normalizing this value. The denominator is putting the result in relation to the average rest lengths. Thus, bigger models will have a deformation comparable to smaller ones. Finally, putting r_l back in, the actual lengths are scaled to compensate for global stretch or squeeze. Thus, a model that has found a slightly smaller self-similarity in the data set and has squeezed in successfully will not be penalized if it maintained proper proportionality among the edges.

Another measure that is important when looking at a model is the placement of its sensors at relevant features in the data. This sensor fit q_s is computed

$$q_s = \mathsf{E}\left(\{e_i^{01} : i \in \{1, \ldots, n\}\}\right), \tag{4.8}$$

where e_i^{01} are the normalized sensor intensities introduced in § 4.2.1. Now, having an indicator for the internal conditions of a model, its shape deformation q_d, and one for the external fit of its sensors to the data q_s, an overall quality of fit q can be computed as follows

$$q = w_s q_s + (1 - w_s) e^{-\|q_d\|}. \tag{4.9}$$

This is a convex combination of the two criteria that have been introduced. To put different emphasize on either deformation or sensors the weight w_s is set to a value between 0 and 1. The exponential of the negative squared q_d is just a trick to map it to the interval $[0, 1]$. Since q_s is in that interval as well, the result of this equation will be a value between zero and one. Since a number of normalizations is involved in computing the quality measure q it is now a useful tool to compare solutions of different models that each try to fit to the data.

4.3.3 Improving Geometry

A model editor is provided with our framework to manually generate a shape model. Sensors can be designed interactively, nodes and edges are constructed and set up using the GUI (see § 5.3.2).

As the discussion in § 2.2.3 has already pointed out, it is relatively easy to obtain *some* set of parameters that produce satisfactory results by default. The problem though is to set up their parameters *correctly*. It is typical for statistical models to improve as more training input is provided. It would be desirable to obtain a similar behavior for a physical model. As already pointed out, approaches to this problem exist. Here, another simple method of reducing the arbitrariness of an initial guess of physical parameters is described.

The existing approaches are both based on statistical models. Physically based models are just analyzed in parallel to obtain more stable initialization or a regularization of the search process. This means our particular method of staying within the physical model and refining it statistically has not appeared so far.

The basic idea is simple and described in a few words. Some deformable model is initially constructed and applied to different datasets. It might need some guidance and manual refinement to produce a perfect match. But in general, it is way more convenient to use than manual placement of landmarks. The confirmed correct matches of the deformable model are stored as training data.

The statistics applied from the data are just local length variations of the springs. No co-variation is investigated. Nevertheless, this is not generally impossible. A solution to consider would be to internally relate correlated springs forming higher order multi-springs, which is corresponding to the modes of variation.

Despite of this tempting possibility, the goal of this analysis is to improve the physical model as it is (and not to modify it until it becomes a statistical model). For that reason the local length variations are obtained along with the average length of the edge. This information is then used to change the spring parameters.

$$\begin{aligned} s_{ij} &= \mathsf{E}\left(\|\mathbf{d}_{ij}\|\right) \\ k_{s_{ij}} &= \frac{\mathsf{E}\left(\|\mathbf{d}_{ij}\|\right)}{\alpha \mathsf{S}\left(\|\mathbf{d}_{ij}\|\right)} \end{aligned} \tag{4.10}$$

The rest length is set to the average length of the edge among the training data and the spring constant is set to the inverse of the standard deviation scaled by an influence factor α. The latter adjustment needs some tweaking though. If an edge hardly changes its length or even is the same length all the time the spring constant would be very high. This would cause *numerical instability* of the method. Because of that the resulting spring constants are clamped against a maximum value. The effect of the influence factor α is evaluated in § 6.2. The result of this refinement operation it that it helps to determine places in the shape that are more likely to be deformed than others. This does not influence the shape in a way that is as versatile as a completely statistical model would be. Nevertheless, the resulting setup of the springs is improved at least a bit. The fact that some springs have become weaker and others got stronger bears the potential to more sensibly distinguish between 'good' and 'bad' deformations.

4.4 Guiding the Search

The deformable model explained in the last section performs a *local* matching of its shape to feature constellations in the data. That means if the model is placed sufficiently close to the target structure it will correctly drag itself to the final position. Only very few interactions are required to have a complex model placing all its nodes at the right positions in the image. In very ambiguous and distorted images expert intervention might be the only way to guarantee correct results. The approach of dynamic models nicely facilitates an opportunity for high-level control

layers, such as automatic systems or human experts. In our approach the latter is employed to assist the creation of models and the first is the *automatic search engine* that we are going to look at now.

The task of the search engine is to control numerous dynamic models on their way through the data. Each model can be seen as a specific approach to fit its shape to the data. The searcher starts new approaches at places of high expectation and in parallel canceling mislead approaches. Thus, it keeps an evolving population of rivaling approaches each trying to optimally fit its shape to the features in the data.

The system presented here makes use of some of the mechanisms of Genetic Algorithms and stochastic sampling as discussed in § 2.2.4. The general idea is to perform optimization by stochastically sampling the search space and to incorporate some kind of memory of past successes and failures to guide the search process.

In the following, the important aspects of our method are highlighted. The first step is to reduce the cost of searching by *restricting the search space*. Secondly, an appropriate method for *sampling* is discussed. Thirdly, an efficient and representative *selection of winners* is discussed to provide hints for further exploration. Finally, *expectation maps* are introduced to incorporate prior knowledge to narrow the search to areas of high expectation and to provide a method of input to the search engine.

4.4.1 Reducing the search space – the property vector

The *full search space* for a geometric model is a concatenated vector consisting of the (x, y) coordinates of all nodes. The goal of the searcher can be understood as determining significant maxima in the *quality function*. As defined in § 4.3.2 this function returns a scalar quality measure for a given embedding of points. Directly searching this space would most likely not yield any useful result as the number of possible node combinations is unmanageably big.

The idea followed in our approach is to *split the effort* of searching. On a more detailed level the *local search* is performed by the dynamic model that has been described in § 3.2. In a certain local environment it is automatically converging to an optimal solution. Simultaneously, it is maintaining smoothness constraints on the coordinates of the points. This leaves the searcher to guide the local searches on a coarser scale.

The global search is then performed on the four-dimensional space of (c_x, c_y, s, ϕ), where c_x and c_y are the coordinates of the centroid $\mathbf{c}$ of a shape, s is an indication of the size of the shape, and ϕ is the angle of its orientation. A representation of a shape in this space is called its *property vector*. The direction ϕ is evaluated using the centroid and a specially marked direction node. The standard deviation of the nodes from the centroid is used as a 'radius' indicating the

size s. It is possible to extract the property vector of a given set of points. Also, the points can be transformed to comply with a given property vector. The transformations are *translation, rotation and scaling around the centroid*. It is convenient that these three operations can be carried out independently and in an arbitrary order, always yielding the same result.

As a shape adapts itself to the data its property vector changes, its internal arrangement of nodes is changed accordingly. This deformation is not reflected by the property vector. Thus, the high dimensionality of the search space is hidden from the searcher. A shape develops distinct deformations when adapting to a local environment. If the searcher places two differently deformed shapes at the same (c_x, c_y, s, ϕ) they most likely – but not necessarily – converge to the same solution. This fact has to be taken into account when placing the models.

4.4.2 Stochastic search with memory

Spawning a deformable model in the dataset means to select a specific incarnation of the model and to transform it to fit a given property vector. As explained above the dimensionality of the search space is reduced to four. The incorporation of differently deformed models adds more hidden dimensions to the problem making a stochastic search more efficient than a regular sampling of the search space[2]. To improve the quality and reliability of the results it is possible to make some modifications to the original method. Previous knowledge about the domain that is being sampled can be incorporated by choosing specific probability distributions when generating new samples. Another improvement can be made by keeping track of information gained from past samples. This history can be used to improve success when spawning new samples. Both improvements are discussed in more detail in the following two sections.

The technique of Genetic Algorithms mentioned above has a certain relation to stochastic sampling. The important difference is that for GAs new samples are generated by recombining old ones only adding slight mutation (randomization) to the next generation. By increasing the mutation rate, the permuting effect of crossing specimen becomes less important, yielding more a stochastic search.

The property vector can be understood as the genes of a model. Actually, in GA it is more typical to use discrete numbers to encode a problem. The use of real valued vectors more alludes to the GS approach. Anyways, the different components of the vector have distinct meanings for the model and makes up its most important properties. For a given set of genes it is possible to spawn a deformable model. The model will immediately start local optimization adapting its deformation to the data. At some point it will come to rest in a local optimum. At this point the searcher can query its fitness (or quality) using the result to make

[2]This kind of sampling is known as *Monte Carlo method*. Its accuracy is similar to regular sampling for four-dimensional problems and outperforming it for a dimensionality greater than 4.

decisions on how to proceed to find the most significant optima.

4.4.3 Efficient clustering algorithm

As the shape population evolves it is important to select winners from time to time. The resulting list of winners is sorted by fitness and serves two different purposes: 1. the winners are the *output* to a higher-order control instance, and 2. the winners may have a larger influence on the further *evolution* of the population. The winners are chosen to have a locally optimal fitness and to appropriately spread over the search space. They are chosen representatives of a spatially coherent environment in the search space. Thus, the problem of finding local winners can be understood as a *clustering problem*. For our application the choice of winners has to be consistent and fast. To achieve that, two different methods have been taken into consideration:

1. using a fixed *grid of bins* equally distributed over the feasible search space,
2. extract *ranked clusters* of shapes from best to worst generating new clusters if a certain distance to the fittest representative is exceeded.

For the *fixed binning method* the search space is divided into quadratic cells along the x and y components of the centroid. An example of such grid is shown in Fig. 4.5. To generate enough winners (at least one per object) the distance of the bins is set to less than the radius of the best shape. Then each shape is allotted to the according bin and the adjacent bins. The best model within a bin is the chosen representative and thus a local winner. This approach works fine if two assumptions hold: 1. if a bin contains *correct* and *incorrect* matches the fittest is a correct one, and 2. there are no multiple objects of one shape class in one bin (assumption of *no occlusion*).

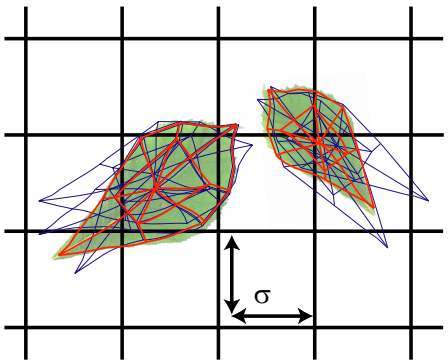


Figure 4.5: **A regular grid of bins** is used to determine local winners. The grid size is chosen to be less than the radius of an average shape (using the standard deviation σ of its point distribution).

Unfortunately, for cluttered images it is not realistic to assume correct fitness maxima and absence of occlusion. The method of *ranked clusters* has a better ability to cope with these situations. The principle is described by the following steps:

1. generate a *ranked list* of the population sorted by fitness,
2. pick (and remove) fittest member from the ranked list making it the *representative* of a new cluster,
3. remove all members from ranked list that are within a given distance from the representative and *merge* them to the cluster (see discussion of performance issues and suitable distance measures below),
4. proceed from step 2. until ranked list is empty or fitness is below a certain ratio from the best.

If old winners (from a previous ranking) are merged into new clusters the old cluster-IDs are recovered to maintain temporal coherence in cluster naming. The worst case complexity is $\mathcal{O}(n^2)$ if each of the n members of the population is making up an own cluster. To reduce the computational costs the *fixed bin grid* described above is used during merging to quickly prune shapes that are too far off in the search space. Now, every shape is only considered for merging as often as different clusters fit into one bin. The number of clusters per bin is bounded because the size of the shapes has a lower bound. By setting the bin size accordingly this number can be made both, small and constant, reducing runtime to only $\mathcal{O}(n)$ for the clustering.

Another criterion for the performance and the quality of the clustering is the *distance measure* involved in comparing similarity of shapes. The following list comprises different measures from a coarse to a more detailed level (with increasing computational costs). The choice of maximum norm in preference to Euclidean or average distance is that it is more sensible to variations in details rather than slight overall differences.

- Euclidean distance of *centroids*,
- distance of property vectors: maximum norm of *weighted component differences* (see definition of d_{cmp} in Eq. 4.11 below),
- maximum distance of *corresponding points* – only applicable for two instances of the same shape,
- distance of *closest points* – for each point in one shape the closest point in the other shape is chosen looking for the maximum distance within these pairs[3].

[3]Using the largest distance to the closest points is as well known as the *Hausdorff distance*, taking the average distance is known as the asymmetric *Chamfer distance*

When comparing two shapes based on their property vectors each of their components (position, scaling, orientation) have to be treated separately. The distance measure used in our work is composed as follows:

$$d_{cmp}(A,B) = \max\left[\frac{1}{\omega_c}|\mathbf{c}_A - \mathbf{c}_B|,\ \frac{1}{\omega_s}\left(1 - \min(\frac{s_A}{s_B}, \frac{s_B}{s_A})\right),\ \frac{1}{\omega_\phi}|\phi_A - \phi_B|\right], \quad (4.11)$$

where $\mathbf{c}_{A/B}$ are the centroids of the shapes A and B, $s_{A/B}$ are their sizes, and $\phi_{A/B}$ their orientations. The weights $\omega_{c,s,\phi}$ put different emphasis on each component. During the merging step all shapes that have a $d_{cmp} < 1$ to the representative are added to the cluster. The actual size of the clusters is then determined by setting up the weights $\omega_{c,s,\phi}$. In our implementation these weights are set to $\omega_c = 0.15 s_R, \omega_s = 0.3, \omega_\phi = 0.5\pi$, where s_R is the size of the representative of the cluster. The weights have been chosen intuitively. Increasing or decreasing them results in more or less winners. For the further processing it is important to have enough winners, so the 'correct' matches will be among them. If there are too many winners the further processing will slow down, but will not fail to work. Hence, no further attempt has been taken to justify the cluster sizes as long as they are small enough. The $\omega_{c,s,\phi}$ have been incorporated in such a way that the cluster sizes adapt to the properties of the representative.

To further speed up processing it is desirable to remove unnecessary shapes. There are two different kinds of models that can be removed:

1. redundant shapes – that have a low distance of corresponding points to the representative are considered to be identical and are thus *melted* together,
2. lazy shapes – that have not become winners and have already got stuck in a local minima (indicated by a low overall point speed).

The melting to a model is noted in an instance counter. This provides an indication for the attractiveness of a certain solution that may be used for an additional voting mechanism in the search. Only shapes below a certain distance d_{cmp} are considered for melting, reducing the need for computing the more expensive corresponding point distance.

Shapes that have been selected as winners are used to spawn new generations of shapes. Mutations are done to the position, the size, the orientation and to the points of a model. Each with a separate probability and as well with decreasing rate of distortion that is exponentially decreasing over time. This method helps to distribute shapes with certain deformations and properties that have been proven successful. It is possible to reduce the number of mutations and entirely rely on newly migrating shapes that are generated from given random distributions. In that case the best winner is chosen as prototype for newly spawned shapes. The list of winners is used to locally focus attention to places of previous success. This allows one to incorporate *acquired knowledge* about the fitness distribution during the search.

4.4.4 Expectation Maps

An approach to incorporate *prior knowledge* into the search process is to choose the probability distributions used to generate new samples in the search space. This technique is known as *importance sampling*[4]. Insights from previous searches or training data (see § 4.5.2) may be incorporated here to focus the attention in new searches.

In our method an *expectation map* generates the samples in the search space. It is a collection of probability distributions each covering a certain range in the search space. A simple example of such a map is depicted in Fig. 4.6. During search different hypotheses are contributing to this map. The distributions are weighted differently to reflect the degree of expectation according to the generating hypothesis. Having no knowledge corresponds to a uniform distribution over the entire (bounded) search space.

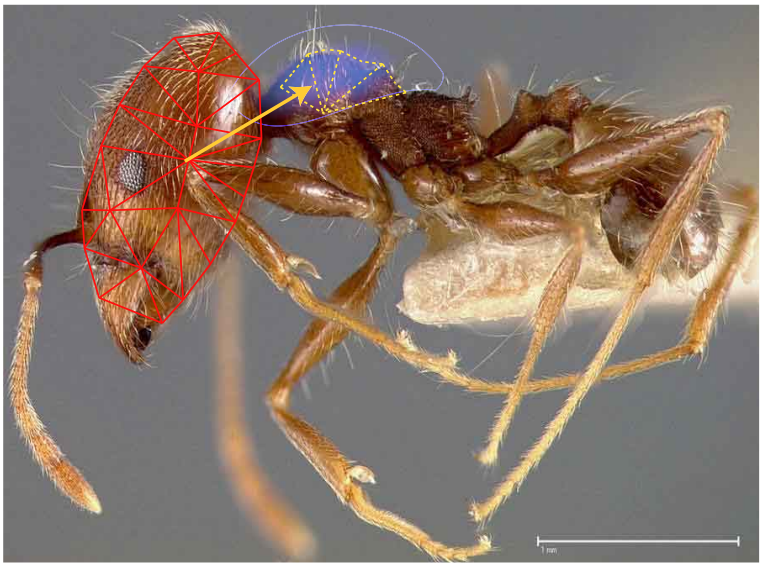


Figure 4.6: **Expectation Maps** are passed to the searcher to indicate areas of higher anticipation. The image is showing *Pheidole carribaea sloanei*. Based on the position of the head that has been found successfully (red), a search area (blue) for the back can be estimated. The dotted yellow line shows the resulting match for the back. Inversely, expectation maps are as well used to rate spatial relationship between shapes.

Different types of distributions (over the multivariate search space) are available that may be used to represent various kinds of hypotheses:

[4]This method is commonly used in computer graphics for instance when integrating reflectance distributions for photo-realistic illumination calculations.

- rectangular (uniform) distribution,
- Gaussian distribution,
- transformed distribution – using any of the other distributions as input and performing an affine transform,
- combined distribution – a weighted overlay of a number of input distributions.

The distribution combiner is used to collect all kinds of input distributions. It is for instance used to represent the expectation map itself. The first two distributions are axis aligned and may be modified by interposing a transformer.

Another important feature of the expectation map is its *inverse use* to *rate* a given property vector. If a shape has to be rated, a list of references sorted by the degree of correspondence to structurally adjacent shapes is generated. This facility enables the searcher to take both into account – the fitness of a shape and its spatial relationship within given hypotheses.

At this point all important characteristics of the search engine have been illuminated. The basic idea is to have an entity that autonomously performs the search using a compact interface:

> **input:** 1. reference shape with dynamic characteristics (spring model), 2. dataset, 3. expectation maps to reflect certain hypotheses,
>
> **output:** ranked list of winners with assessment

The search is a process evolving over time. It is possible to dynamically change the expectation maps as the exploration yields new insights. Underneath, the dynamic system is doing the job of matching deformable shapes. Using the search engine as an additional layer it is now possible to operate on a higher level – generating hypothesis and evaluating them based on the solutions suggested by the searcher.

4.5 Structural Models

The system described so far is capable to detect simple objects of varying shape. A dynamic system is employed to enable and, at the same time, constrain shape deformation. Additionally, it performs a local adaption to the features contained in an actual image. An additional layer organizes local searches returning a ranked list of locally optimal solutions. *Structural knowledge* is employed to supervise the local searches by exploiting spatial relationships between different deformable shapes. Thus, it is possible to analyze complex objects by considering both – its structural variation and the variation of its shape constituents. This method provides two important advantages. The first point is that the variation of singular shapes can be reduced making the shapes more specific. The second advantage is that concave shapes are more difficult to handle in deformable shape matching.

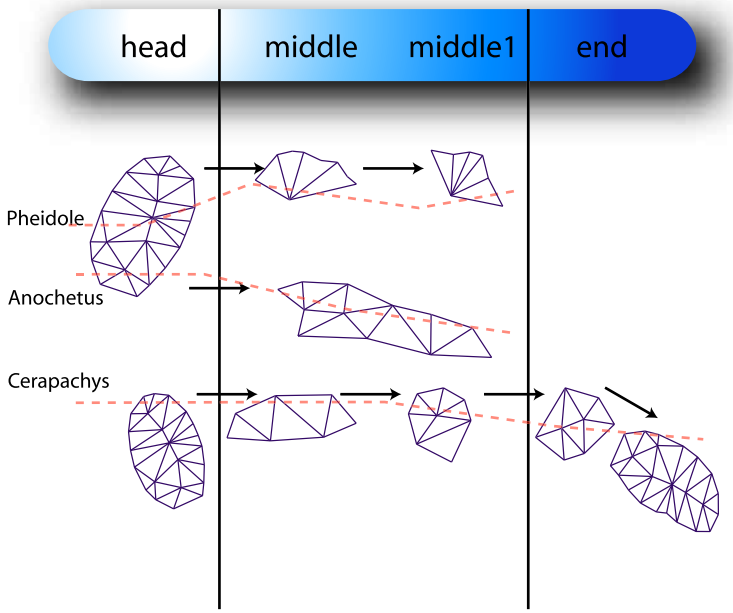


Figure 4.7: **Ant graph** showing all important parts of the structural representation.

Decomposing these into a spatial arrangement of multiple convex shapes may make the recognition process more robust.

In the following a method for representing a structure of deformable shapes is being discussed. After that a searching method for complex deformable shapes is introduced. Finally, the output of the structural search is interpreted resulting in a ranked list of possible structural shape compositions.

4.5.1 Representing structure

The structure shown in Fig. 4.7 illustrates the three constituents of our structural model. The *shapes* are the building blocks of the system. The arrows indicate that *spatial relationships* between these shapes exist and may be analyzed. Additionally, they are interpreted as *rules* when building the structure. It is now possible to construct all kinds of higher order structures using these rules. To finally match to a known object *interpretation sub-graphs* are given (indicated by the dotted red lines). The tabular organization of the figure is only used to improve understanding of the structure. The parts that actually play a role for the program are the shapes,

the spatial relationships and the interpretation graphs.

In addition to the structural representation, the system is supplied with a set of labeled training images. These images contain examples of shapes forming certain structures. The training data is acquired by manually guiding the search process. Thus, it is not necessary to place all landmarks by hand. Instead just a little bit of guidance has to be provided to the dynamic search system.

After the spatial relations of interest are defined and a first set of training data is produced, it is possible to evaluate the transformations. This means to look at the pairs of shapes that relate to each other (indicated by the arrows in Fig. 4.7). Each placed shape has a property vector (as introduced in § 4.4.1). A transformation between two shapes A and B represented by $(\mathbf{c}_A, s_A, \phi_A)$ and $(\mathbf{c}_B, s_B, \phi_B)$ is computed as follows:

$$\begin{aligned} \mathbf{c}_{A\to B} &= \frac{1}{s_A}\mathbf{R}_{-\phi_A}(\mathbf{c}_B - \mathbf{c}_A), \\ s_{A\to B} &= \frac{s_B}{s_A}, \\ \phi_{A\to B} &= \phi_B - \phi_A, \end{aligned} \tag{4.12}$$

where the operator $\mathbf{R}_\alpha$ performs a rotation around the angle α. In addition to this computation of a transformation vector, a forward and an inverse transform for property vectors are defined accordingly. The identity element to this transform is $(\mathbf{0}, 1, 0)$.

For each spatial relationship a set of transformations between the according two shapes is derived from the training data. From this set an *average transformation* and the *standard deviation* from this average is computed. Additionally, the set of vectors is normalized to zero mean and unit variance and a principle component analysis (PCA) it performed[5]. Both methods, axis aligned and covariance transformed Gaussians, have been implemented. The transformation defined in Eq. 4.12 is chosen to produce highly independent variables. But sill, PCA may be used to further narrow variance.

Another important aspect about this transform is the use of *angular difference*. The rotation is mapped to a one dimensional variable allowing to further reduce dimensionality of variation. The effect is especially notable if a corresponding pivot point is chosen instead of the centroids $\mathbf{A}$. It is possible to specify this pivot point manually by selecting a node of the deformable shape. If this selection is omitted, the centroid is used by default.

When performing statistical analysis on the angles it is important to map them to an *appropriate interval* first. This is done by estimating an average direction α by adding the unit vectors of the set of angles first. Then each angle is mapped to

[5]For this kind of normalization the covariance matrix turns to a correlation matrix having a constant diagonal of $\mathbf{1}$.

the interval $[\alpha-\pi, \alpha+\pi]$. If this transform would be omitted the angular statistics would be most likely centered around a wrong direction.

Finally, besides performing statistics on the examples, it is important to note their quantity in a separate counter. This provides an indication of the *reliability of the analysis*. That way, it is possible to make useful decisions if only a few (or even none) examples are contained in the training data. In the following we are going to look on how to produce hypotheses from these statistics.

4.5.2 Inspiring each other: Generating expectation maps

Having gathered all information to represent a structure, the question remains how to apply this knowledge in a search for structural objects. According to the remarks in the introduction, we are following a two-way approach here. Both, *bottom-up* and *top-down* processes, are employed to obtain a list of possible solutions to a given problem. The underlying dynamic model is using bottom-up constraints that have been derived from the data to suggest deformations. If a deformed shape manages to fit nicely to the data, the search engine is selecting it as a winner and passing it on to a higher level processing layer. At this level hypotheses may be posed based on structural knowledge. This top-down aspect of the approach is going to be discussed now.

The search engine described in § 4.4 is setup by passing in a shape prototype and a set of expectation maps that may focus the search. Expectation maps as introduced earlier are probability distributions within the search space. This means, they provide a preferred position, direction and scaling for newly spawned deformable shapes.

The way the structural representation and the search engine collaborate is the following:

1. each shape constituent in the structure graph is assigned a *shape searcher*, each running in parallel with the others;
2. for each shape a list of winners is obtained from the searcher;
3. based on the winners for one shape and the statistics about spatial relationships *expectation maps for adjacent shapes* are being generated and passed to the search engines;
4. winners are assessed using the expectation maps and the quality of fit function. This results in *ratings for the edges and nodes* of the structure graph;
5. an optimally rated *combination of shapes* is searched for along the interpretation subgraph yielding a ranked list of suitable interpretations;
6. the procedure is iteratively repeated from step 2 until all expectation maps have spawned a predetermined number of samples.

The output of this process is a list of rated structural interpretations. Details about the graph search are given below. The point that shall be discussed here, is the generation of expectation maps. In the entire framework this is the key step to interrelate knowledge about shape organization and search for shape deformation. It is the point where top-down and bottom-up processing meet.

Each winner that is returned by the searcher has a certain quality (or 'fitness'). When generating probability distributions for adjacent shapes, this quality is used to weight the integral of the corresponding distribution. Thus, a winner that performs better is more likely to spawn more new partner shapes than weaker winners. Because of insufficient statistical information it may not be possible to generate appropriate maps. In that case a flat background distribution over the entire search space may be used instead.

To obtain a default placement for shapes, the image frame can be added as a structural constituent. In that case, a property vector for the image is generated in the form $\mathbf{0}, sizeX, 0)$ and spatial relationships are analyzed just as for any other real partner shape. The parameter $sizeX$ is the width of the image.

The basic idea is that the expectation maps are used as a means for the search engines to inspire each other. If one encounters a good match for its shape, adjacent searchers are asked to focus on the corresponding positions. This results in a *self-organizing structural search*. The process evolves over time, improving results until convergence. The remaining task is to determine the most promising combinations of shapes and find the most relevant interpretation.

4.5.3 Matching structure

After applying the steps that have been described so far, the task of matching structural information has been transformed to a graph search problem. The information that serves as input for that is a *set of candidate shapes* for each shape constituent of the structure graph. In addition *rated connections* to candidate shapes in the adjacent structure nodes are provided.

As shown in Fig. 4.7 there are interpretation graphs given for certain shape combinations. This information can be applied in two different ways. One is to find the optimal combination of shapes on the graph and find the best rated interpretation graph on these results. The problem with this method is that a given interpretation only considers certain adjacencies to be important. Unfortunately, if no interpretation is given it is unclear which connections have to be optimized. For that reason a separate graph search is performed for each interpretation only involving the shapes selected along the according sub-graphs. The search is sped up using *dynamic programming*, which is simply caching search results. This is a very important improvement because if for 100 candidate shapes per structure node the number of possible paths to check is 100^n for n nodes in the structure graph.

The question that has been left open so far is how to evaluate *the quality of a path*? Different choices are possible for a method for evaluation. The occurrence of a shape in the structure is conditional on the existence of adjacent shapes. This suggests a combination as conditional probabilities for a certain path. The probability of an edge can be directly looked up from the expectation maps that have been generated from statistics. The probability of a winner is determined by relating its quality to the best quality among all candidates for the shape. This is not a precise measure of probability for a shape, but it is a good enough indication for it[6]. The problem with the conditional probability is that it is very small and very sensible to missing shapes. Thus, a weighted additive combination has been chosen instead:

$$q_p = \sum_i \omega_i q_i + \sum_i \theta_i r(e_{i0}, e_{i1}), \tag{4.13}$$

where q_i is the quality of the shape chosen for a node, $r(e_{i0}, e_{i1})$ is the rating for the spatial relationship between two shapes. This is indicated by the start and end indices e_{i0} and e_{i1} of an edge e_i. Finally, the weights ω_i and θ_i are putting different emphasize on shapes and connections. It is possible to obtain good results by simply leaving these weights at a constant value of one. Making one ω higher than others means to make a shape more important than others. This is particularly useful if some shapes are more reliably found than others. The weights for the edges θ are indicating the tolerance given on a spatial relationship in favor of a better placed shape. Reducing a θ_i is releasing the spatial constraints, which is comparable to relaxing a spring in the deformable spring model. Choosing these weights appropriately can be seen as additional knowledge about the structure that is provided by the user. For that reason the weighting of the graph is included in the structural description.

To make the above method applicable to the comparison of different structural interpretations a normalization is required. The problem is that, depending on the sum of the weights, different maximum scores may be reached. To compensate for this effect, the rating for each partly or fully detected structure is divided by the maximum possible score for the substructure. This allows to relate the quality of small structures towards larger ones. Comparisons of this kind are necessary when assessing different interpretations for an image. When comparing different structure matches within one interpretation, the unnormalized score is used. This leads to the effect that always the most complete structure is chosen, even if subsets may produce a higher normalized rating.

The search is terminated after all expectation distributions have generated a pre-determined number of samples. So far this number is set to a fixed count of 500 samples, which is in any case sufficient. A tighter restriction of the total number of

[6]The value is made of the probabilities of the ant classification. Thus, it is not than wrong to think of the sensor values as probabilities.

spawned samples would significantly reduce the total amount of time for a complete search. So far, the system is more designed towards robustness and quality of the results. A further application in practice would demand for a stricter adjustment of this termination criterion.

4.6 Image retrieval

The goal of content-based image retrieval is to obtain images from a database by using their contents as key information for searching. This allows for instance to retrieve images that have similar properties (and thus similar contents) to the one that is presented as request key.

Using the structural matching engine described above, it is possible to find different structural interpretations for an image – with the best-rated one most probably being the correct one. Based on that, it is possible to compare images using a high-level description. The advantage here is that the images have been interpreted semantically. Thus, subtle but important differences may be used to distinguish different images.

Chapter 5

Implementation

This chapter explains how the parts of the framework have actually been realized. The goal driving the system design was to obtain a generally usable and flexible system with versatile areas of application. Some problems specific to ant databases had to be solved. Another goal was to keep the system extendible for further developments.

5.1 Goals and Design

The source written for this project consists of about 15,000 lines of C++ code not counting external libraries. This is accompanied by several Matlab scripts for prototyping. The entire system is organized in a modular fashion: parts of code that serve different purposes are split into different classes that rely on each other in a hierarchical manner. The guideline for the programming was to produce code that could be re-used in subsequent projects. That means, the program is constructed of separate, possibly autonomous modules.

An appealing quality of deformable models at work is that they are interesting to look at, leaving opportunities for user intervention. Thus, it is important to have *high performance* to maintain interactivity of the whole system. C++ is used for its platform independence and the object oriented programming paradigm. Object oriented programming is a very convenient way of structuring the entire project and producing reusable and self contained modules of code. The program runs under *Linux* and *Windows*.

At several points of the project it was better to make use of existing libraries or foreign code to avoid re-inventing the wheel. The following list covers all foreign code and libraries that are used within the project:

- The Standard Template Library (**STL**) provides complex data types such as dynamic arrays, sorted lists, maps, and a number of useful algorithms to efficiently work on the data. This provides a solid basis for most of the tasks

to be solved. The STL has become part of the standard distribution of most C++ compilers.

- As platform independent graphical user interface **FOX** Toolkit has been used. This toolkit also provides image loaders for the most common formats (such as TIFF, JPG, GIF and BMP). `http://www.fox-toolkit.org`
- To visualize the search process and the results **OpenGL** is used for its platform independence and high performance graphics. This makes it possible to add sophisticated interactive visualisation of the results later on. `http://www.opengl.org`
- GNU Scientific Library (**GSL**) is a huge collection of all kinds helpful code for numerical computation. From this package a sniped of code has been used to generate a random normal distribution. `http://www.gnu.org/software/gsl`
- **FFTW** is a library for high performance computation of the discrete Fourier transform. `http://www.fftw.org`
- **CAM Graphics Class** by Andrew Anderson has been used to generate PostScript output of the geometries for some illustrations. `http://www.math.ucla.edu/ anderson/CAMclass/CAMClass.html`

5.2 The parts of the framework

The general constituents of the system have already been introduced in the previous chapters. Here, a compact overview is provided with a focus on issues related to the actual implementation.

5.2.1 Modular Organisation

The code for the graphical user interface (GUI) is kept separate from the code of the actual algorithms. The program is running multiple threads to separate search and animation from user interaction.

In table 5.1 a list of the most important classes is provided. In addition to that, a couple of little helper classes for 2D vectors, managing a collection of sensors, and multi-threading under Linux and Windows is part of the source code.

5.3 The sensor framework

All sensors are derived from a virtual class. The interface is kept as simple as indicated in Fig. 3.1, allowing for input from multiple sensors and scalar or vectorial output within a given image. The dependencies of the sensors are kept as well to maintain appropriate updates if parameters change at an earlier processing stage.

Sensors are shared among the nodes to avoid re-computation. Additionally, a sensor may pre-compute all its values into a caching sheet. This step is even mandatory if the internal processing requires to process the entire sheet at once. This is for instance the case when the Fourier convolution theorem is applied for smoothing (see § 3.1.2).

Sensors are constructed from a command line. It is possible to modify this command line at the runtime of the program and observe the immediate effect to the screen and the running search.

The organization of sensors in a `SensorCollection` allows to share similar sensors between different shapes. For that reason the names of sensors used in different Models have to be unique. This is most easily achieved by referring the same central sensor configuration file from all model description files.

5.3.1 Geometry and physical model

The class `Model` contains all information that is needed to actuate the spring-mass model described in § 3.2. This means nodes and edges are stored along with their physical parameters (mass, spring constant, rest lengths, etc.). The functions to solve the Newtonian differential equation system as discussed in the context of Eq. 3.5 are in this class. To keep the model alive, its update function is called from a separate thread in equidistant time steps.

To speed up computation, the descriptors of the current state of the model (such as property vector, quality of fit) are only computed once and then cached until the next update of the geometry is performed.

Internally an *adjacency matrix* is used to represent properties of the shape graph for certain processing steps. This data structure is commonly used in graph-based algorithms. In our case it is used to maintain consistency of edge and node information.

Another very important issue that has been pretty much omitted so far is that of varying scale of the datasets. When thinking about appropriate algorithms and evaluating them this is indeed not such an interesting question. To get the implementation to work the different scaling of the images has to be taken care of somehow. The scale bars in the images have been extracted by hand providing a measure of *pixels per millimeter*. Internally pixels are used store positions of the nodes and when computing size of the object. Only if statistical information is passed, e.g., to compare models among different datasets, scale is changed to *mm*. This allows to obtain comparable transformation vectors between the shape constituents of a structure applied to different datasets.

5.3.2 Creating shapes

The basic steps to obtain a deformable model can be summarized as follows:

1. Choose an image that clearly shows the target object the model shall be created for.
2. Create sensors for the most discriminative object features (e.g., edge detector, blurred intensity, coarse corners) and choose an appropriate scale for the sensor (see § 3.1.2).
3. Create nodes along the contour and, if possible, within the object as well.
4. Connect the nodes along the contour and triangulate inside the object.

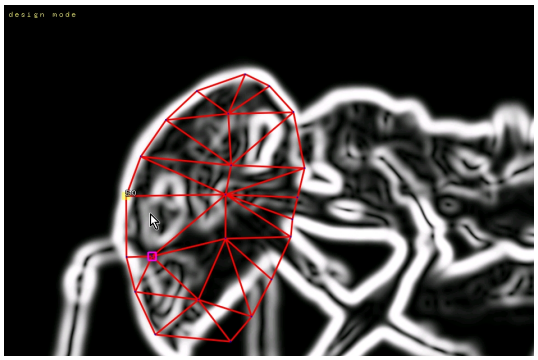


Figure 5.1: **Design mode.** A special mode of the user interface allows to interactively construct geometry along with its set of sensors.

The user interface that is shown in Fig. 5.1 has been included to support the process of shape creation. There are at least two more important sets of parameters to set up, the node masses m_i and the spring constants $k_{s_{ij}}$ for the edges. Initially, these are set to default values ($m_i = 1$ and $k_{s_{ij}} = 0.1$) and can be refined later (see § 4.3.3). Automatic assistance to this process could greatly improve model creation by making it *easier* and *less arbitrary*. The list of steps can be shortened by incorporating automatic procedures to solve certain steps. The aspect of arbitrariness alludes to the fact that a manually built model incorporates a number of heuristic and sub-optimal decisions. Automating some steps might makes them more optimal. Furthermore, an automatic choice of parameters will be repeatable and easier to standardize than a completely hand crafted model. So far the framework takes a user specified model as input and then applies a number of automatic refinements to it. This can already be seen as an intermediate level approach to model creation. The future goal should be to further shift the responsibility for choices from the user towards an automatic system. The intention of this development is not to completely ban user interaction from the system. It should just be restricted to the most significant decisions on a high level control layer not bothering the user with detail questions.

5.3.3 Intelligent search

The basic mechanisms behind the search technique have been given in § 4.4. The implementation is taking care to perform multiple searches in parallel. This means that not only multiple models are being updated in parallel, but instead whole populations of models of different shapes are controlled by different searchers. For that reason the searcher is as well sequentially triggered by a background thread. It is possible to plug in new expectation maps during the search. This may affect the shape population because it will change two important things. The sampling of the search space and the rating of the spatial relationships among shapes of different classes. This means shapes that have no chance to become part of a promising structural relationship are removed from the population.

Each time an evolution step is performed the winners are obtained using the clustering method described in § 4.4.3. Each winner entry contains a list of adjacent winners in other shape classes that have provided a promising rating of the spatial relationship.

It is possible to manually restrict the search range. This is useful when initially placing a model that may have a globally optimal placement somewhere else in the dataset. The user just places the model somewhere near the finally correct position. The searcher will then spawn new attempts restricted to a certain radius. This restriction of the search space is later done by the spatial relationships within the structure.

5.3.4 Managing structure

To represent the structural information three basic classes are important. The `StructTable` controls all information. It contains the shape nodes (`MStruct`) and *interpretation paths*. The `MStruct` node contains links to other nodes. These links are associated with spatial relationships. Thus, they are represented by a separate class `SubStructure`. This is necessary because such links are collecting statistical information about spatial relationships. Later on this information is used to connect the searchers for different sub-shapes.

The interrelation of shapes in a structure is done by generating expectation maps. One structure node returns a list of candidates (winners). These are used in connection with the SubStructure links to generate expectation maps towards adjacent searchers. Different distributions have been implemented for that purpose: uniform, axis-aligned scaled Gaussian, transformed Gaussian. It is possible to switch among different distributions from an outside script file. So far the latter distribution is used because it most precisely reflects the variation of the relationship.

The structure table is the topmost control layer matching different structural interpretations to the current sets of shape candidates. The process of the search is

going on over time, improving as new observations are made. After a certain time it is approaching an optimal and stable state. The termination criterion is based on the number of samples that have been generated by the expectation distributions. Each distribution is assigned a maximum number of samples. If all of them are done the search is stopped and the different interpretations are evaluated.

Class	Description
`Node`	a node in the geometry; has position, mass, velocitiy, attached forces, list of adjacent edges; has a `Sensor` attached
`Edge`	connects two nodes; has parameters for spring model, such as rest length, spring constant, damping constant
`Model`	holds information about geometry (`Node, Edge`), sensors, performs spring model calculation, and supplies all necessary state information to the controlling `Brain`
`Dataset`	handling the data set, its loading, display, and access by the Sensors; derived from `class Sensor`
`Sensor`	virtual class for accessing a source (data set or other sensor) and filtering this input; can preprocess a data sheet for the entire data set; base class for `IntensitySensor`, `SmoothIntensitySensor`, `GradMagSensor`, `CornerSensor`, ...
`Brain`	controls a number of models and their states, manages a separate thread for spring model calculation and decision making
`Searcher`	performs a search for a given `Model`
`ExpMap`	list of `EMDistribution` objects that are generating specially distributed samples into the search space `Model`
`MStruct`	a node of the model structure containing a shape `Model` and links to other nodes, additionally it controls a `Searcher` by providing an `ExpMap`
`StructTable`	manages the complete structure graph, interpretations are used to find optimal paths selecting winners and appropriate spatial relations
`SpeciesDB`	is managing database queries selecting entries by given patterns, capabilities of STL template `map` are used to efficiently obtain query results

Table 5.1: Overview of important classes used in the implementation.

Chapter 6

Results

This chapter is divided into several sections, each presenting the results of a specific part of the framework. Additionally, short discussions are provided relating to the observations.

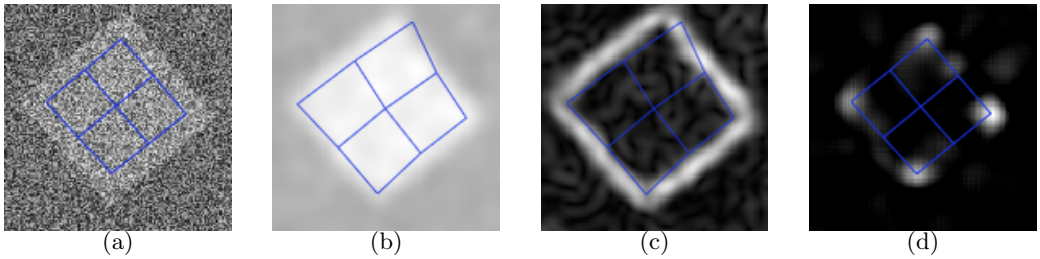
6.1 Feature Extraction

The extraction of features forms the basis of the system. Features of the data that are omitted by the sensors are not available to subsequent processing stages. As a result, this layer has a high influence on the success of the overall system.

6.1.1 Shape features from intensity distributions

The shape prototypes for the matching consist of edges and nodes. Since these have to be placed at corresponding positions in the data, it is desirable to have feature extractors that generate high responses along straight contours, especially around corners. Fig. 6.1 shows the result of some sensors available in the framework. The original image is subject to strong noise artifacts superposing a rectangular structure. Using smoothing most of the high-frequency noise can be suppressed which allows for a clear extraction of edge and corner features. Low frequency noise may still be present, but the remaining signal is sufficient to provide guidance to the attached nodes.

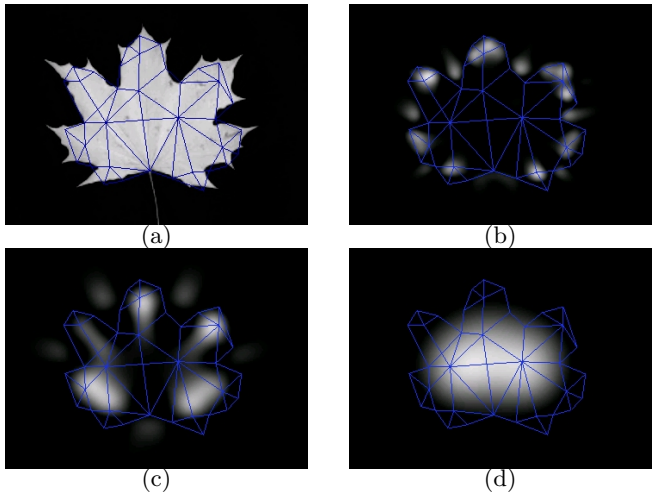
In addition to considerations about what type of sensors are suitable to guide a geometric model, a decision about the scale has to be made. In most cases, changing to a coarser scale increases the radius of influence of the features towards nearby geometric models. The example of a corner sensor, as depicted in Fig. 6.2, is revealing another aspect of scale. The curvature extracted among different scales of resolutions reflects distinct shape features. The hierarchical shape structure of the maple leaf is splitting up towards the outside. This hierarchical splitting is represented by differently placed curvature maxima as scale increases. The nodes



(a) (b) (c) (d)

Figure 6.1: **Different types of sensors**, (a) original image, the blue quad shows the matched shape, (b) Gaussian smoothing, (c) gradient magnitude (edge detection), (d) corner detection.

towards the inside of the shape model are assigned to these basic shape features and provide a coarse orientation of the shape that may be refined by additional nodes towards the outside.



(a) (b) (c) (d)

Figure 6.2: **Corner sensor at different scales of resolution**, (a) original image, the blue contour is overlaid for better comparison, (b) scale 2^6, (c) scale 2^7, (d) scale 2^8.

6.1.2 Detecting bodies of ants

Another problem that has to be solved by the sensor framework is the handling of color images. Since color is a multi-dimensional input, it has to be evaluated prior to the extraction of edges and corners. In § 4.2 a sensor to extract color properties of the bodies of ants has been described. Fig. 6.3 provides an illustration of the two distributions of sample colors that have been extracted from a set of images. For every color that is encountered in the image its distance towards the statistical

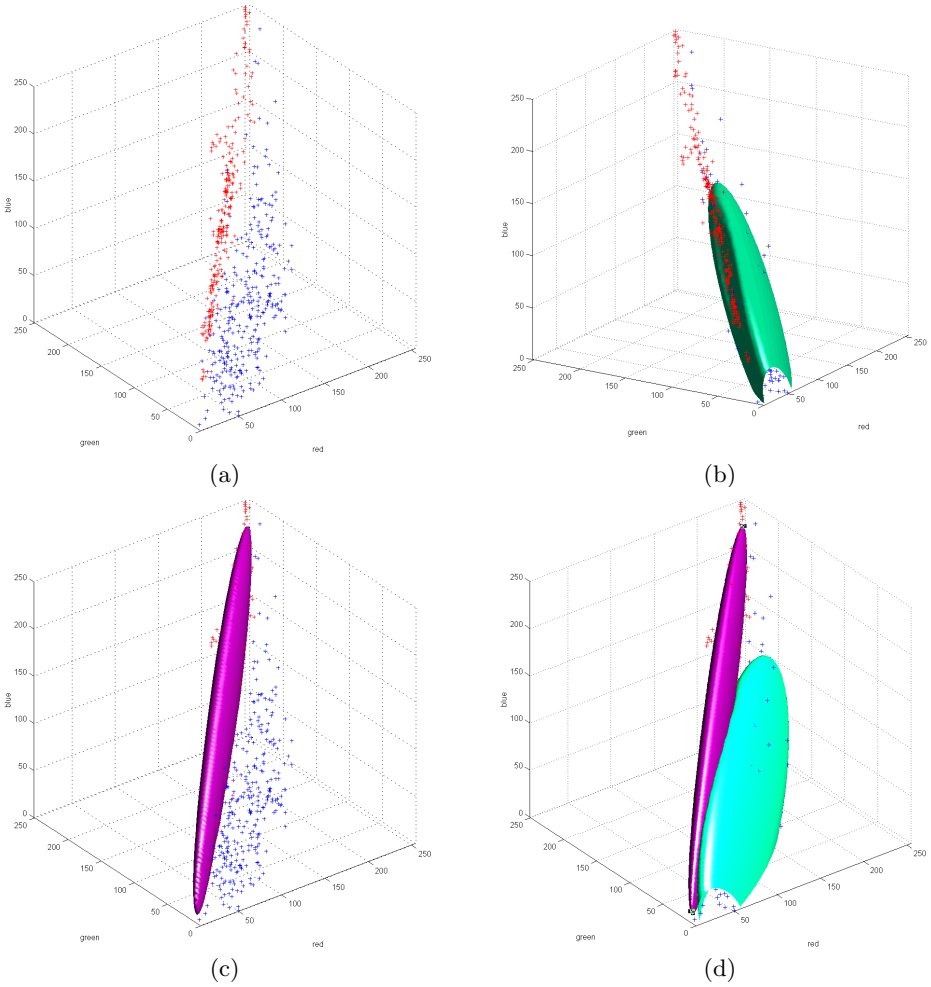


Figure 6.3: **Distribution of ant body and background color**, (a) scatter plot in RGB space (blue crosses indicate ant body, red is background), (b) Gaussian model of the ant body color (iso-surface at $P(ant|c) = 0.05$), (c) $P(background|c) = 0.05$, (d) both. (true positive ant is 96.62%, false negative background is 6.52%, number of ant samples 7067, number of big samples 3403)

sample distribution is computed. The decision boundary is formed by comparing these distances. For multiple class centers, this results in a *blob-like* modeling of the class boundary.

In fact, it might be possible to improve the representation of ant colors by using multiple statistical distributions resulting in a *mixture of Gaussians*. We found that the current model yields satisfactory results. Further support for this simple model is given by looking at the actual sample distribution in Fig. 6.3. The two point clouds appear to be quite separable. The inclusion of the class boundaries further underlines this observation. An evaluation over the given samples results in a 96.62% of true positive ant and 6.52% of false negative background. It would be possible to further influence this ratio by moving the class boundary. Since untruly classified background or foreground are equally decreasing the run of a contour, we have decided to stay with the cut along the equal error rate.

A previous choice of a suitable sensor to distinguish ants and background in the images was the use of a saturation sensor. It can be seen from Fig. 6.3(b) why this approach works. The saturation can be interpreted as the distance of a color from the diagonal in the color cube. The distribution of background colors is pretty much restricted to a certain low radius around this diagonal. An important improvement of the new statistical model for ant colors can be seen in the improved classification for ant colors of lower intensity. Here, a fixed radius around the diagonal resulted in too many dark ant colors being treated as background. Besides this, the generation of a sensor from statistical data has a better justification than the manual setup of its parameters.

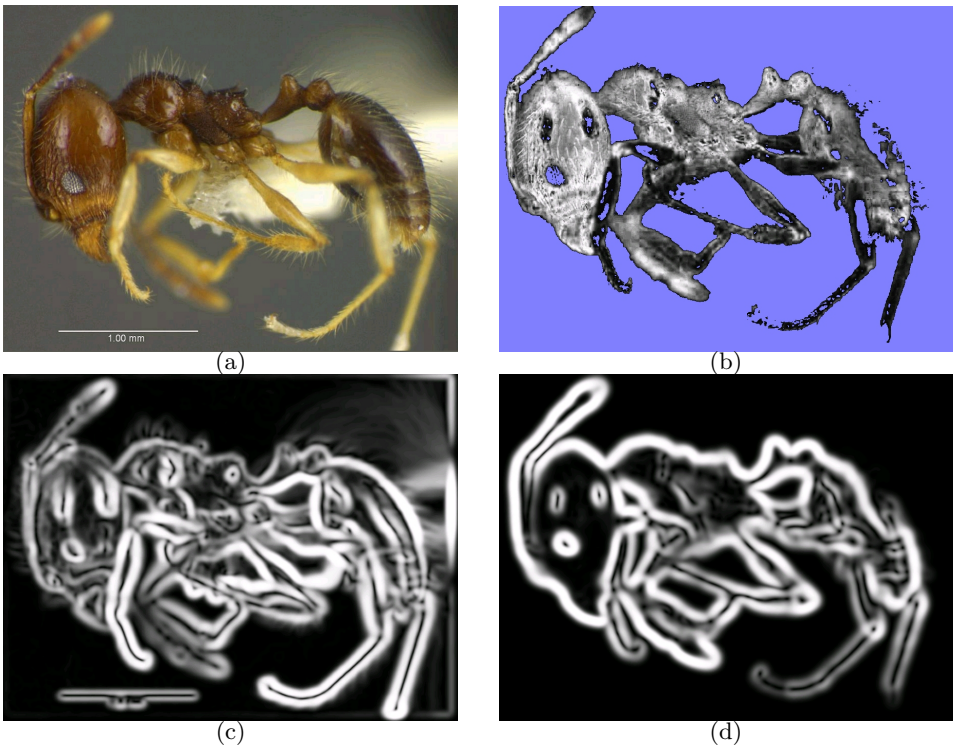
The two sets in Fig. 6.4 and Fig. 6.5 show examples of the successful application of the sensor. The two images in Fig. 6.4(c+d) provide a comparison between the multi-channel gradient sensor (described in § 3.1.1) and edge extraction after the ant sensor. The latter is superior in reducing internal edges that otherwise severely distract the shape matching process.

The major advantage of our framework is that we are not performing a binary classification but instead preserve gradual probabilities for further processing. This allows to more sensibly incorporate shape knowledge to compensate for uncertainties or misclassifications.

6.2 Matching Shapes

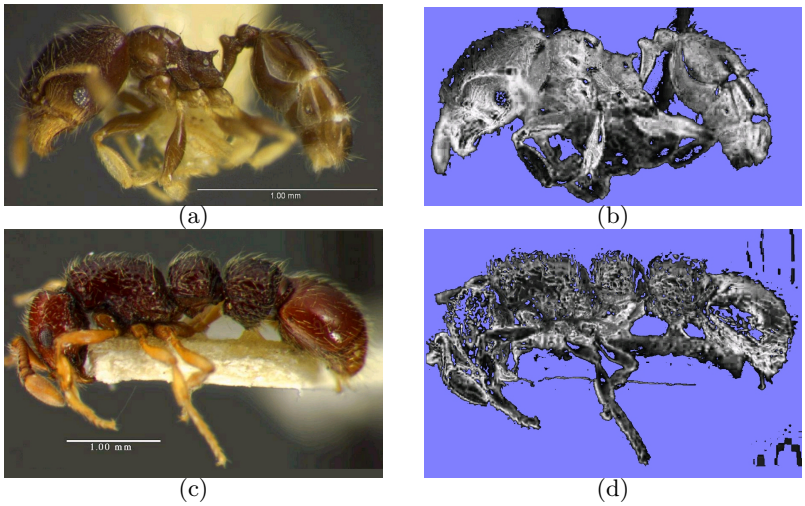
Substitutional for an animation, the image sequence in Fig. 6.6 gives an indication for the convergence behavior of the dynamic model used in the framework. The illustration shows how well the method works if it is placed sufficiently close to the target structure.

A problem of physically based dynamic models is the justification of the setup of their parameters. The discussion in § 4.3.3 has proposed a setup of spring constants



(a) (b) (c) (d)

Figure 6.4: **Results of the ant body classifier.** (a) *Pheidole aenescens*, (b) clamped probability of ant, (c) multi-channel gradient magnitude, (d) edges of (b)



(a) (b) (c) (d)

Figure 6.5: **Examples of ant classifier.** (a,b) *Pheidole albipes*, original and classification (c,d) *Cerapachys vitiensis* original and classification. The small inner holes are closed on a coarser scale.

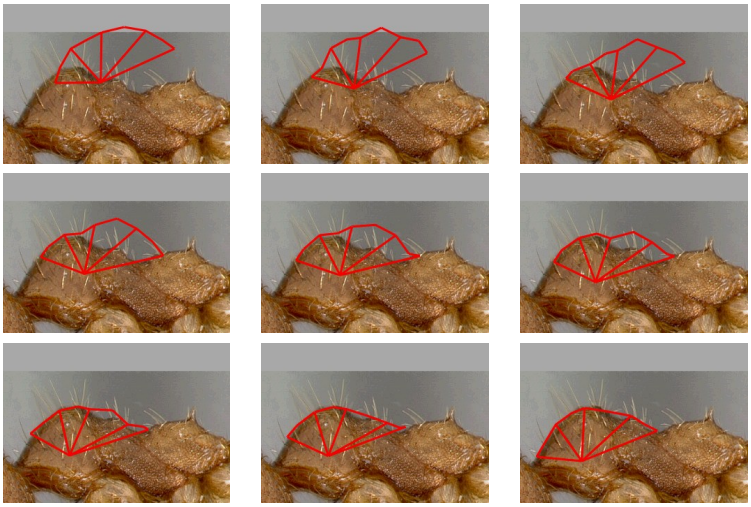
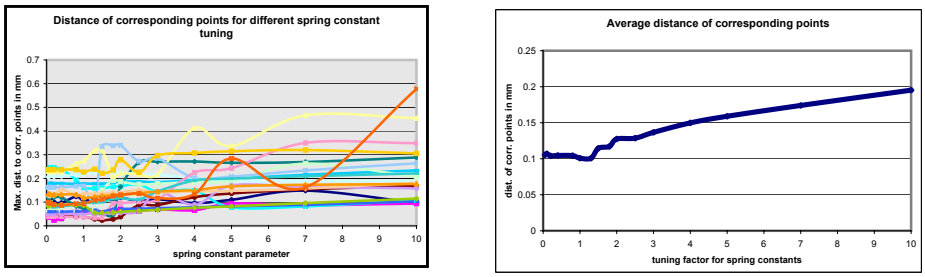


Figure 6.6: **Dynamic model in action.** The image sequence (read in a row-wise order) shows the convergence behavior of the dynamic model. After being placed close to the structure, the forces automatically drag it towards the final position where it reaches equilibrium. The depicted process takes about 5 seconds if real time steps are taken.

by the inverse standard deviation of the edges length in the training data. The actual procedure can further be influenced by a factor α (see Eq. 4.10). The general effect of the adjustment and the influence of α in particular have been evaluated in a series of tests, depicted in Fig. 6.7. The difference in effect between small and large values for α is that a low α yields higher spring constants and a higher value results in more flexible edges having lower spring constants. Because of the clamping against a maximum, possible spring constants with very low values for α result in a model with all edges being equally stiff. As the factor rises, the length variations start to take effect refining the model. At some point the flexibility is extended too much, which results in increasing distances towards the reference shapes. The graph in Fig. 6.7(b) reveals a minimum in the average trend at $\alpha = 1$. Thus, we have chosen to adjust its value at 1. This improves correspondence of the general template towards the training shapes as much as possible.

6.3 Finding Structure

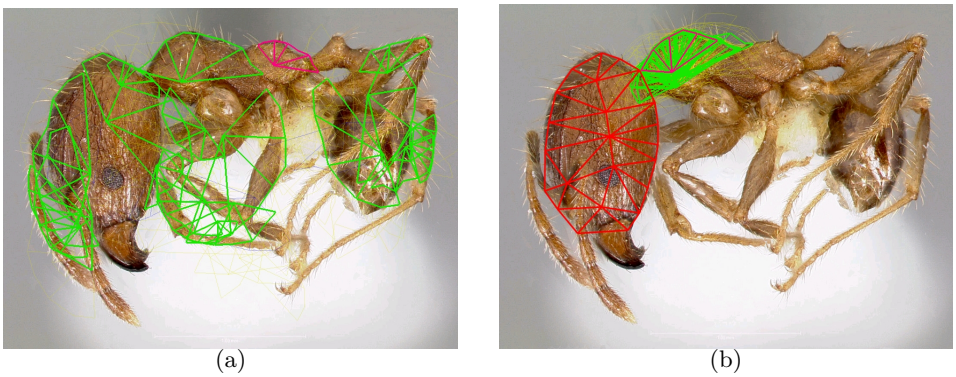
An important point of this thesis is to establish a control instance for multiple deformable shapes. This is realized by applying structural knowledge about the spatial relationships between shapes. The example in Fig. 6.8 shows different outcomes for two different searches on a thorax template. The unconstrained search in



(a) (b)

Figure 6.7: **Adjustment of spring constants** using the adjustment factor α of Eq. 4.10 (a) difference between template and reference shapes for different adjustments, (b) average of distribution show in (a) indicates a global optimum at 1.

Fig. 6.8(a) finds possible winners distributed all over the image. There is nothing wrong to this result since the template just looks for a round shaped contour of an non-empty object. Such round parts may validly be found at all kinds of scales and orientations over the entire image. The result in Fig. 6.8(b) is the corresponding search receiving guidance trough a previously found head model. This constrains the search range. Another effect is an additional structural rating that is provided to later select an optimal combination of head and thorax.

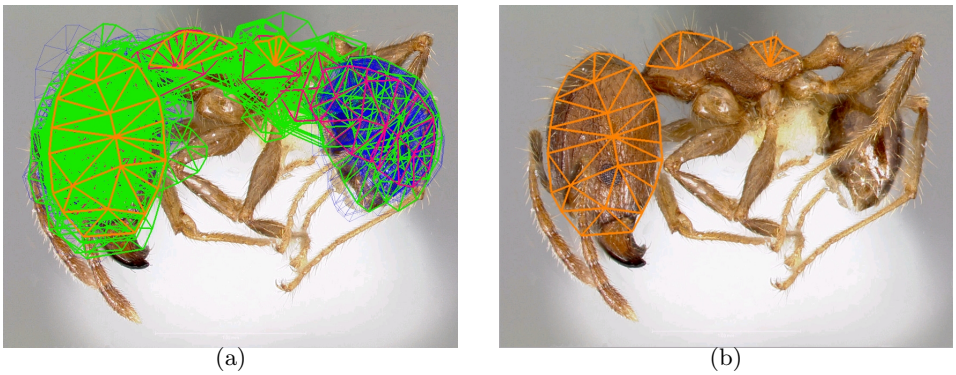


(a) (b)

Figure 6.8: **Structural guidance**, *Pheidole fervens* (a) the single thorax-template is globally applied to the image. Green shapes are local winners after clustering. The purple shape has highest quality. (b) The search is guided using the structural relationship to the head.

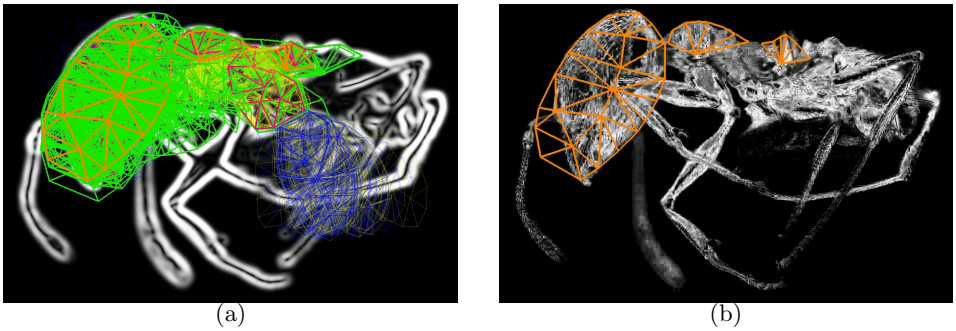
The structural model applied in the following searches consists of three different kinds of ants, Pheidole, Anochetus, and Cerapachys. Each has distinct characteristics that are represented by different numbers of shape templates. An example

of full search is depicted in Fig. 6.9. Among a large set of possible candidates, for each possible structural interpretation, the optimal combination of sub-shapes is selected. The evaluation of the interpretation paths is done by computing a weighted sum of the shape qualities and of the rated spatial connections. Fig. 6.9(b) shows the best rated final interpretation that is correctly recognizing the parts of Pheidole. Another example of a possible outcome for a different species is shown in Fig. 6.10.



(a) (b)

Figure 6.9: **Multi-structure search**, *Pheidole fervens* (a) the ant is correctly classified with a probability of 81% (anochetus 77%, cerapachys 55%). (b) Only the correct match is shown.

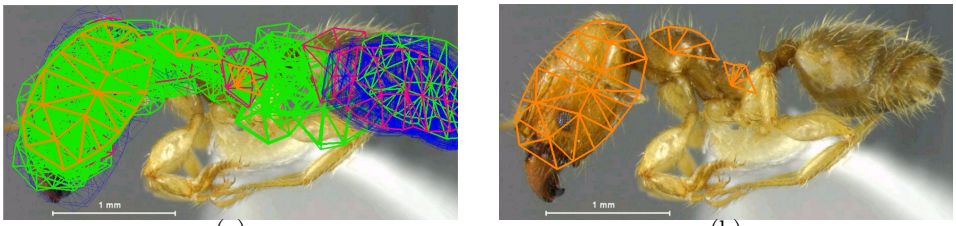


(a) (b)

Figure 6.10: **Multi-structure search**, *Pheidole carribaea* Three different interpretations are applied to the image (showing the extracted edge features) (a) The output of all searches is blended on top. The green shapes are possible candidates. (b) The interpretation as pheidole is emphasized as structure of highest probability shown in front of the output of the ant color sensor. (pheidole 75.4%, anochetus 72.0%, cerapachys 61.6%)

Fig. 6.11 is another example of a successful structural match. The full search has terminated after 70 seconds. Nevertheless, the finally correct interpretation has

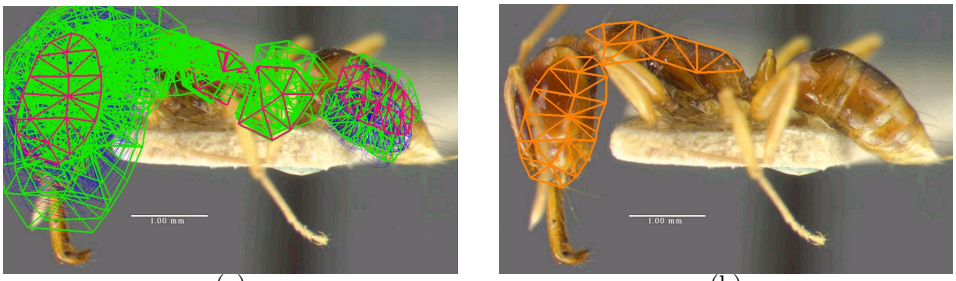
already been found after only three seconds. The reason for this huge difference is the depth of the *Cerapachys* structure consisting of five levels. Sub-shapes are spawned hierarchically descending in all levels. It is difficult to state which of the shapes is most important. Thus, it is not straightforward to speed up the structural search by only descending into the most promising interpretations. Moreover, only the full match of all shape constituents results in a quality measure that rules out all other interpretations. For that reason, it is not trivial to cut down on search branches in the structural matching.



(a) (b)

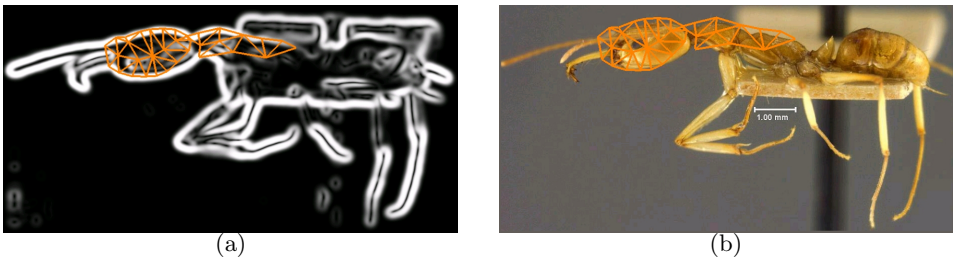
Figure 6.11: **Multi-structure search**, *Pheidole subarmata* (a) all shape candidates with most probable interpretation (orange). (b) The head is slightly displaced, the small peak in the lower thorax has been detected although being partly occluded by the leg. (probabilities of different interpretations: pheidole 82%, anochetus 76%, cerapachys 62%)

All structural searches illustrated on the previous images and below are done using the same complete structure graph. The following examples in Fig. 6.12,6.13, and 6.14 demonstrate the choice of a different interpretation if an image of a different type of ant is presented.



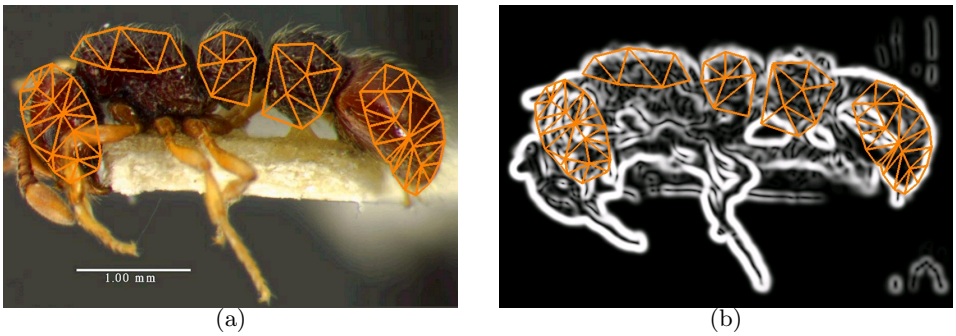
(a) (b)

Figure 6.12: **Multi-structure search**, *Anochetus cato* (a) all shape candidates with most probable interpretation (orange). (b) The head is slightly displaced, the small peak in the lower thorax has been detected although being partly occluded by the leg. (probabilities of different interpretations: pheidole 74%, anochetus 81%, cerapachys 65%), searching time: 50 seconds



(a) (b)

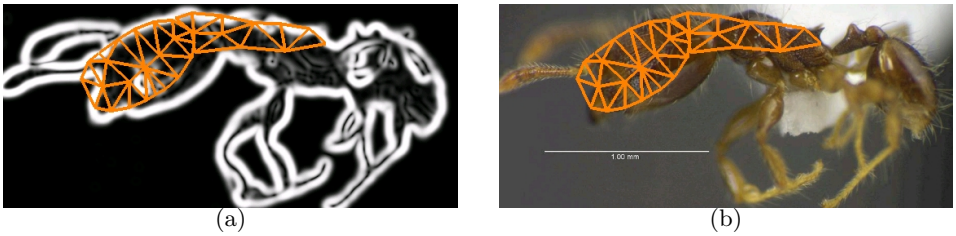
Figure 6.13: **Multi-structure search**, *Anochetus haytianus* (a) the ant is correctly classified with a probability of 79% (pheidole 50%, cerapachys 0%). (b) The background results in an insufficient edge signal after the ant sensor. The remaining weak signal is supported by the expectation from structural knowledge.



(a) (b)

Figure 6.14: **Multi-structure search**, *Cerapachys vitiensis* An important property of the relative structural rating is that it can compare structures of different lengths.

A crucial point when comparing different structures is to compensate for the effect of different numbers of shape parts on the rating score. As already stated earlier, when comparing different structural matches within one interpretation, always the most complete one is chosen. This is achieved by simply using the unnormalized score. When comparing different interpretations a relative score is used that normalizes by the maximum possible score for the subset of the structure that has been explored so far.



(a) (b)

Figure 6.15: **Ambiguous interpretations.**, *Pheidole amazonica* (a) mistaken interpretation of a pheidole and as Anochetus, (b) the characteristic round shape of the back is lost at a coarser scale of resolution, which has been used to increase the range of convergence.

The result in Fig. 6.15 shows an example of misinterpretation. The insufficient edge signal that could be obtained from the sensors has lead to an ambiguity between pheidole and anochetus interpretation that could not be resolved. This is a good example for a situation where the application of higher level knowledge fails if the low-level processing is not capable to extract a minimum of distinct features.

In addition to these pictorial results a test on 75 datasets of class pheidole has been performed. A manual evaluation determined that 12 of them have been classified incorrectly. This corresponds to a ratio of 84% correctly recognized images. Nevertheless, this result has to be interpreted with care. The operation has not been performed on the full database. The species used as input were chosen for good image quality and a clear appearance of the depicted species. So the ratio of 84% correct detections applies to images were all significant features are present. The remaining rate of misclassification is due to a bad response to the crucial feature detectors. Furthermore, similarities in the shape and structure of the ants may increase their probability to become mixed up. In that case it might have helped to add more distinct characteristics, such as the antennae of an ant, to the structural description.

Chapter 7

Summary

The previous chapter already provided discussions along with the results. This chapter ties the strings together to provide an overall evaluation and conclusions towards further investigations.

7.1 Discussion

The system described in this thesis has been developed along the task of classifying ants in an image database. Nevertheless, the problems and solutions found are not restricted to our specific application. The general idea of using structural knowledge to assist local searches for sub-shapes has proven to be a useful improvement over isolated searches for singular shapes. The resulting system is a combination of dynamic and statistical approaches.

One goal of this work was to explore a further application along the lines of research done on ASSM (see § 2.2.5). The grouping of sub-shapes to form a more complex object can be seen as knowledge that is provided about inner correlations within a shape model. Such information may be obtained by a statistical analysis of point distributions of training data. Statistical models are capable to automatically detect correlated subsets within multidimensional distributions. Nevertheless, since they are modeling a point distribution by a linear combination of principle components, they are not capable to reflect non-linear relationships between points. Especially variations that are caused by rotations around inner joints can not be represented compactly. Instead, they will result in multiple modes of variation. As a consequence, variations within these rotated subsets will not be represented adequately. This is the point were the structural model presented here is superior to linear statistical models. The automatic detection of non-linear correlation is replaced by a manual definition of sup-shapes. These may be subject to further statistical investigation separate from each other. Additionally, the transformations between the parts may be analyzed. The resulting model will more precisely

reflect the variation that is inherent to the nature of the object.

A statistical analysis has shown to improve an ad hoc setup of physical parameters for the dynamic model. Furthermore, a statistical analysis of transformations between shapes is employed to determine characteristic relationships. The detection rates show that the whole system is capable to make sensible decisions on a chosen subset of images. The results presented so far are indicating that the method basically works. In particular, it is shown that combined structural models may be used to classify ants based on their characteristic morphology. A more general evaluation using more structural templates of more different classes would help to further emphasize the general usability of the approach.

A problem that might become relevant in more complex systems is the discrimination between similar structures. To allow for further investigations the structural matching not only returns one optimal result but instead provides the optimal match for all applicable interpretations along with their ratings.

For the implementation, attempts have been taken to justify all crucial sets of parameters for sensors, shapes, and structure. An issue that has not been addressed so far is the setup of the weights for rating the structural constituents. For the current approach these weights are adjusted manually. The purpose of these weights is to put different emphasize on the parts of a structural model. Thus, these weights are a means to indicate *relevance* of different structural features. Parts that are more reliably found than others may get a higher rating. The weights for the structural links have a certain relationship to the adjustment of spring constants. Both sets of parameters are enforcing model constraints. By putting more emphasis on the connections, the quality of the connected shapes may become less important. Useful results are obtained by just leaving these weights set constant to one. Nevertheless, tuning them may increase robustness and reliability of the search. Since the other crucial parameters are chosen with some justification, it would be desirable to have a systematic approach to setup the weighting as well. A possible solution could be the incorporation of *relevance feedback*. Alternatively, it is possible to derive an appropriate weighting from the training data – just as it is done for the spatial relationships.

Another contribution within this thesis is the development of a feature extractor for ants. The resulting sensor is successfully using color information. The chosen sensor may not provide satisfactory results for all species of ants. To solve this problem it would be possible to follow two different approaches. The first one is to make the ant sensor more versatile applying more sophisticated statistical modeling. This could be achieved by extracting a Bayesian classifier from a set of hand-labeled images. The other option would be to derive a set of multiple sensors for various species. With the current framework it is possible to use several shapes having different sensors. This allows to specify distinct shape templates for the extraction of different appearance features. Other possible improvements would be the incorporation of textural features. Each of these extensions may easily be

integrated into the existing framework.

7.2 Future Work

This section points out, which directions might be promising for further improvement. Since there are many ideas on several issues, the following analysis is provided in form of a list.

Improving the shape matching could greatly speed up the search and increase reliability of the results. The current implementation of physical models provides a convenient way of performing shape search even with insufficient training data. The realization of shape search is independent from the control by the searcher. Hence, it would be possible to replace the dynamic model by a statistical one as soon as enough training data has been acquired. Furthermore, it is possible to improve the physical models themselves by incorporating multi-springs or an analysis of free vibration modes. Another issue that could further be explored along these lines are alternative ways for online refinement of shape descriptions.

The automatic construction of models could be done by providing a set of clean example images with binary masks overlaid. In subsequent steps it could be possible to derive basic geometric properties and a model of appearance of the subshapes. To ease the task of automatic model construction it would be possible to manually specify a set of significant feature extractors. Different issues to consider are: feature selection (type and scale), landmark placement, geometry generation, structuring, obtaining statistics.

Level-of-detail (LOD) shape description and matching is an interesting solution to increase convergence of the search and to improve quality of the final match. The question of detail levels is concerning almost all layers of the system, which would require a slight extension of the design. Firstly, the sensors have to work on multiple resolutions, adapting automatically as a finer scale is chosen. Secondly, the geometry of the shape can be significantly simplified for coarser scales. This also requires a concept for representing geometry at different levels of detail. Here, the discussion of implicit vs. explicit models comes into play again. Thirdly and most importantly, the search method should profit from the multi-scale approach to make earlier and more general decisions of which structures to prune and which paths to follow. Here lies the major benefit of an LOD approach.

Multiple structures in one image have not been tested so far. When looking for objects and generating hypothesis of possible structures, it is assumed that only one structure is contained in the image. Generally, it is possible to extend the

chosen approach to multiple structured objects within an image. The clustering mechanism described in §4.4.3 is capable to detect multiple deformable shapes. The detection of multiple structures is similar to the problem of finding several optimal paths along the shape winners in the structure graph. Multiple candidate paths should be extracted for the same structural interpretation. The implementation within the framework is fairly straightforward. The only problem is the selection of the structures. Here, it might useful to employ assumptions about spatial separation of structures or at least disjunct sets of shape candidates.

The property vectors introduced in § 4.4.1 are chosen to reflect the most fundamental characteristics of a shape. The inclusion of additional model parameters into this property description would allow for a more elaborate analysis of relationships between shapes. As a consequence, the searcher would have to control a larger search space. It is straightforward to increase the dimensionality of the property vector in the current framework. An issue to consider here is the definition of transformations for the additional dimensions. Possible features to add to the description would be basic parameters for attached sensors or a control of different modes of deformation from outside. The latter would enable an analysis of co-deformations among different shapes in the structure.

Time varying data is a very promising application for the system as it is now. The physics in the underlying dynamic model could be adapted to emulate the real world behavior of the objects shown in the images. This way, a natural method of extrapolating the movement of occluded objects could be obtained. So far, the computation of features is done on floating point numbers using higher order mathematical operations. This problem would have to be solved for real-time tracking by developing a faster (and probably simplified) feature extraction.

Extending the approach to 3D can be understood in two different ways. A first option is to employ three dimensional shape and structure models and apply them to perspectively transformed 2D images. This would allow to recognize objects from different perspectives. The other approach would be to look at volumetric data. This would require to consider resource and performance issues in 3D feature extraction.

The Exploration of further applications would help to underline the general usability of the model. Suitable domains would have to exhibit multiple similar objects within one image (e.g., quantification of microorganisms). For that purpose it is worthwhile to improve the genetic search strategies that have already been prepared by the selection of winners in the searcher. The use of crossing schemes would allow to distribute well formed shapes to other candidate locations.

The application to CBIR is a major direction for the model as it is now. The field of image retrieval may gain major improvement by a more generalized extraction of semantic features. A key issue to solve, is the construction of a structural model that provides a sensible mapping to all contents present in the database. In terms of the ant database this would involve to find a general structural description that applies to all ants (or at to least a basic subset of them). Since the extraction of detailed structure may fail, it could be useful to additionally include a very basic 'structure' that just extracts fundamental properties of the image. This would provide a reliable default behavior in cases where the structural model does not apply very well. In addition to the matching of a structure it is necessary to extract features from the arrangement of the shapes. A large variety of anatomic of morphological measures is imaginable here. A possible input to a further statistical analysis could be a vector of all property vectors of sub-shapes along with their spatial relationships (formulated as property transformations).

This list of further directions is even longer than the issues this thesis has started with. It is not necessary to explore all of them. So far, they shall just be noted because they implicitly reflect lessons learned during the work on this thesis.

7.3 Conclusions

The goal of this thesis was to establish a structural model that may become a superior method to detect complex shapes. In consideration of issues with existing approaches, a number of solutions has been developed to apply the idea to the classification of ants in image databases. For that purpose, a framework has been developed that, due to its hierarchical modular organization, allows for a convenient incorporation of future enhancements.

The major achievements in this thesis are:

- implementation of a physically-based model for matching deformable shapes,
- online refinement of physical parameters to improve correspondence to typical deformations,
- flexible design of various sensors for the extraction of 'geometric' features, such as edges and corners in intensity distributions across different scales
- extraction of saturation and multi-channel gradient to apply the model to color images,
- creation of a parametric statistical sensor to classify ant bodies more precisely,
- use of a stochastic searcher to avoid local minima in shape search involving an efficient clustering mechanism,
- introduction of expectation maps to reflect spatial relationships between shapes,

- statistical representation of structural knowledge applied to guide parallel searches for sub-shapes,
- a flexible rating function to reflect varying relevance within structures and making differently sized structures comparable,
- efficient search for optimal interpretations using dynamic programming,
- termination criterion based on exhaustive sampling of expectation maps,
- successful automatic classification of ant images by using the structural framework.

Despite the long list of solved issues, there still is a considerable amount of work to be done to put the system to general applicability. A first results of 84% recognition rate for a test subset of 75 images has to be seen as a *proof-of-concept*. To form a complete and robust system a number of issues still has to be solved. The bottom-line is that the development of a multi-layered system for the recognition is a lot of work if done in a basic language like C++. In turn for the amount of work, a lot of new insights have been derived opening several exciting directions for further research.

Bibliography

[1] Stephan Al-Zubi. Survey of deformable models. Technical Report TR-ISGBV-02-03, Dept. of Simulation and Graphics, Otto-von-Guericke University, 2002. 2

[2] Stephan Al-Zubi and Klaus D. Tönnies. Extending active shape models to incorporate a-priori knowledge about structural variability. In Luc Van Gool, editor, *Proceedings Pattern Recognition, 24th DAGM Symposium, volume 2449 of LNCS*, pages 338–344, Zurich, Switzerland, September 2002. Springer Verlag. 2.2.5

[3] David Baraff and Andrew Witkin. Large steps in cloth simulation. *Computer Graphics*, 32(Annual Conference Series):43–54, 1998. 3.2

[4] Steven Bergner, Regina Pohle, Stephan Al-Zubi, Klaus D. Tönnies, Annett Eitner, and Thomas R. Neu. Segmenting microorganisms in multi-modal volumetric datasets using a modified watershed transform. In Luc Van Gool, editor, *Proceedings Pattern Recognition, 24th DAGM Symposium, volume 2449 of LNCS*, pages 429–437, Zurich, Switzerland, September 2002. Springer Verlag. 2.1.1

[5] Chad Carson, Megan Thomas, Serge Belongie, Joseph M. Hellerstein, and Jitendra Malik. Blobworld: A system for region-based image indexing and retrieval. In *Third International Conference on Visual Information Systems*. Springer, 1999. 2.3

[6] T. Cootes and C. Taylor. Statistical models of appearance for medical image analysis and computer vision. *Proc. SPIE Medical Imaging*, 4322:236–248, 2001. 5

[7] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *Lecture Notes in Computer Science*, 1407:484–499, 1998. 2.2.2

[8] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, January 1995. 2.2.2

[9] T.F. Cootes and C.J. Taylor. Combining point distribution models with shape models based on finite-element analysis. *Image and Vision Computing*, 13(5):403–409, June 1995. 2.2.3

[10] Timothy F. Cootes and Christopher J. Taylor. Combining elastic and statistical models of appearance variation. In *Proc. of the European Conference on Computer Vision (1)*, pages 149–163, 2000. 2.2.3

[11] Rh.H. Davies, T.F. Cootes, C.J. Twining, and C.J. Taylor. An information theoretic approach to statistical shape modelling. In *Proc. of the British Machine Vision Conference (BMVC'01)*, volume 1, pages 3–12. BMVA Press, 2001. 2.2.3

[12] G. Dietrich, R. Hundt, G. Kopprasch, G. Kummer, K. Lobeck, I. Meincke, R. Stade, and H. Theuerkauf. *Wissensspeicher Biologie – 3. Auflage.* Volk und Wissen Volkseigener Verlag, Berlin, 1979. 1

[13] P. F. Felzenszwalb. Representation and detection of deformable shapes. In *Proc. of IEEE Conf. on Comp. Vision and Pattern Recognition*, pages 102–108. IEEE, June 2003. 2.2.4

[14] Margaret Fleck, David Forsyth, and Chris Bregler. Finding naked people. In *IEEE European Conference on Computer Vision*, volume 2, pages 592–602, 1996. 2.1.2

[15] M. Gleicher. Projective registration with difference decomposition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 331–337, 1997. 2.2.1

[16] G. Hamarneh, T. McInerney, and D. Terzopoulos. Deformable organisms for automatic medical image analysis. In *Proc. of MICCAI'01*, pages 66–75, Utrecht, The Netherlands, October 2001. 2.2.4, 4.3.2

[17] A. Hill and C. J. Taylor. Model-based image interpretation using genetic algorithms. *Image and Vision Computing*, 10(5):295–300, 1992. 2.2.4

[18] R.-L. Hsu, M. Abdel-Mottaleb, and A. K. Jain. Face detection in color images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(5):696–706, May 2002. 2.1.2

[19] Q. Iqbal and J. K. Aggarwal. Combining structure, color and texture for image retrieval: A performance evaluation. In *16 th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 438–443, August 2002. 4.2.1

[20] Bernd J"anhe. *Digitale Bildverarbeitung.* Springer-Verlag, Heidelberg, 2002. 2.1

[21] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. In *Proc. of the CVPR 99*, volume 1, pages 274–280, 1999. 2.1.2

[22] Sarang Joshi, Stephen Pizer, P. Thomas Fletcher, Paul Yushkevich, Andrew Thall, and J. S. Marron. Multi-scale 3-d deformable model segmentation and statistical shape analysis using medial descriptions. *IEEE Trans. on Medical Imaging*, 21(5):538–550, May 2002. 2.2.1, 2.2.3

[23] Sarang C. Joshi, Stephen M. Pizer, P. Thomas Fletcher, Andrew Thall, and Gregg Tracton. Multi-scale 3-d deformable model segmentation based on medial description. *Information Processing in Medical Imaging*, pages 64–77, 2001. 2.2.1

[24] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988. 2.2.1

[25] T. Lindeberg and B. M. ter Haar Romeny. Linear scale-space. In B. M. ter Haar Romeny, editor, *Geometry -Driven Diffusion in Computer Vision*, pages 1–41. Kluwer, Dordrecht, The Netherlands, 1994. 2.1

[26] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):77–116, 1998. 2.1

[27] Margaret W. Matlin. *Sensation and Perception — Second Edition.* Allyn and Bacon, Boston, MA, 1987. 4, 1

[28] T. McInerney and D. Terzopoulos. A finite element model for 3D shape reconstruction and nonrigid motion tracking. In *Proc. Int. Conf. on Computer Vision (ICCV'93)*, pages 518–523, Berlin, Germany, 1993. 2.2.1

[29] T. McInerney and D. Terzopoulos. Topologically adaptable snakes. In *Proc. Fifth Int. Conf. on Computer Vision (ICCV'95)*, pages 840–845, Cambridge, MA, June 1995. 2.2.1

[30] D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(6):580–591, 1993. 2.2.1

[31] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space. In *Proc. International Workshop on Image Databases and MultiMedia Search*, pages 35–42, 1996. 2.3

[32] G. Mori, S. Belongie, and J. Malik. Shape contexts enable efficient retrieval of similar shapes. In *Proc. of IEEE CVPR*, pages 723–730, 2001. 2.3

[33] A.P. Pentland and S. Sclaroff. Closed-form solutions for physically-based modeling and reconstruction. *IEEE trans. Patt. Anal. Mach. Intell.*, 13(7):715–729, July 1991. 2.2.1, 3.2

[34] S. M. Pizer, P. T. Fletcher, S. Joshi, A. Thall, J. Z. Chen, Y. Fridman, D. S. Fritsch, A. G. Gash, J. M. Glotzer, M/ R. Jiroutek, C. Lu, K. E. Muller, G. Tracton, P. Yushkevich, and E. L. Chaney. Deformable M-reps for 3d medical image segmentation. *International Journal of Computer Vision*, 55(2):85–106, November 2003. 2.2.1

[35] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990. 2.2.5

[36] S. Sclaroff and J. Isidoro. Active blobs. In *Proc. of the 6th IEEE Int. Conference on Computer Vision*, pages 1146–1153, 1998. 2.2.1

[37] K. Siddiqi and B. Kimia. Toward a shock grammar for recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 507–513, San Francisco, CA, June 1996. 2.2.5

[38] G. Stiny and J. Gips. Shape grammars and the generative specification of painting and sculpture. In O. R. Petrocelli, editor, *The Best Computer Papers of 1971*, pages 125–135. Auerbach, Philadelphia, 1972. copy available at `http://www.shapegrammar.org`. 2.2.5

[39] M Styner, G. Gerig, J. Lieberman, D. Jones, and D. Weinberger. Statistical shape analysis of neuroanatomical structures based on medial models. Technical report, Dept. of Computer Science, Image Analysis Group, Oktober 2001. submitted to Medical Image Analysis. 2.2.3

[40] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988. 2.2.1

[41] Andrew Thall, Stephen Pizer, and Tom Fletcher. Deformable solid modeling using sampled medial surfaces: A multiscale approach. Technical Report TR00-005, Dept. of CS, University of North Carolina at Chapel Hill, 01 2000. 2.2.1

[42] R. Veltkamp and M. Tanase. Content-based image retrieval systems: A survey. Technical Report UU-CS-2000-34, Utrecht University, 2000. 6

[43] V. Vezhnevets, V. Sazonov, and A. Andreeva. A survey on pixel-based skin color detection techniques. In *Proc. of Graphicon-2003*, Moscow, Russia, September 2003. available online at `http://graphics.cs.msu.su/en/publications/`. 1

[44] Andrew Witkin, David Baraff, and Michael Kass. Physically based modeling (SIGGraph 2001 course notes (25)). `http://www.pixar.com/companyinfo/research/pbm2001/`. 3.2

[45] A.L. Yuille, D.S. Cohen, and P.W. Hallinan. Feature extraction from faces using deformable templates. *IJCV*, 8(2):99–111, August 1992. 2.2.1

[46] S. Zhu and A. Yuille. FORMS: A flexible object recognition and modeling system. *Int. J. Comp. Vision*, 20(3):187–212, 1996. 2.2.5

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und nur unter der Verwendung der angegebenen Quellen angefertigt habe.

Magdeburg, den 10. Dezember 2003

Steven Bergner