A Framework for Segmentation and Contour Approximation in Computer-Vision Systems

Dissertation
zur Erlangung des Grades des
Doktors der Naturwissenschaften
der Universität Dortmund
am Fachbereich Informatik
von

Dipl.-Inform. Christian Leubner

Lehrstuhl VII – Graphische Systeme
Fachbereich Informatik
Universität Dortmund

Dortmund
2002

Tag der mündlichen Prüfung: 2. Dezember 2002

Dekan: Prof. Dr. Bernhard Steffen

Gutachter: Prof. Dr. Heinrich Müller, Prof. Dr. Martin Riedmiller

Contents

Chapter 1

Introduction

Computer vision has become an important and growing field of research in recent years. While related applications are manifold, especially human-machine interaction is an important part of nearly all technical devices and as such has received a lot of research attention. Generally, the interaction should be intuitive and comfortable in order to be accepted by the user and potential customers. Computer vision is one of the key techniques to realize user-friendly interfaces as for example gesture recognition because it enables a natural and wireless interaction by observing the user's gestures. The user does not need to learn the handling of specific input-devices such as mouse or keyboard and is free to move. He can simply use his arms or his hands to intuitively interact with the machine.

Computer vision systems normally consist of a whole pipeline of image processing operations in order to extract the required information from video images. This work is dedicated to one of the first processing steps, the so called *segmentation* of images. Usually, segmentation denotes the distinction between different parts of an image, for instance different objects in a scene. Gonzalez and Woods in [GON92] wrote:

> "Segmentation subdivides an image into its constituent parts or objects." (p. 413).

Depending on the specific goal of respective applications, the segmentation results are utilized to distinguish between relevant and irrelevant image content. Consequently, the amount of image data that has to be processed in further processing steps is reduced enormously. In the context of human-machine interaction, segmentation often denotes the recognition and separation of humans within natural environments. More generally, this task is often referred to as "figure/ground" separation [BAL82]. Since the beginning of image processing and computer vision, segmentation and especially figure/ground separation has been and still is an important part of nearly any image processing application. Based on the regions of interest that have been determined by the segmentation phase, further processing steps may follow, which for instance extract more detailed information.

As one of the first processing steps, image segmentation is crucial for the performance of the whole application because further processing steps have to rely on the results of the segmentation phase and consequently on their quality. Thus, Gonzalez and Woods recommend in [GON92]:

> "[...] considerable care should be taken to improve the probability of rugged segmentation." (p. 413).

In order to achieve a stable segmentation, most systems have a couple of restrictions. For instance, the background, in front of which the segmentation is performed, may not be allowed to consist of arbitrary colors. Furthermore, many systems need specific expert knowledge, e. g., about the appearance of objects, at least at the start of the system to work properly. Despite these aspects, computer vision systems are strongly depending on illumination conditions. Especially varying illumination may cause severe trouble for the segmentation phase. As a consequence, many approaches are only suitable for indoor applications.

Basically, there are two different approaches to figure/ground separation. The first approach utilizes some kind of a priori knowledge about objects and their appearance. The second needs information about the background and determines objects by subtracting the known background from current images. Background subtraction has the advantage that assumptions about the appearance of objects, e. g., about skin color or markers that have to be worn, do not have to be made. A disadvantage, however, is that background changes caused, e. g., by varying illumination conditions have to be distinguished from actual objects. Moreover, the background itself is required to be static for this approach. Nevertheless, background subtraction seems to be preferable because it allows for more flexible applications and does not depend on appearance.

In this work, a framework for typical image segmentation tasks in applied computer vision systems is presented. Basically, our approach is based on background subtraction and extracts contour pixels of objects of interest in front of known background. It extends previous techniques and tries to overcome typical problems as for instance changing illumination. Background subtraction approaches are based on the assumption that changes only occur where new objects have moved into the scene. However, often this assumption is problematic because several disturbing effects as for example changing illumination conditions may also lead to severe changes in the image. Moreover, sometimes the background may slightly change, for instance a chair may be moved or a door may be opened. In order to cope with these effects and to distinguish between them and actual objects, we employ a multi-level segmentation structure. It consists of different processing levels which are intended to overcome these typical problems. As a consequence of the background subtraction approach, our segmentation system is restricted to scenarios, in which video cameras are mounted at fixed positions. The background that can be seen by the cameras is required to be more or less static, i. e., moving objects are not allowed except of the objects of interest. As these circumstances occur very often in applied computer vision, we consider this restriction to be not too constrictive. However, our system is not applicable in the context of robot vision systems, in which robots augmented with cameras autonomously navigate through their environment. But even in a scenario with fixed cameras, the background is not totally static and may change within one or more parts of the image from time to time, for example a chair may be moved. By integrating fuzzy logic and pattern recognition methods, these permanent changes in the background are recognized by our system, if they endure for a predefined time. Moreover, formerly learned situations can be recalled, if it is appropriate. This is the reason, why we relax the restriction of a ”static” to a ”more or less static” background. Due to the multi-level segmentation approach, the system is able to cope with varying illumination conditions smoothly. Noise and other disturbing influences as for example aliasing effects of the cameras can be removed almost completely. These are key problems of many segmentation techniques. Internal feedback mechanisms between the different processing levels within the multi-level structure allow for a continuous learning and updating of background knowledge and enable the system to autonomously adapt to chan-

ging background content and varying illumination conditions. The multi-level segmentation system is explained in chapter 2.

Our segmentation algorithm aims at finding pixels constituting the contour of the objects of interest in the image. The obtained contour pixels usually do not yield a closed contour in the sense of a 4- or 8-neighborhood relationship of pixels, which would easily lead to a complete spatial separation of different objects. Normally, the contour contains a lot of gaps, so further processing steps are necessary to close these gaps and to obtain a closed contour. This is a typical problem for almost every segmentation algorithm. Within our segmentation framework, we present a new approach to polygonal contour approximation that is adjustable as far as the accuracy of the approximation is concerned and that works in realtime. The contour approximation allows a strict distinction between background regions and the objects of interest. Moreover, the shape is considered to be an important feature of objects, so the determined contour may be incorporated within further pattern recognition processing steps. Our new approach to polygonal contour approximation is introduced in chapter 3. Finally, chapter 4 is dedicated to examples of applications. In chapter 5, a summary and a conclusion is drawn. Fuzzy logic methods are utilized to cope with the typical fuzziness in image processing systems, which results from noise and other disturbing influences. Moreover, fuzzy rule bases are utilized for classification purposes. Thus, a brief course in fuzzy logic is given in appendix A.

1.1 Survey of the System

In order to compare the introduced multi-level segmentation system with the state of the art, we first of all present a brief survey of the system, so the relevant fields of research become obvious.

The usage of the presented segmentation system can be divided into two different phases:

- An initial *learning phase* and
- the *application phase*.

During the initial learning phase, the system analyzes the background and acquires information about it. During the application phase, the system has sufficient knowledge to perform the segmentation task.

The system utilizes a hierarchical multi-level processing approach to yield the segmentation results. One of the key ideas of the system is to have feedback mechanisms enabling each processing stage to control and adjust the underlying processing stages. Thus, the system has the following sequence of processing:

1. **Low-Level Processing Stage**
   The low-level processing stage performs a pixelwise classification of the image content. For this purpose, color and edge information of current video images is stored in fuzzy knowledge bases and compared with the initially learned background content. As a result, contour pixels are determined which ideally belong exclusively to new objects. The details are explained in section 2.1. However, sometimes pixels are classified as ”foreground”, although they actually belong to the background.

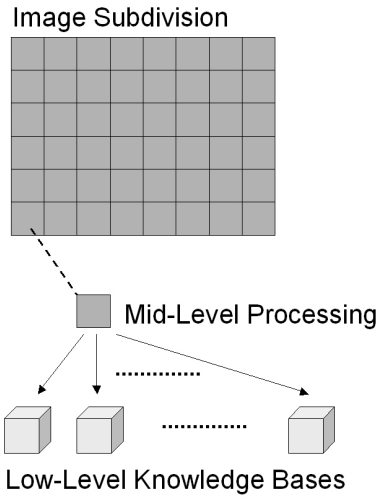


Figure 1.1: For each mid-level box, different low-level knowledge bases are administrated in order to cope with different background content.

2. **Mid-Level Processing Stage**
   The goal of the mid-level processing stage is the independent determination of background regions. Consequently, wrongly classified pixels in these regions can be masked out. For this purpose, the image area is subdivided in small rectangular boxes. In addition to the information which is available pixelwise for each box from the low-level segmentation, further similarity measures are employed that are able to detect changes in the box area compared to the initially learned background. As a result, each box is classified either as ”background” or ”foreground”. The details of the similarity measures and the employed fuzzy pattern recognition system are explained in section 2.2.

   Moreover, several low-level knowledge bases can be administrated for each box area in order to cope with different background content. The system is able to determine automatically which low-level knowledge base should be used for the respective image content. In the case of a permanent change in the background, new color and edge knowledge bases are initialized and added to the set of already existing low-level knowledge bases. The reoccurrence of a previously learned background can be recognized facilitating the access of respective knowledge bases. This scheme is illustrated in figure 1.1.

3. **High-Level Processing Stage**
   During the high-level processing stage, application-dependent knowledge is incorporated to determine the presence and position of a foreground object. For this purpose, the objects within the image are determined as connected components of ”foreground” boxes as classified by the mid-level processing stage. Properties of the expected foreground objects such as their size, their shape, or just the number of objects can be incorporated in order to determine the final objects of interest and to remove further dispensable or wrongly classified foreground pixels. The high-level processing stage is explained in section 2.3.

4. **Feedback Mechanisms**
   The results of the high-level processing stage are passed back to the mid-level stage,

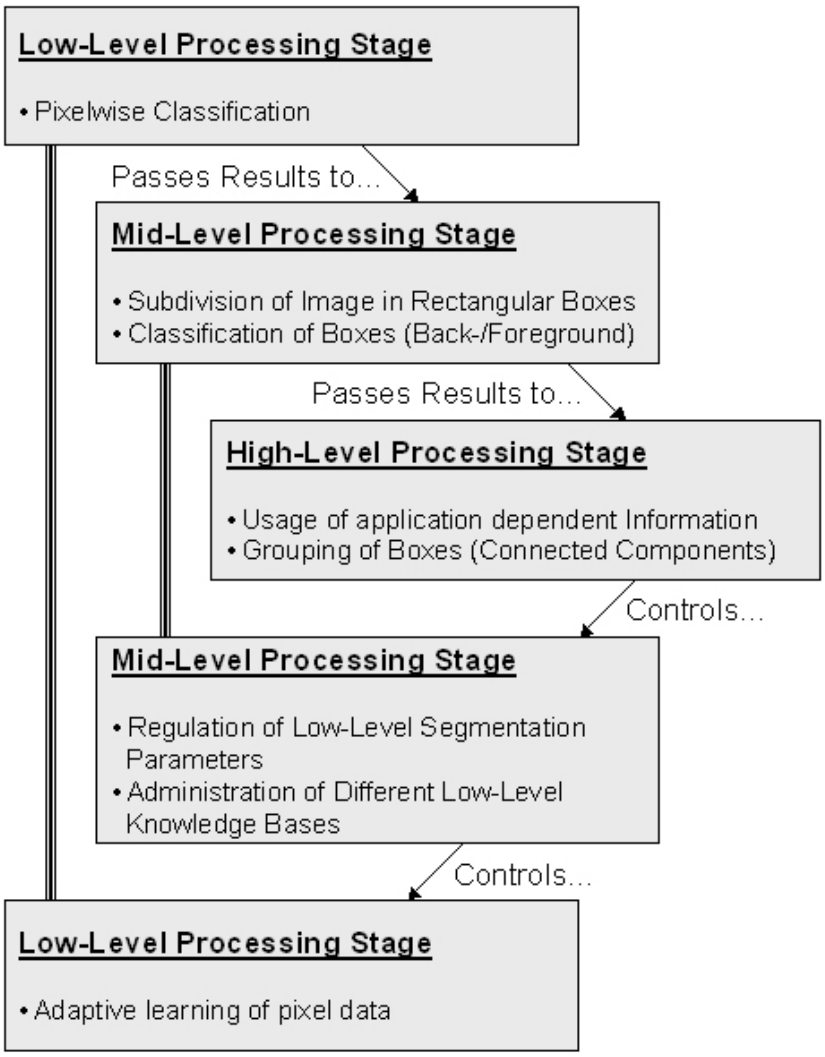


Figure 1.2: Principal structure of the segmentation system.

where parameters are adapted or changed depending on the results. The mid-level processing stage chooses suitable low-level knowledge bases or instantiates new knowledge bases, if appropriate. The low-level processing stage finally performs the pixelwise determination of contour pixels of the recognized objects. Moreover, information about the determined background regions is learned adaptively and integrated into the knowledge bases in order to maintain sufficient knowledge about the background. The details of this technique are explained in section 2.4.

The processing stages and their interfaces are illustrated in figure 1.2. In chapter 2, the processing stages are explained in more detail.

1.2 State of the Art

The definition and the meaning of ”segmentation” differs depending on the respective publication and its context. Mostly, segmentation denotes the distinction of relevant and irrelevant

image content, for example the separation of a person from the background in video images. A good survey of early used techniques is given in [HAR85] from 1985. More recent surveys can be found in [SKA94] or [SON98]. In this work, segmentation denotes the figure-ground separation as described in [BAL82].

The interaction of humans with machines and computers gains more and more interest. A significant part of interest is dedicated to the analysis of human gestures and motion. Generally, systems for sensing and processing of movement can be divided in *active sensing* and *passive sensing* [MOE01]. Active sensing employs devices, which are attached to the subject of interest or its surroundings. These devices generate or receive specific signals, which are used for the recognition process. Active sensing normally allows for simple processing and is especially suited for well controlled environments. Passive sensing incorporates mainly natural signal sources such as visual light or more generally other electromagnetic wavelengths. The usage of markers which are attached to the subject of interest is an exception that may simplify the segmentation and recognition process [MOE01]. Computer vision techniques can be considered as one of the key approaches to passive sensing. The attachment of markers on the one hand may be inconvenient or on the other hand impossible depending on the respective applications. Realizing a stable and reliable recognition of the objects of interest, which is moreover convenient and does not require any manually performed adjustments, is thus an important aspect for computer vision systems.

In a recent work, Moeslund and Granum [MOE01] have reviewed more than 130 publications on computer vision-based human motion capture systems. Our system is not restricted to the recognition of humans, which is subject of the survey, but can be utilized successfully for this kind of applications as shown in chaper 4, where several applications of our system are presented. Moreover, most of the problems which are related to the segmentation of humans are applicable for the segmentation of arbitrary objects of interest as well, at least, if a background subtraction approach is utilized. Within the survey, the following typical assumptions related to appearance are listed in ranked order according to their frequency. As far as the environment is concerned:

1. Constant lighting
2. Static background
3. Uniform background
4. Known camera parameters
5. Special hardware

As far as the subject of interest is concerned:

1. Known start pose
2. Known subject
3. Markers placed on the subject
4. Special colored clothes
5. Tight-fitting clothes

Considering these assumptions, our system is independent of most of these aspects. We explicitly take into consideration that the lighting may change during the application of our system. The assumption ”static background” applies to our approach. However, we aim at minimizing any negative impact of this restriction by allowing and automatically recognizing permanent changes. Our system does not have to know any camera parameters, the image data is taken as it is. Features of the background are extracted and learned automatically. We do not require any ”special hardware” as for instance IR-cameras. Assumptions concerning the objects can be specified optionally, for example the number of expected objects or assumptions about their size. The assumption ”known start pose” is not applicable for our system because we do not perform any pose estimation, for which this assumption could be necessary. It is partly the same for ”markers placed on the subject”. On the one hand, the system does not need any markers to determine the user in front of the background, on the other hand, our system does not interpret the user itself in the sense of determining the position of for example the hand, the head or the shoulders. Principally, there are no assumptions as far as the clothes of the user are concerned. However, depending on the subsequent application, for example the recognition of pointing arm gestures as presented in section 4.2, it may be necessary that the user wears ”tight-fitting clothes”. For the determination of the position of the user in front of the background – the result of our system – it is not required.

In their survey [MOE01], Moeslund and Granum identify mainly two approaches to figure-ground segmentation in computer vision systems: the use of *temporal* or the use of *spatial* data. They introduce two subclasses for the temporal data case, namely *subtraction* and *flow*. Subtraction generally means the performance of a kind of pixelwise subtraction of images, either of consecutive images or of a static background image from current video images. If the scene is static – a property that is mostly assumed for this approach – a background image can be recorded and used as a reference image for the subtraction [NAK98]. The continuous updating of the background image data is also a common approach and is mostly realized using statistical approaches, e. g., [HAR98, AZA96]. The second subclass, namely *flow*, aims at identifying coherent motion of points or features between image frames. Because of the obvious membership of our approach to the first subclass *subtraction*, we do not go into details of the second subclass.

The distinction between different levels of image processing are very common in computer vision systems. Ballard and Brown in [BAL82] characterize high-level processes as *cognitive processes, geometric models, goals* and *plans*. Moreover, they wrote

> ”Goals and knowledge are high-level capabilities that can guide visual activities, and a visual system should be able to take advantage of them.” (p. 2)

As low-level capabilities they name

> ”[...] for example, our ability to extract *intrinsic images* of ”lightness”, ”color” and ”range”.” (p. 3)

However, Ballard and Brown consider segmentation, the figure/ground discrimination, as a mere low-level processing capability, while our system also incorporates high-level knowledge in order to solve the segmentation task. Background subtraction is also treated as a filtering method in the context of ”early processing”. Marr in [MAR82] developed a model for the decomposition of visual processing into subsequent feed-forward steps, including low, intermediate and high-level stages. Moreover, he presented a rough correspondence between these

stages and areas in the human cortex. However, a sharp distinction between low, intermediate and high-level functionalities is difficult and even the assumption of a hierarchical organization of the human perception of vision is questionable because partial outputs from low-level processes initiate high-level processes, and, vice versa, the outputs of high-level processes feed back to influence the low-level processing [MCC86, LEE98].

According to this notion, systems for *image retrieval* have been designed, which employ multi-level structures. For example, a multi-level segmentation approach is used in [DUY01] and [XU00], where the authors try to narrow the gap between low-level features such as color or shape information and the semantic level, which models the way how people describe objects. A multi-level approach is applied in order to find and merge homogeneous regions as far as color properties are concerned. However, this method is intended to be employed in image retrieval systems and is not at all comparable to our approach because segmentation itself aims at finding and separating objects in arbitrary scenes and images in this context.

The gap between low-level, maybe even pixelwise obtained information and high-level knowledge and concepts has to be narrowed by nearly all computer vision-based applications. In [KIL94], which presents an architecture for an adaptive video-based traffic sensor, the low-level processing is followed by the determination of primitives such as lines or regions. Afterwards, the high-level processing stage performs a scene interpretation in order to detect and track cars on a highway. In our context, the work of Kilger [KIL94] is especially interesting because within his image processing concept an adaptive regulation of parameters is performed, for example parameters of the segmentation stage like the threshold are regulated adaptively. The segmentation is performed as background subtraction using a reference image that is continuously updated in the recognized background regions.

Adaptive regulation of parameters and adaptiveness to changing image content in general is an important aspect for computer vision systems. Especially in the 1950's and 1960's, a lot of research has been done in the field of adaptive regulation. In his survey, Weber [WEB71] characterizes three typical properties of an adaptive system:

1. **Identification**
   Continuous measurement of the current state of the system.

2. **Decision**
   Comparison of the real state of the system as measured by the identification and the desired state, and a decision and choice of appropriate actions in order to reach the desired state.

3. **Modification**
   Modification of the regulator based on the decision that has been made in the decision phase.

Adaptiveness plays an important role in computer vision systems, mostly justified by changing illumination conditions. Significant improvements with adaptive approaches have for instance been realized in [TIA98] where outdoor video images are processed for plant detection. Compared to a static segmentation approach, the number of correctly classified object pixels improved between 26.9 and 54.3 % under different weather conditions. The *PFinder* system [AZA96, WRE95] adaptively updates background statistics in order to keep track of changes. Fuzzy logic has found particular interest in recent years as far as image processing

[BEZ99, TIZ98], adaptive control [KAN94], or ”intelligent” systems in general [YAG92] are concerned. Fuzzy logic enables the specification of rule bases using linguistic terms and is thus especially suited for transforming human decision making processes into algorithms. A brief introduction in fuzzy logic is presented in appendix A.

Our segmentation system requires a more or less static background. For computer vision systems, we consider this restriction as unproblematic because video cameras are often mounted at specific fixed positions. However, because of this reason our system is not applicable in the field of robot vision, where video cameras are integrated in robots, which move through their environment.

A lot of research in recent years has been focused on segmentation in the context of the ISO MPEG-4 standardization where moving objects can be coded separately from the background within the video stream [MPEG4]. A survey on recent techniques as far as coding-oriented segmentation is concerned can be found in [NER98]. Because of the specific field of application, the extraction of the moving object has not to be as exact as it may be required within computer vision applications. Moreover, these techniques often incorporate motion estimation and consequently depend on the quality of the results of this estimation and their underlying models. Thus, segmentation algorithms in the context of the MPEG-4 standardization are not comparable to our approach.

Another large field of research is the field of *image understanding*, i. e., the recognition of objects within images, which is closely related to the field of *image indexing* and *image retrieval* from large databases. In these kinds of applications, segmentation mostly aims at detecting boundaries between different objects and second at recognizing objects, for example for indexing purposes. As far as image retrieval is concerned, comprehensive surveys of the state of the art can for example be found in [HUA97, RUI99]. Proposed segmentation algorithms in this context for instance use color segmentation including clustering based on histograms [PUZ99], minimum volume ellipsoids, fuzzy c-means or Gibbs random field (GRF) modeling in a Bayesian framework, to name just a few. A detailed review on gray and color image segmentation methods can be found in [PAL93]. In order to achieve meaningful results, color- and edge-based segmentation results are combined in this context as well. In [SAB97], a method for color image segmentation and edge linking is proposed. The basic idea is to identify contiguous regions by color segmentation and to split and merge regions. Then, a consistent and reasonable result is obtained referring to the edge map. This method is not intended to segment moving objects and the segmentation time for one image is specified ”a few minutes”. Because of the often insufficient reliability of just one approach, combinations of different segmentation techniques are widely spread in literature. For instance, the combination of color and shape information has been employed for the mere retrieval of image data from large-volume image databases [JAI96]. In [DUB93], good results were achieved by combining edge detection and a split-and-merge paradigm with motion segmentation in order to find driving cars. The combination of color and texture features has also been used to improve the segmentation and boundary detection in natural images [KUB98].

Within the mid-level processing stage of our system, the similarity between initially learned background and current video images for box-sized regions is determined. This task seems to be closely related to typical image retrieval functions because the similarity between images has to be determined for indexing and retrieval purposes as well. However, the specific aim is different. While image retrieval applications try to recognize objects, even if they are partly occluded or twisted, we aim at recognizing even small changes, which may indicate the en-

trance of a foreground object into the box region. Because of this reason, publications from this field of research cannot be applied in our context. An extensive review on measures of similarity for different fields of application is presented in [COX95]. A survey on distance-based functions such as *Hausdorff*-based or the *root of the mean square error* distance, which are utilized for image comparison purposes, is presented in [DIG99]. A general introduction to pattern recognition can be found in [DUD00].

Because of the variety of known segmentation methods and in order to compare the introduced system to known techniques, a concentration on methods which have similar requirements and methods is reasonable:

- static background
- automatic learning of background information
- segmentation of moving object in front of known background
- adaptive to changes in background
- suitable for interaction purposes
- combination of color and edge information
- multi-level structure.

In [BIC94], the segmentation task is identical. A statistical approach on object and background probability is utilized to perform the segmentation. In contrast to our approach, the color distribution is calculated for the whole background [BIC94] or at least for squares of the image [DUB93], whereas our technique collects color information for each pixel separately. A closely related work to [BIC94] can be found in [AZA96] and [WRE95]. The described *PFinder* and *SPFinder* systems model the scene as a set of distinct classes. The person is represented as a set of connected blobs, having a spatial and color Gaussian distribution as well as a support map that indicates for each pixel to which class it belongs. Adaptiveness is realized by recursively updating the background statistics. In contrast to our approach, this technique collects information about the color distribution of the foreground objects as well. This approach may be problematic, if the appearance of objects changes often. Moreover, objects consisting of several different colors may be difficult to model. In *PFinder* and *SPFinder*, the classification of pixels is done by measuring distances in the color space accomplished by spatial priors, as for example the spatial neighborhood of pixels, and connectivity constraints of the blobs. Because of the statistical approach, these systems may not be able to handle changes that do not fit into the statistical assumptions, even if they have occurred before and thus could be known. As a consequence, we should utilize a more flexible technique that is able to store detailed color information and to administrate several different background situations.

In some computer vision applications, the segmentation phase is intended to find skin colored regions, which can be interpreted for example as face or hand regions [SAX96, STA95, TER98]. However, in the context of pattern recognition, which often follows the segmentation phase, the segmentation may already be part of the feature extraction, for example by working out areas of human skin color. Thus, segmentation, although performed in almost every image processing system, depends heavily on the respective application and its specific realization.

For this kind of systems, our segmentation system may be utilized for localization purposes. Afterwards, the search for human skin color may be performed within the prelocalized areas nonetheless, but this is not part of our framework.

1.3 Contribution

In section 1.2, we have shown that substantial effort has been spent to cope with the figure/ground separation problem since the beginning of computer vision. However, the concrete techniques and methods have always been adapted to state of the art computational power and capabilities. Kilger [KIL94] first of all determines the number of processing instructions, which can be spent for each pixel in order to realize the recognition of cars in video images at a sufficient frame rate. In 1994, he calculated 11 instructions per pixels for a state of the art Intel i486DX CPU (25 MHz) and deduced that segmentation under these circumstances is a difficult task, which it certainly is.

Considering a typical PC equipment in 2002 with an Intel Pentium IV easily topping 2.5 GHz and main memory of approximately 1 GB and more, the technical possibilities of standard hardware have enormously improved and as far as it can be foreseen today, they are going to continuously improve in future. The presented system tries to exploit the capabilities of current hardware and offers the user a technique that provides reliable results with as less restrictions as possible in a convenient manner.

The aim to obtain reliable results is achieved by extending the well-known background subtraction method in different ways. Instead of keeping just one reference background image, our system introduces a new methodology to store and recall voluminous histogram and edge information for each pixel separately. Fuzzy logic techniques are utilized in order to administrate and apply the collected knowledge. A second, so called mid-level processing stage is introduced that subdivides the image area in rectangular boxes. A pattern recognition system is employed to detect changes in the box areas partly independent of the first processing stage. The presented similarity measures are designed to be insensitive against changing illumination conditions. One of the basic concepts of the system is to collect information about slight changes in the same knowledge base and about severe changes in separate knowledge bases. Having several knowledge bases for each box area, an appropriate one for current situations is chosen by the pattern recognition system on the basis of the similarity measures.

The multi-level structure enables the incorporation of application-dependent knowledge concerning the aim of segmentation. The usage and installation of the system itself is simple and convenient. Due to the self-learning approach, the cameras just have to be mounted at the desired positions, the system then automatically acquires necessary background information and is able to perform the figure/ground discrimination already after a few seconds. By incorporating adaptive control mechanisms, the system is able to automatically adjust parameters and thus to improve continuously. The unpleasant restriction of having a static background is considerably weakened by allowing and automatically recognizing permanent changes in the background. Former situations are stored and can be recalled, if this is appropriate. Further restrictions such as a uniform background or users wearing specific clothes are not required.

Besides this new interpretation of a multi-level segmentation system, a new approach to polygonal contour approximation is introduced that works in realtime and is adjustable as far as the accuracy is concerned. Moreover, we will show that our approach works geometrically more exact than other state of the art algorithms.

1.4 Scenarios of Application

The scenarios, for which we consider the segmentation system to be suitable, can be characterized by the following assumptions. As a major requirement, the background is expected to be more or less static. As we have already explained in section 1.2, this restriction is considerably weakened by allowing and automatically recognizing permanent changes in the background. However, the cameras have to be mounted at fixed positions. Consequently, zooming, moving, or rotating of the cameras is not allowed during the application phase. Otherwise, a new initial learning phase has to be started or the system must be given a short while until the permanent background change is recognized automatically.

The segmentation itself, i. e., the determination of contour pixels of foreground objects, is realized as a kind of background subtraction between current video images and learned background content. In order to work properly, the foreground objects have to differ from the background. However, incorporating several color channels, it is rather unlikely that large parts of a foreground object are nearly identical to the background. Nevertheless, in front of a uniformly colored background a uniformly colored object may not be recognized sufficiently well as far as the mid-level similarity measures are concerned. As a second aspect, the segmentation also incorporates edge information, enabling the system to partly eliminate failures of the color segmentation or vice versa. Nevertheless, the edge segmentation may partly fail, if the contrast of the video images is low, for example because of a dimmed illumination. Another reason for insufficient results of the edge segmentation may be a smooth gradient between foreground object and background, causing a gradient-based edge operator not to detect any edge at all. This may be the case, when objects move very fast leading to a blurred appearance in the image. Both color and edge segmentation are intended to supplement each other in order to achieve reliable and stable results. Nevertheless, the above listed aspects have to be taken into consideration because the occurrence of one or more of these aspects may lead to failures in the segmentation system.

During the high-level processing stage, foreground objects are determined as connected components of ”foreground” boxes. In order to work properly, we expect objects – as far as there is more than one object – to appear disjunctively, i. e., occlusions are not recognized by the system. Otherwise, the system may not be able to distinguish between several objects and identify them as one. For the high-level processing stage, the distance criterion between objects which has to be met to guarantee that they are not recognized as one object is provided by the neighborhood relationship of boxes and consequently the size of the boxes. As far as the contour approximation algorithm presented in chapter 3 is concerned, we mathematically analyze the connectivity/disconnectivity criteria in section 3.7.

Another important aspect for our system is the expectation of certain movements of the foreground objects. A still object that does not move for a predefined time is supposed to be a permanent change of the background and is learned as ”background”. Consequently, it will not be segmented in subsequent images until it moves again. However, the predefined time may be chosen with respect to the scenario, so the behavior of objects can be anticipated.

Chapter 2

Multi-Level Segmentation System

In this chapter, our multi-level approach to segmentation is introduced. The system consists of three processing levels: the low-level, the mid-level and the high-level processing stage, which are discussed in detail.

The basic approach to segmentation of our system is to perform an extended kind of subtraction between a known background and current video images. A brief survey of the system was given in section 1.1. The usage of several processing stages is principally optional. Thus, the system may consist of either the mere low-level, the low-level and the mid-level or all the processing stages. The basic segmentation task, the determination of foreground pixels by analyzing differences between the known background and current video images, is performed by the low-level processing stage. Thus, the low-level stage, if applied separately, can be employed to perform the segmentation task as well [LEU01]. The low-level processing stage is principally comparable to the well-known background subtraction approach that is based on the subtraction of a known reference image from current video images. However, our low-level approach is a combination of both edge and color based segmentation, leading to significantly more stable results than just one single segmentation approach would provide. Moreover, the low-level segmentation utilizes color and edge knowledge bases that are able to store extensive background knowledge, increasing the limited possibilities of a reference image enormously.

Additionally, the mid-level processing stage can be appended to reconsider the results of the low-level stage by applying pattern recognition mechanisms to rectangular box-sized image regions. While the low-level processing stage works pixelwise, the mid-level stage analyzes the box-sized areas and performs an autonomous comparison between known background and current image content. Thus, the mid-level processing stage identifies foreground object regions and misclassified pixels in recognized background regions can be masked out. Moreover, the pattern recognition system is utilized to administrate different background content. If a permanent change within the background is recognized by the system, the knowledge bases of the former background content are stored and new knowledge bases are instantiated for the current content. Due to the mid-level pattern recognition mechanisms, formerly learned background content is recognized and respective knowledge bases can be recalled.

As a further improvement, the high-level processing stage can be applied after the mid-level stage in order to reconsider the results of the mid-level stage based on application-dependent knowledge. Finally, feedback mechanisms are an important aspect of the system, which allow for a continuous adaptation of the parameters and an improvement of the results. The low-level processing stage is explained in section 2.1, the mid-level processing stage in section 2.2, and

the high-level processing stage in section 2.3. The feedback mechanisms are discussed in section 2.4. In section 2.5, an evaluation of the results is performed. The suitability of our system for multi-threaded execution is discussed in section 2.6 and the results of the segmentation system are summarized in section 2.7.

2.1 Low-Level Processing Stage

The low-level processing stage performs a pixelwise classification of the image. For this purpose, both color and edge information are utilized. In order to cope with slight changes in the illumination, we developed a *fuzzy color knowledge base* that is able to store detailed histogram information of background colors for each pixel separately. An advantage of this approach is the possibility to continuously update background color information for the recognized background regions during the application phase of the system without losing previously acquired background knowledge. This method realizes a steady adaptiveness to slight changes, which may be caused for example by changing illumination conditions.

The *fuzzy edge knowledge base* stores average edge information of the background for each color channel and each pixel separately. In order to yield a final segmentation result, the intermediate results which are provided pixelwise by the knowledge bases are combined. Fuzzy logic methods are utilized in order to cope with the typical fuzziness in digital image data. The segmentation itself is realized as a kind of background subtraction, both for color and edge information. Due to the detailed information about the background, which is provided by the knowledge bases, the algorithm is able to determine changes within the image that are interpreted as a moving object. The combination of color and edge segmentation yields more reliable results than only one approach could provide because two different kinds of information are utilized to obtain the result. As a consequence of the usage of edge segmentation, the algorithm determines contour information of the foreground objects instead of contiguous areas. However, we do not consider this aspect to be disadvantageous because contiguous areas can be easily determined based on a closed contour, for instance by polygon filling [FOL94].

As already mentioned, several knowledge bases may be utilized for the same image area in order to cope with different background content that may occur during a long period of usage. However, slight changes within the color values that may result from varying illumination can be stored in one color knowledge base. In the case of a severe change, new color and edge knowledge bases are additionally instantiated by the mid-level processing stage (see chapter 2.2).

2.1.1 Color-Based Segmentation

We propose a *fuzzy color knowledge base* that is able to store an extensive amount of color histogram information for each pixel separately.

Definition 2.1.1 (Fuzzy color knowledge base)

A color space *is a set Q of N_K-tuples with fixed N_K. A discrete image I is defined by a function:*

$$q : (x, y) \longmapsto Q \tag{2.1}$$

with

$$(x, y) \in \mathbb{N}_0 \times \mathbb{N}_0, \quad 0 \leq x < w, \quad 0 \leq y < h$$

where

$$w, h \in \mathbb{N} \quad : \quad \textit{width and height of the image counted in pixels}$$

A color value $q(\vec{p}) = (q_1(\vec{p}), ..., q_{N_K}(\vec{p})) \in Q$ of a pixel $\vec{p} = (x, y)$ consists of N_K color channels, which may have N_W different discrete values.

A fuzzy color knowledge base *for I and Q is defined by N_K arrays $C_{\vec{p},i}$ containing N_W entries of values $\mu \in [0, 1]$ (with $i = 1, ..., N_K$). $C_{\vec{p},i}(j)$ refers to the j-th entry of color channel i for pixel $\vec{p}$ ($j = 0, ..., N_W - 1$) and is considered to be a fuzzy membership value concerning the semantic property "is background color".*

As an example for Q, the *RGB color space* Q_{RGB} is given by a set of 3-tuples:

$$Q_{RGB} = \{(r, g, b) \mid r, g, b \in \{0, ..., 255\}\ \}.$$

A color value $q(\vec{p}) = (q_1(\vec{p}), ..., q_{N_K}(\vec{p})) \in Q$ is learned by the rule

$$C_{\vec{p},i}(q_i(\vec{p})) = \min(C_{\vec{p},i}(q_i(\vec{p})) + \sigma,\ 1) \tag{2.2}$$

with

$\sigma \in [0, 1]$: controlling parameter
$\vec{p}$: pixel position $\vec{p} = (x, y)$ with $0 \leq x < w$ and $0 \leq y < h$
$N_K \in \mathbb{N}$: number of color channels
i : color channel with $i \in \{1, ..., N_K\}$.

The parameter σ affects the impact of a single occurrence of a color on the knowledge base.

During the learning phase, a sufficient number of iterations of learning has to be performed in order to train the color knowledge base and to acquire sufficient information about the background. Figure 2.1 shows an example of learning for the RGB color space at a specific pixel position. For the currently occurring RGB color values, denoted as r, g and b, the corresponding entries are increased, depicted by the black boxes. The membership degrees are saturated at the maximum membership degree of 1 as it is the case in the figure for the B-channel.

A previously trained color knowledge base can be employed to perform a pixelwise fuzzy classification of a current video image I. For this purpose, the membership values of the semantic property "is background color", which are provided by the fuzzy color knowledge base, are used to calculate the membership values of the semantic property "is foreground color" by employing a fuzzy "not" operator. The membership $\mu_c(\vec{p})$ of a color $q(\vec{p}) = (q_1(\vec{p}), ..., q_{N_K}(\vec{p})) \in Q$ for a specific pixel $\vec{p}$ to the property "is foreground color" is obtained by looking up the corresponding array entries:

$$\mu_c(\vec{p}) = 1 - \min_{i=1,...,N_K} C_{\vec{p},i}(q_i(\vec{p})) \tag{2.3}$$

This method is illustrated in figure 2.2, again for the RGB color space. The currently occurring color values, denoted by r, g and b, are looked up in the knowledge base for the respective pixel position. The entries for the different color channels are combined using a

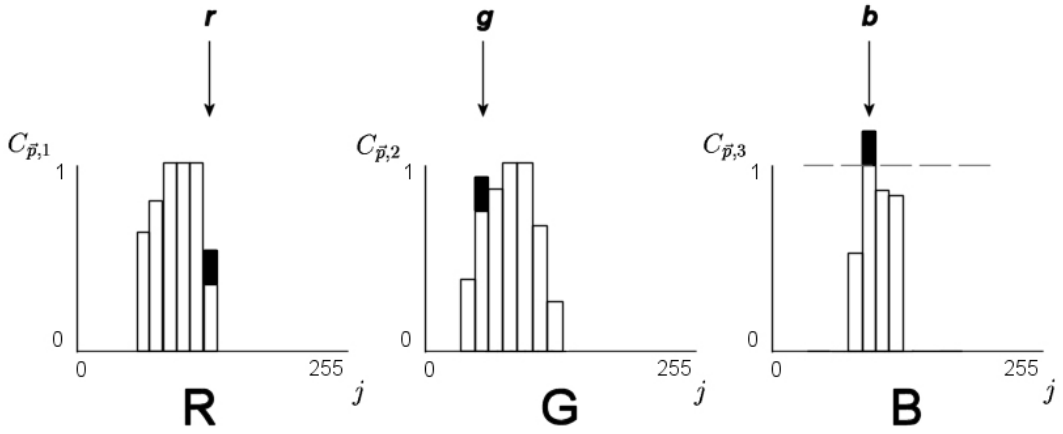


Figure 2.1: Schematic example of learning for RGB color space. Black boxes denote amount of increment, the membership degrees are saturated at the maximum membership degree 1 (B-channel).

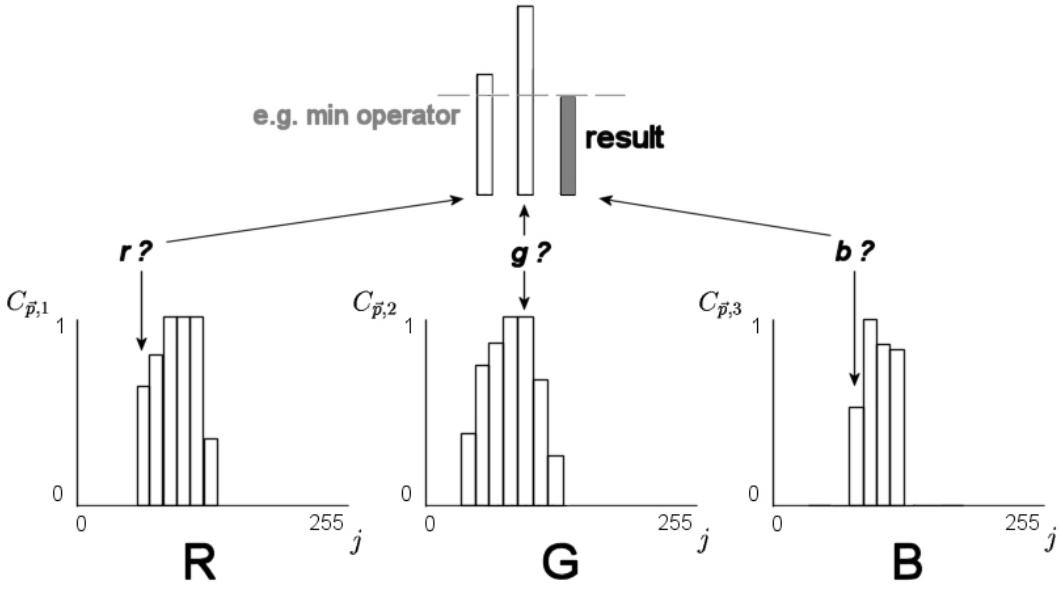


Figure 2.2: Classification of current color values with fuzzy color knowledge base.

(a) Original video image (b) Color-segmented visualization

Figure 2.3: Example of visualization. The original image (a) is color segmented and the result is visualized as gray-level image in (b).

fuzzy *and*-operator as for example the $\min$-function. The final result for the respective pixel is the membership degree denoted by the gray box in figure 2.2.

Applying $\mu_c(\vec{p})$ to every pixel $\vec{p} \in I$ yields a fuzzy segmentation of I. In order to remove noise in the fuzzy results, we propose the usage of an *averaging threshold* filter that averages the membership values of a pixel $\vec{p}$ and of the pixels $\vec{p}_i$ of the 8-neighborhood of $\vec{p}$. The calculated average membership value is compared to a threshold and the membership value $\mu_{color}(\vec{p})$ is determined by

$$\mu_{color}(\vec{p}) = \begin{cases} 0 & : \quad \frac{1}{9}\mu_c(\vec{p}) + \frac{1}{9}\sum_{i=1}^{8}\mu_c(\vec{p}_i) \;\leq\; \upsilon_c \\ \mu_c & : \quad \frac{1}{9}\mu_c(\vec{p}) + \frac{1}{9}\sum_{i=1}^{8}\mu_c(\vec{p}_i) \;>\; \upsilon_c \end{cases} \tag{2.4}$$

where $\upsilon_c \in [0,1]$ is used as a threshold parameter.

Alternatively, for the removal of noise and small, most probably wrong classified pixel areas a *Median* filter or morphological operators, e. g., an erosion followed by a dilation, may be applied. For illustration purposes, the result of the fuzzy classification of an image I can be transformed into a gray-level image. For example, an 8-bit gray-level image can be obtained by multiplying the fuzzy membership values $\mu_{color}(\vec{p})$ by 255 (see figure 2.3). In order to obtain reliable segmentation results, the fuzzy color classification results are combined with the edge segmentation explained later.

It has to be taken into account that color values that have been learned once may lose their correctness over time. Noise and other influences may accumulate small membership values in the array entries, which may grow to large and interfering membership values over a long time of application. Thus, after a certain number of iterations of learning the following formula is applied:

$$\forall \vec{p} = (x,y) \text{ with } 0 \leq x < w,\ 0 \leq y < h;\ \forall i = 1,...,N_K;\ \forall j = 0,...,N_W - 1:$$

$$C_{\vec{p},i}(j) = \max(0,\ C_{\vec{p},i}(j) - \rho) \tag{2.5}$$

where $\rho \in [0,1]$. Similar to the parameter σ in formula (2.2), the value of ρ regulates the speed of aging and dropping of the acquired knowledge. The aging of color knowledge is illustrated

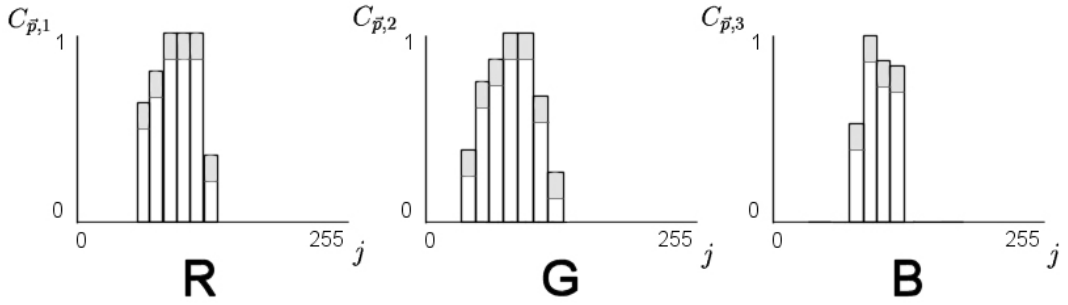


Figure 2.4: Aging of knowledge. Each entry is decreased by the given amount ρ (depicted as gray boxes).

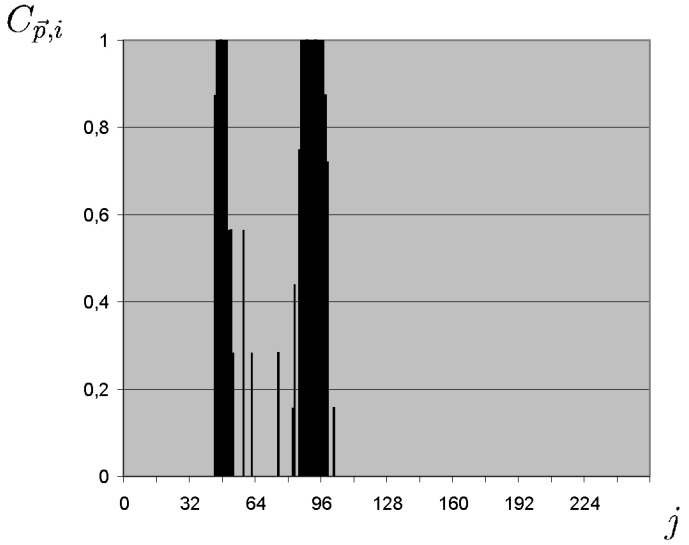


Figure 2.5: Example of color knowledge base for one color channel with two significant intervals of high membership degrees.

in figure 2.4. From each entry in the knowledge bases, a defined amount is subtracted, depicted by the gray boxes.

We consider our approach to be more flexible than statistical approaches because completely arbitrary situations can be learned. For instance, a background pixel that is affected by varying illumination and mainly changes between two colors can be handled sufficiently well. The situation is illustrated as an example for one color channel in figure 2.5. The well-known background subtraction approach which determines the average and the standard deviation of color values would have problems to cope with such a situation because the average color value is not within the intervals of background colors in this example, but between the two peaks. So the average color will be classified as background by the statistical approach, although it does not belong to the background. Our approach to simply store occuring background colors is able to cope with such a situation by providing high membership degrees only for known colors.

As already mentioned in chapter 1.3, we try to exploit the possibilities of state of the art hardware resources. For example, a fuzzy color knowledge base for an image size of 192×144 pixels, $N_K = 3$ color channels and $N_W = 256$ different color values requires approximately

(a) Original video image (b) Edge-filtered image

Figure 2.6: Example of application of *Sobel* edge detection operator.

20 MB of memory. Keeping in mind that we intend to maintain several knowledge bases in order to cope with permanent background changes, a multiple of 20 MB of memory is required for the color knowledge bases. Moreover, increasing the image size to 384×288 leads to a requirement of 81 MB of memory for just one knowledge base for each pixel.

2.1.2 Edge-Based Segmentation

Basically, the edge-based segmentation works out foreground edges by a subtraction of known background edges from current video images. For this purpose, first of all edges in video images have to be determined.

There are a lot of well-known methods for edge detection in the field of image processing [GON92]. Good results have been achieved by using gradient-based filter functions like the *Sobel* or the *Laplacian* operator. Such an edge operator is applied to each of the N_K color channels of an image separately, providing N_K edge values $e(\vec{p}) = (e_1(\vec{p}), ..., e_{N_K}(\vec{p}))$ for every pixel $\vec{p} = (x, y)$. In order to abstract from any specific edge filtering method, we assume $e_i(\vec{p}) \in [0, 1]$ (with $i = 1, ..., N_K$). $e_i(\vec{p})$ is considered to be a fuzzy membership value of the semantic property "is edge". An edge value $e_i(\vec{p}) = 0$ indicates that there is no edge at pixel position $\vec{p}$ in color channel i, while $e_i(\vec{p}) = 1$ denotes the maximum edge value. Such a general representation can be achieved by scaling. If the edge operator for example yields the discrete values $\tilde{e}_i(\vec{p}) \in \{0, ..., M\} \subset \mathbb{N}$, the formula

$$e_i(\vec{p}) = \frac{1}{M} \tilde{e}_i(\vec{p}) \tag{2.6}$$

will perform the transformation into the required interval $[0, 1]$. An example of application of an edge detection operator (in this case the *Sobel*-operator) is shown in figure 2.6.

As already mentioned, the edge-based segmentation is realized as a background subtraction approach comparable to the color-based segmentation. During the initial learning phase, edge knowledge is collected. Afterwards, during the application phase, the contour of foreground objects can be determined by subtracting known background edges. Moreover, there may be background edges that are occluded by foreground objects leading to significant differences at these pixel positions as well. In order to determine contour pixels of foreground objects, we

store and maintain two types of information for each pixel in so called *fuzzy edge knowledge bases*. First, we calculate the average edge values for each color channel of each pixel during the initial learning phase. Second, an indicator for the existence of an edge in the surroundings of each pixel is determined. During experimental tests, we have found that the existence of edges in the surroundings of a pixel under consideration is an important aspect because existing edges tend to change their appearance considerably when the illumination conditions are unstable. We maintain information about this aspect in order to cope with such effects. Background edge information is collected in a fuzzy edge knowledge base:

Definition 2.1.2 (Fuzzy edge knowledge base)

Let width w and height h, measured in pixels, be the size and N_K the number of color channels of the input images.

Then a fuzzy edge knowledge base *is defined by a set of functions*

$$\bar{e}_i : (x, y) \longrightarrow [0, 1] \tag{2.7}$$

where $\bar{e}_i(\vec{p})$ is the average edge value of color channel i at position $\vec{p} = (x, y)$ (with $i = 1, ..., N_K$ and $0 \leq x < w,\ 0 \leq y < h$). $\bar{e}_i(\vec{p})$ is determined by averaging during the initial learning phase and may be continuously adapted by, e. g., the calculation of the running average. Moreover,

$$\eta : (x, y) \longrightarrow [0, 1] \tag{2.8}$$

is an indicator for the existence of an edge in the surroundings of pixel $\vec{p} = (x, y)$. The indicator function η is defined by:

$$\eta(\vec{p}) = \max\{\max\{\bar{e}_i(\vec{p}), \max_{j=1,...,8} \bar{e}_i(\vec{p}_j)\} |\ i = 1, ..., N_K\} \tag{2.9}$$

where $\vec{p}_j$ are the surrounding pixels of $\vec{p}$ concerning the 8-neighborhood relationship of pixels.

The acquisition of the average background edges during the initial learning phase is realized by averaging the edge values of the N_S sample images I_j for every pixel $\vec{p}$ and color channel i separately (with $i = 1, ..., N_K$ and $j = 1, ..., N_S$):

$$\bar{e}_i(\vec{p}) = \frac{1}{N_S} \sum_{j=1}^{N_S} e_{i,j}(\vec{p}) \tag{2.10}$$

where $e_{i,j}(\vec{p})$ is the edge value of color channel i in the sample image I_j at the pixel position $\vec{p}$.

The classification of current video images during the application phase is performed by comparing the edge knowledge base to the current edge values $e(\vec{p}) = (e_1(\vec{p}), ..., e_{N_K}(\vec{p}))$. In order to classifiy current images I, we have developed the following fuzzy membership function for the semantic property "is foreground edge":

$$\mu_e(\vec{p}) = 1 - \frac{1}{1 + \frac{1}{N_K}[\alpha + \beta(1 - \eta(\vec{p}))^2] \sum_{i=1}^{N_K} (\bar{e}_i(\vec{p}) - e_i(\vec{p}))^2} \tag{2.11}$$

where

$$\begin{array}{rcl} e_i(\vec{p}) & : & \text{current edge value of color channel } i \text{ at position } \vec{p} \\ \bar{e}_i(\vec{p}), \eta(\vec{p}) & : & \text{edge knowledge as provided by edge knowledge base} \\ N_K & : & \text{number of color channels} \\ \alpha, \beta \in \mathbb{R} & : & \text{controlling factors.} \end{array}$$

Briefly described, the fuzzy membership function μ_e sums up the differences between learned background edges and current edge values and divides it by the number of color channels N_K. The term

$$\alpha + \beta(1 - \eta(\vec{p}))^2 \tag{2.12}$$

is a function that provides a controlling factor with respect to the determined existence of known background edges. If a background edge exists in the surroundings of pixel $\vec{p}$, the resulting factor will be small. On the contrary, the factor will be large, if a background edge does not exist in this area.

In order to remove noise, an averaging threshold filter as explained in section 2.1.1 is applied to the fuzzy classified edge membership values $\mu_e(\vec{p})$:

$$\mu_{edge}(\vec{p}) = \begin{cases} 0 & : \quad \frac{1}{9}\mu_e(\vec{p}) + \frac{1}{9}\sum_{i=1}^{8} \mu_e(\vec{p}_i) \ \leq \ \upsilon_e \\ \mu_e & : \quad \frac{1}{9}\mu_e(\vec{p}) + \frac{1}{9}\sum_{i=1}^{8} \mu_e(\vec{p}_i) \ > \ \upsilon_e \end{cases} \tag{2.13}$$

where $\upsilon_e \in [0, 1]$ is used as a threshold parameter. Alternatively, a *median filter* may be utilized for this purpose. The membership function μ_{edge} yields a second fuzzy membership value for every pixel $\vec{p} \in I_c$. Both membership functions, μ_{color} (see section 2.1.1) and μ_{edge}, semantically describe foreground object properties.

2.1.3 Combination of Fuzzy Knowledge

In the previous sections 2.1.1 and 2.1.2, two independent fuzzy knowledge bases have been introduced, which provide the membership functions μ_{color} and μ_{edge} for each pixel $\vec{p}$ of a current image I. The membership functions semantically describe the foreground object properties "is foreground color" and "is foreground edge", respectively. In order to combine both results, a fuzzy "and" operator, e. g., the $\min$-function is employed (see appendix A). Thus, for every pixel $\vec{p}$ a fuzzy result concerning the semantic property "is foreground" is yielded by calculating

$$\mu_{foreground}(\vec{p}) = \min(\mu_{color}(\vec{p}), \ \mu_{edge}(\vec{p})). \tag{2.14}$$

For some applications, a two-valued segmentation result may be necessary. This is achieved by a defuzzification of the fuzzy results. For this purpose, the averaging threshold filter is modified:

$$\vec{p} \longrightarrow \begin{cases} 0 & : \quad \frac{1}{N_N}[\mu_{foreground}(\vec{p}) + \sum_{i=1}^{N_N-1} \mu_{foreground}(\vec{p}_i)] \ \leq \ \upsilon_a \\ 1 & : \quad \frac{1}{N_N}[\mu_{foreground}(\vec{p}) + \sum_{i=1}^{N_N-1} \mu_{foreground}(\vec{p}_i)] \ > \ \upsilon_a \end{cases} \tag{2.15}$$

where

$\vec{p}_i$: surrounding pixels of $\vec{p}$ that fulfill the 8-neighborhood relationship
N_N : number of pixels in the 8-neighborhood (may be less than eight)
$\upsilon_a \in [0,1]$: threshold parameter.

2.1.4 Learning and Continuous Updating

Our technique is intended to be adaptive to slight changes in the background. Especially the color knowledge base is able to store a large amount of histogram information for each pixel separately. Thus, it is possible to add further color information to the knowledge base during the application phase, when background regions are determined. As briefly explained in section 1.1, the low-level processing stage gets a feedback from the mid-level stage containing the information at which pixel positions the current image content has to be learned as "background" and at which positions the current image content is "foreground". In the recognized background areas the current colors are learned and incorporated into the color knowledge base as explained in section 2.1.1.

The edge knowledge base is not able to store different edge values per pixel, but only the average edge value for each color channel. However, the background is required to be static, so edge values are not likely to change completely. Nevertheless, edge values slightly change under varying illumination. Thus, it is advantageous to adapt the fuzzy edge knowledge base during the application phase. For this purpose, the learned background edge values $\bar{e}_i(\vec{p})$ are adapted to current edge values $e_i(\vec{p})$ (with $i = 1, ..., N_K$) using the running average:

$$\bar{e}'_i(\vec{p}) = \gamma e_i(\vec{p}) + (1-\gamma)\bar{e}_i(\vec{p}) \tag{2.16}$$

with $\gamma \in [0,1]$. $\bar{e}'_i(\vec{p})$ denotes the new entry of the edge knowledge base that replaces $\bar{e}_i(\vec{p})$. The parameter γ regulates the impact of the current edge values where $\gamma = 0$ corresponds to "no effect" and $\gamma = 1$ replaces $\bar{e}_i(\vec{p})$ with $e_i(\vec{p})$. Each time the edge knowledge base is updated with current edge values, the indicator function η is updated as well. If a significant change of a large area of the image is detected, new instances of color and edge knowledge bases will be created for these regions. This mechanism is controlled by the mid- and high-level processing stage and explained in section 2.4.

2.1.5 Supplementary Image Processing

During experimental evaluations, we have found that some supplementary image processing operations are useful in order to increase the reliability and overall performance of the segmentation. Basically, we need two different input sources, namely a color image and an edge-filtered image. In order to yield a smooth color image, we apply a *median* filter to the original input image that smoothes the image and removes noise. For the determination of the edge-filtered input image, we apply a combination of different *Sobel* operators [GON92], namely the horizontal, the vertical, and the diagonal filter kernels in order to yield as much edge information as possible. Afterwards, we apply a *Gaussian* low-pass filter operator in order to smooth the edge image. The averaging threshold filters are controlled by the parameters υ_c, υ_e, υ_a, α, and β as explained in equations (2.4), (2.13) and (2.15) and the classification membership function in equation (2.11). Depending on the specific application, it is not possible to

specify universally valid optimal values for these parameters. The quality of any choice can only be evaluated with respect to the classification quality in specific applications. However, good results have been achieved with $\alpha = 10$, $\beta = 90$, $\upsilon_e = 0.6$, $\upsilon_c = 0.7$ and $\upsilon_a = 0$.

2.1.6 Examples of Application

Because of the variety of known segmentation techniques, it is rather difficult to compare them to our introduced approach. In principle, both the color and the edge segmentation are comparable to the common background subtraction approach. Our technique supplements this approach by enabling the storage of a large amount of color values, which are weighted with respect to the quantity of their occurrence. In this section, we aim at briefly demonstrating and illustrating the capabilities of the introduced segmentation approach. An extensive evaluation of the whole segmentation system is performed in section 2.5.

The video images have a size of 192×144 pixels and the RGB color space was chosen ($N_K = 3$, $N_W = 256$). The knowledge bases were trained with $N_S = 80$ sample images during the initial learning phase. After this training, some interaction was performed in front of the camera. During this interaction, the fuzzy color knowledge base was continuously trained on the identified background regions. The parameter σ, introduced in equation (2.2), which controls the speed of learning as far as the color knowledge base is concerned, was set to $\sigma = 0.18$.

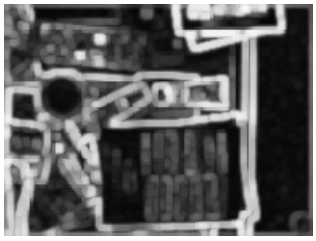
In figure 2.7, the segmentation task is to find and determine the position and the contour of the user's hand. The result of the segmentation is taken as input for the *Zyklop* hand posture classification system that is explained in chapter 4.3. As can be seen in figure 2.7(a), the background (in this case an office desk) is allowed to consist of arbitrary things. The exemplary image was taken after 350 images. In figure 2.7(b) and (c), the edge knowledge base is visualized, in (b) the averaged edge values in the R-, G- and B-color channels and in (c) the maximum edge value in the surroundings of each pixel. In figure 2.7(e), the result of the application of function μ_e (see equation (2.11)) is illustrated. Afterwards, it is filtered using the averaging threshold filter, which is shown in (f). Dark colors indicate low, and bright colors high membership degrees. In figure 2.7(g) and (h), the color segmented result (see equation (2.3)) and the filtered color segmentation result are shown, respectively. Finally, the results of the edge and the color segmentation (figure 2.7(f) and (h)) are combined in (i). Figure 2.7(j) shows the final result of segmentation. An approximated contour, superimposed in yellow, is calculated for the determined foreground object pixels using the contour approximation algorithm that is presented in chapter 3. Moreover, an enlarged convex hull for the foreground pixels is determined. Outside the hull polygon, the color and edge knowledge bases are adaptively trained on the current image content (indicated by the darkened area in the image). As can be seen in figure 2.7(h), the color segmentation wrongly classifies a small area of pixels above the user's hand. Due to the combination of color and edge segmentation, these faults can be corrected.

In figure 2.8, another example of application is illustrated. As can be seen in figure 2.8(f) and (h), both the edge and the color segmentation provide faulty results. The edge segmentation misclassifies a few pixels on the right of the user's hand and the color segmentation yields several wrongly classified pixels. However, the combination of color and edge segmentation in figure 2.8(i) eliminates the misclassifications, leading to good results in figure 2.8(j).

The second key benefit of our approach, the adaptiveness to slight changes in the background colors, is illustrated in another example (figure 2.9). There are already changes in the

(a) Original

(b) Learned background edges (function $\bar{e}$)



(c) Maximum edge values (function η)

(d) Current edge image

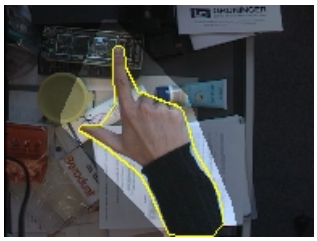
(e) Differences in edge values (function μ_e)

(f) Filtered edge result (function μ_{edge})

(g) Color-segmented image (function μ_c)

(h) Filtered color result (function μ_{color})

(i) Combination of color and edge results



(j) Final result, contour approximation (in yellow)

Figure 2.7: Example of application of low-level processing stage.

background colors when a foreground object enters the scene, which may be related to automatic adjustments of the camera. The image in figure 2.9(a) was recorded immediately after the arm was moved into the scene. The result of color segmentation is illustrated in (b) and as can be seen, it is of poor quality. The recorded series consists of 547 images and after completion, the series is repeated from the beginning with the trained knowledge bases. During the

(a) Original

(b) Learned background edges (function $\bar{e}$)

(c) Maximum edge values (function η)

(d) Current edge image

(e) Differences in edge values (function μ_e)

(f) Filtered edge result (function μ_{edge})

(g) Color-segmented image (function μ_c)

(h) Filtered color result (function μ_{color})

(i) Combination of color and edge results

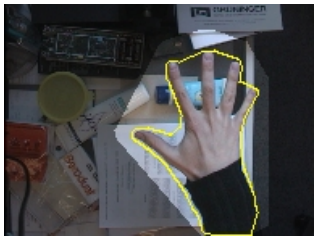
(j) Final result, contour approximation (in yellow)

Figure 2.8: Misclassifications of both the edge and the color segmentation, in (f) and (h), are eliminated due to the combination of both methods, shown in (i).

(a) Original

(b) Color segmented image, first run

(c) Color segmented image, second run



(d) Result of segmentation

Figure 2.9: Example for improvement of color segmentation due to continuous adaptiveness.

second run of the series, the color segmentation of the same input image provides considerably better, although not perfect results, as illustrated in figure 2.9(c). However, the combination of color and edge segmentation, which has been performed both during the first and the second run, yields a sufficient result as shown in 2.9(d).

2.2 Mid-Level Processing Stage

For the mid-level processing stage, the image area is subdivided in small rectangular box areas. Basically, the mid-level processing stage extends the idea of background subtraction to box-sized larger regions. For example, we used 12×9 boxes on images with a size of 192×144 pixels. For each of the boxes, a pattern recognition system is trained, which extracts features of the box regions during the initial learning phase. Wrongly classified pixels of the low-level processing stage can be eliminated by identifying foreground and background regions autonomously. Consequently, box regions which have been recognized as background can be masked out. As an important aspect, the change detection or similarity measures introduced in this section are intended to be largely independent of changing or varying illumination conditions. Moreover, the mid-level processing stage is able to administrate different low-level knowledge bases for each box area separately. This enables the segmentation algorithm to cope with permanent changes in the background by instantiating new low-level knowledge bases for the respective box areas. If the old background situation reoccurs, the system is able to recognize this and will use the formerly learned knowledge bases again. This methodology was illustrated in figure 1.1 in chapter 1.1 and is realized by utilizing the pattern recognition system.

In the next sections, we introduce the change detection measures (section 2.2.1), propose a rule-based classification method (section 2.2.2), evaluate the measures' performance empirically (section 2.2.3), and explain the application of the mid-level processing stage in the overall context (section 2.2.4).

2.2.1 Change Detection Measures

In this section, five change detection measures are proposed. The background is observed and typical features of the background are extracted during the initial learning phase. Afterwards, the detectors are able to measure the similarity between the initially learned content and current video images. The definition of the detectors is general and they are intended to be applied for each box area separately. As a result, a fuzzy membership value that describes the degree of membership to the semantic property "is background" is obtained for each box area.

2.2.1.1 Edge-Based Mutual Similarity Measure

The first detector does not work on color values, but on edge information within the image data, which is obtained by applying an edge detection operator as explained in section 2.1.2. In order to abstract from any specific edge operator, we assume the results of an edge filtering algorithm to be edge values $e_i(\vec{p}) \in [0,1]$, where $i \in \{1,...,N_K\}$ denotes the color channel and $\vec{p}$ the pixel under consideration. Exemplary methods to achieve such a general representation are explained in section 2.1.2.

One major aspect of using edge information for similarity comparisons is the desire for an illumination independent measure. However, edge information also depends on varying illumination conditions because a bright illumination leads to larger edge values than a dark illumination. In order to have a similarity measure that is able to cope with these influences, a mutual comparison scheme for the edge values is introduced. Let N_P denote the number of pixels under consideration. In order to measure the mutual difference between the N_P pixels, a feature $N_P \times N_P$-matrix $\underline{M}$ is used, whose elements are defined by

$$m_{ij} = \max_{k=1,...,N_K} e_k(\vec{p}_i) \; - \; \max_{k=1,...,N_K} e_k(\vec{p}_j) \tag{2.17}$$

Learning Phase

During the initial learning phase, an $N_P \times N_P$-matrix $\underline{R}$ is calculated for each box area separately. For each image I_t (with $t = 1,...,N_T$) occurring during the learning phase, a corresponding feature matrix $\underline{M}_t$ is determined by analyzing the box area under consideration. The matrix $\underline{R}$ is calculated as:

$$\underline{R} = \frac{1}{N_T} \sum_{t=1}^{N_T} \underline{M}_t \tag{2.18}$$

Averaging the differences between the edge values of a sequence of N_T image frames yields the matrix $\underline{R}$. For the elements r_{ij} of $\underline{R}$ the equations

$$r_{ij} = -r_{ji} \tag{2.19}$$

and

$$r_{ij} = 0 \text{ with } \; i = j \tag{2.20}$$

hold. Thus, it is sufficient to calculate only either the upper or the lower triangular matrix.

Application Phase

During the application phase, the similarity between the current video image I_t and the initially learned images is measured by comparing the mutual differences $\underline{M}_t$ between the edge values of I_t and $\underline{R}$. The membership value to the semantic property "similar" is calculated as

$$\mu_{releedgesim}(\underline{M}_t) = \frac{1}{1 + \frac{\delta_1}{N_D} \sum_{i=1}^{N_P} \sum_{j=i+1}^{N_P} (m_{ij} - r_{ij})^2} \tag{2.21}$$

where

m_{ij}	:	elements of the current feature matrix $\underline{M}_t$
r_{ij}	:	elements of averaged differences matrix $\underline{R}$
N_D	:	number of elements in the triangular matrix, $N_D = \frac{1}{2} N_P(N_P + 1)$
$\delta_1 \in \mathbb{R}$	:	controlling factor.

Adaptive Update

Realizing the unstableness in digital video images over time raises the question of adaptiveness as far as the membership function is concerned. Updates of the feature matrix $\underline{R}$ with current feature matrices $\underline{M}_t$ should be performed only if $\underline{M}_t$ represents the background and not an object which occasionally moved in the foreground. Otherwise wrong image content is learned as background. In order to avoid this, the system relies on five detectors and a fault-tolerant fuzzy rule-based classification process (section 2.2.2).

The centroid matrix $\underline{R}$ can be updated with a current feature matrix $\underline{M}_{t+1}$ to $\underline{R}_{t+1}$ by applying the following formula:

$$\underline{R}_{t+1} = \frac{t \cdot \underline{R}_t + \underline{M}_{t+1}}{t+1} \tag{2.22}$$

The introduced detector $\mu_{releedgesim}$ may work very well and reliably, if the observed background area contains a lot of edge pixels and if differences between them are noticeable. Nevertheless, this detector may partly fail, if the observed area is smooth because a smooth foreground object may not be recognized.

2.2.1.2 Mean Color Similarity Measure

Realizing the problem that absolute values, both color or edge values, are heavily depending on the illumination conditions, we also propose a mutual color measure that additionally describes the structure of the considered box area.

The box area itself is subdivided in arbitrary parts. In each part, the mean color value of each color channel is determined and the difference to the mean color values of the other parts is calculated. The determined differences are employed to measure the similarity between current and initially learned images. We usually subdivide the box areas in four equally sized quarters. Figure 2.10 illustrates the idea for the subdivision in quarters. For the *mean color similarity* measure, the differences of the mean color values in the quarters are calculated and considered as feature. The arrows indicate the differences which are calculated for the upper left part of the box area.

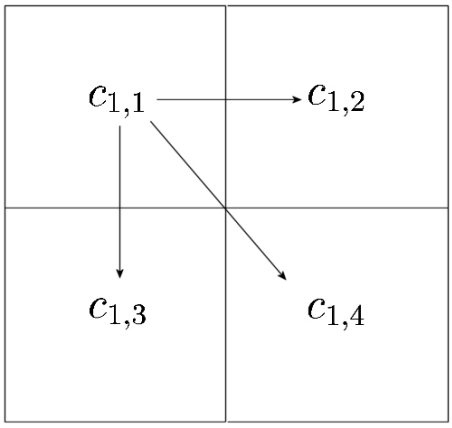


Figure 2.10: Example of *mean color similarity* application. The box area is subdivided in four equally sized quarters. The mean color values for each color channel in each quarter are calculated, here for color channel 1. The differences between the $c_{k,i}$ are measured as indicated by the arrows for the first, upper left quarter.

Learning Phase

During the learning phase, N_T images are used to calculate the averaged differences. For each of the N_K color channels, a feature matrix $\underline{R}_k$ (with $k = 1, ..., N_K$) is required. Averaging the color values in the N_A parts A_i of the box areas yields the mean color values $c^t_{k,i}$ (with $i = 1, ..., N_A$ and $t = 1, ..., N_T$) for each of the N_K color channels. Moreover, the color values are transformed into the interval $[0, 1]$, so $c^t_{k,i} \in [0, 1]$ holds. This transformation can easily be achieved by scaling.

The differences between the box area parts are determined and averaged for all the N_T images of the learning phase. This results in the $N_A \times N_A$ matrices $\underline{R}_k$ whose elements $r_{k,ij}$ are calculated as:

$$r_{k,ij} \;=\; \frac{1}{N_T} \sum_{t=1}^{N_T} c^t_{k,i} - c^t_{k,j} \tag{2.23}$$

where

$r_{k,ij}$	:	elements of the averaged difference matrix $\underline{R}_k$
$c^t_{k,i}$	:	mean color value of color channel k in area A_i in frame t.
N_T	:	number of frames during the learning phase.

Similar to the edge-based mutual similarity measure introduced in section 2.2.1.1, the following equations hold for the elements $r_{k,ij}$ of $\underline{R}_k$:

$$r_{k,ij} = -r_{k,ji} \tag{2.24}$$

and

$$r_{k,ij} = 0 \text{ for } \; i = j. \tag{2.25}$$

Thus, it is sufficient as well to calculate either the upper or the lower triangular matrix for this detector.

Application Phase

During the application phase, a current image I_n and more specific the current box area can be compared to the initially learned box areas. For that purpose, the current average color values $c^n_{k,i}$ in each part A_i of the box area (with $i = 1, ..., N_A$) and for each color channel $k = 1, ..., N_K$ are calculated again. The current feature matrices $\underline{M}_k$ are determined analogously to the learning phase:

$$m_{k,ij} \;=\; c^n_{k,i} - c^n_{k,j} \tag{2.26}$$

The similarity between the initially learned image content and the current image content can be measured with the following fuzzy membership function:

$$\mu_{meancolsim}(\underline{M}_1, ..., \underline{M}_{N_K}) = \frac{1}{1 + \frac{\delta_2}{N_D} \sum_{k=1}^{N_K} \sum_{i=1}^{N_A} \sum_{j=i+1}^{N_A} (\frac{m_{k,ij} - r_{k,ij}}{2})^2} \tag{2.27}$$

where

$m_{k,ij}$	:	elements of the current feature matrix $\underline{M}_k$
$r_{k,ij}$	:	elements of averaged differences matrix $\underline{R}_k$
N_D	:	number of elements under consideration, $N_D = \frac{1}{2} N_K N_A (N_A + 1)$
N_A	:	number of box areas
N_K	:	number of color channels under consideration
$\delta_2 \in \mathbb{R}$	:	controlling factor.

High membership degrees of $\mu_{meancolsim}$ denote a high similarity between the initially learned background and the current image content and vice versa. As experiments have shown, this detector is indeed rather insensitive to changing illumination conditions. Its robustness is discussed in more detail in section 2.2.3.

An adaptive update of the initially learned centroid matrices $\underline{R}_k$ can be performed analogously to formula (2.22) presented in section 2.2.1.1.

2.2.1.3 Mean Edge Similarity Measure

The mean color similarity measure introduced in the previous section 2.2.1.2 is a rather general approach to image content comparison. Thus, we propose to use the same detector with color edge images as input. Such an edge image can be easily derived from the original image by applying an edge operator such as the Sobel or the Prewitt operator [GON92]. This approach is principally similar to the edge-based mutual similarity measure (see section 2.2.1.1), except for the fact that the edge values are averaged for a larger part of the box area and that an arbitrary number of color channels can be taken into consideration. Averaging the edge values for a larger part of the box area seems to be reasonable because aliasing effects may cause edges to move for some pixels around their originally determined position.

Due to the identical structure of the detector (only the input image is altered), we do not repeat the formulas and refer to section 2.2.1.2 for details. The similarity is measured by

$$\mu_{meanedgesim}(\underline{M}_1, ..., \underline{M}_{N_K}) = \frac{1}{1 + \frac{\delta_3}{N_D} \sum_{k=1}^{N_K} \sum_{i=1}^{N_A} \sum_{j=i+1}^{N_A} (\frac{m_{k,ij} - r_{k,ij}}{2})^2} \tag{2.28}$$

where $\delta_3 \in \mathbb{R}$ is a controlling factor.

2.2.1.4 Sum of Foreground Edges Measure

As already explained in section 2.1, the low-level segmentation performs a color and an edge segmentation, which are based on calculating differences between the initially learned background and current video images. As a result, the edge segmentation returns for each pixel $\vec{p} = (x, y)$ a fuzzy membership value $\mu_e(\vec{p}) \in [0, 1]$ to the semantic property "is foreground edge" (equation (2.11)). Taking into account the semantic notion of "is foreground edge", high membership degrees denote a foreground edge, while low membership degrees indicate that there is no edge at pixel position $\vec{p}$. Thus, a simple change detector sums up the determined foreground edges within the box area.

The basic function for the determination of foreground edges was introduced in equation (2.11) and contains a controlling factor function as shown in equation (2.12). Experimental results have proven that this controlling factor function is well suited for the determination of foreground edge pixels. However, we obtained better results for the purposes of a similarity measure with a fixed factor instead, regardless of how edgy the background is. Thus, we modify equation (2.11) slightly to:

$$\mu_{\tilde{e}}(\vec{p}) = 1 - \frac{1}{1 + \frac{1}{N_K}\alpha_f \sum_{i=1}^{N_K} (\bar{e}_i(\vec{p}) - e_i(\vec{p}))^2} \tag{2.29}$$

where

$e_i(\vec{p})$	:	current edge value of color channel i at position $\vec{p}$
$\bar{e}_i(\vec{p})$	:	edge knowledge as provided by edge knowledge base
N_K	:	number of color channels
$\alpha_f \in \mathbb{R}$	:	fixed controlling factor.

The foreground edge detection measure first of all determines $\mu_{\tilde{e}}(\vec{p})$ for all the N_P pixels $\vec{p}$ of the image area under consideration. Then a feature vector $\vec{M} = (\mu_{\tilde{e}}(\vec{p}_1), ..., \mu_{\tilde{e}}(\vec{p}_{N_P}))^T$ is used to measure the similarity to initially learned background images:

$$\mu_{edgesum}(\vec{M}) = \frac{1}{1 + \frac{\delta_4}{N_P} \sum_{i=1}^{N_P} \mu_{\tilde{e}}(\vec{p}_i)^2} \tag{2.30}$$

with $\delta_4 \in \mathbb{R}$ as a controlling factor. High membership values of $\mu_{edgesum}$ denote a similar or even identical box area content to the initially learned content, whereas low membership values denote a significant change within the box area.

This detector does not require any additional handling during the initial learning phase because the necessary collection of background edge information is already done within the low-level processing stage.

2.2.1.5 Sum of Foreground Pixels Measure

Analogously to the *sum of foreground edges* measure, we propose a detector that measures the number of foreground pixels, which have been determined by the color segmentation within the low-level processing stage (section 2.1.1). As a result of the color segmentation, a fuzzy membership value $\mu_c(\vec{p}) \in [0, 1]$ to the semantic property "is foreground color" is returned for

each pixel $\vec{p} = (x, y)$ (equation (2.3)). Thus, a simple change detector sums up the determined foreground pixels within the box area. Let the box area consist of N_P pixels, then a feature vector $\vec{M} = (\mu_c(\vec{p}_1), ..., \mu_c(\vec{p}_{N_P}))^T$ can be used to measure the similarity to initially learned background images:

$$\mu_{colorsum}(\vec{M}) = \frac{1}{1 + \frac{\delta_5}{N_P} \sum_{i=1}^{N_P} \mu_c(\vec{p}_i)^2} \tag{2.31}$$

with $\delta_5 \in \mathbb{R}$ as a controlling factor. High membership values of $\mu_{colorsum}$ denote a similar or even identical box area content to the initially learned content, whereas low membership values denote a significant change within the box area.

This detector does not require any additional handling during the initial learning phase because the necessary collection of background color information is already done within the low-level processing stage.

2.2.2 Framework for the Classification Process

So far we have introduced five different similarity measures which reflect the similarity between a current image and known background for each box area separately. In this section, we present a fuzzy rule-based classification method that allows the incorporation of an arbitrary number of fuzzy detector results. Realizing the strong dependency of each box on the respective background content, we supplement each detector with two reliability indicators. The first reliability indicator gives information about the reliability of high membership degrees of the detector, i. e., the reliability of a "background" classification. Vice versa, the second reliability indicator provides information about the reliability of low membership degrees of the detector, i. e., a "foreground" classification. Due to this approach, we are able to take into account the possibly different applicability of each detector to the respective box content.

2.2.2.1 Fuzzy Rule Base for Evaluation

In order to have a common basis for the evaluation of the obtained results, we specify a fuzzy logic rule base that is able to take into account the results of the similarity measures on the one hand and the reliability indicators of the detectors on the other hand. Moreover, this approach enables the evaluation of an arbitrary number of detectors. It can be extended easily for further similarity measures, which may be developed in future work.

Let μ_i with $i = 1, ..., N_D$ denote the fuzzy membership functions of the similarity measures we have defined so far, where N_D is the number of detectors. Then we introduce two fuzzy reliability indicators $\mu_{i,bgr}$ and $\mu_{i,fgr}$ for each detector. High membership degrees of $\mu_{i,bgr}$ indicate a high reliability of a "background" (*bgr*) classification of detector i. Analogously, a high membership degree of $\mu_{i,fgr}$ indicates a high reliability of a "foreground" (*fgr*) classification of detector i. In the context of the feedback mechanisms, the reliability indicators will be continuously manipulated, if poor classification results of respective similarity measures for specific box areas are noticed by the high-level processing stage. The details concerning this aspect are explained in section 2.4.

In order to interpret the results of the detectors with respect to their reliability indicators and to find a final classification result, we utilize a fuzzy rule base, which is designed following human notions of the decision making process. A brief course into fuzzy logic and the

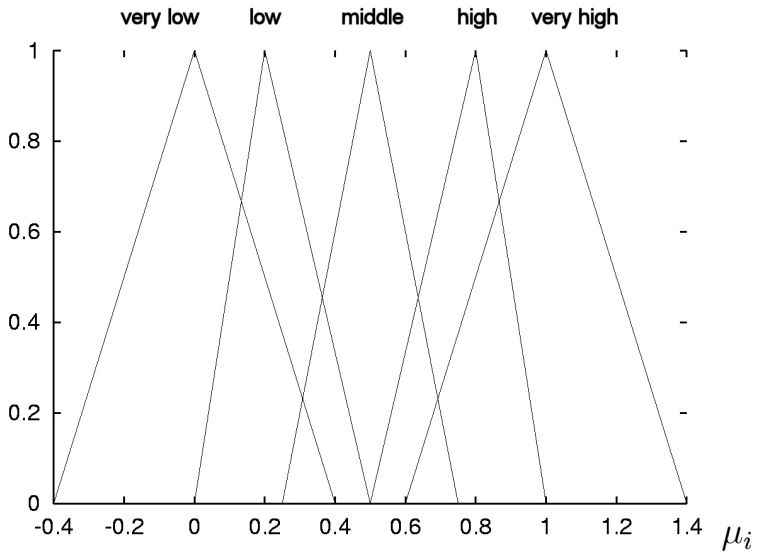


Figure 2.11: Fuzzy sets of the linguistic terms for the similarity measures.

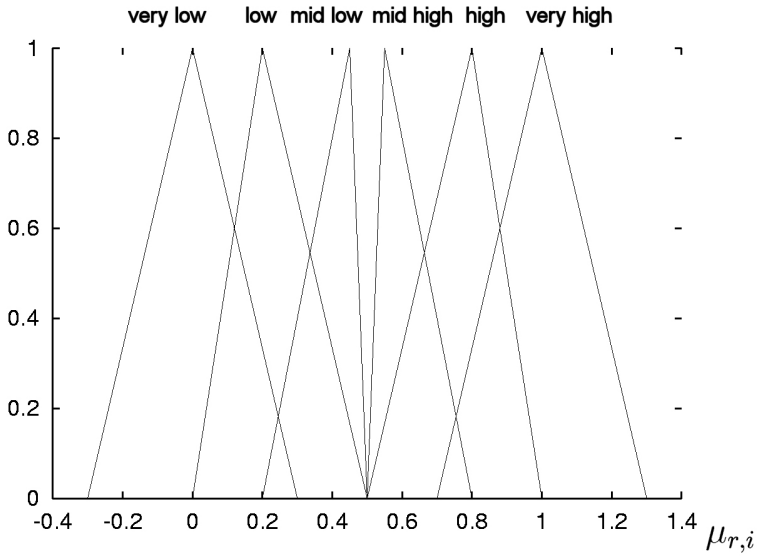


Figure 2.12: Fuzzy sets of the linguistic terms for the resulting fuzzy sets.

evaluation of fuzzy rule bases is made in appendix A. First of all, the measured similarities are interpreted in linguistic terms using the fuzzy sets *very low, low, middle, high* and *very high*. A possible definition of these fuzzy sets is illustrated in figure 2.11. We use the same ten rules for each detector (see table 2.1), which are mapped onto the resulting fuzzy sets $\mu_{i,r}$. For these resulting fuzzy sets, the linguistic terms *very low, low, mid low, mid high, high* and *very high* are used (see figure 2.12).

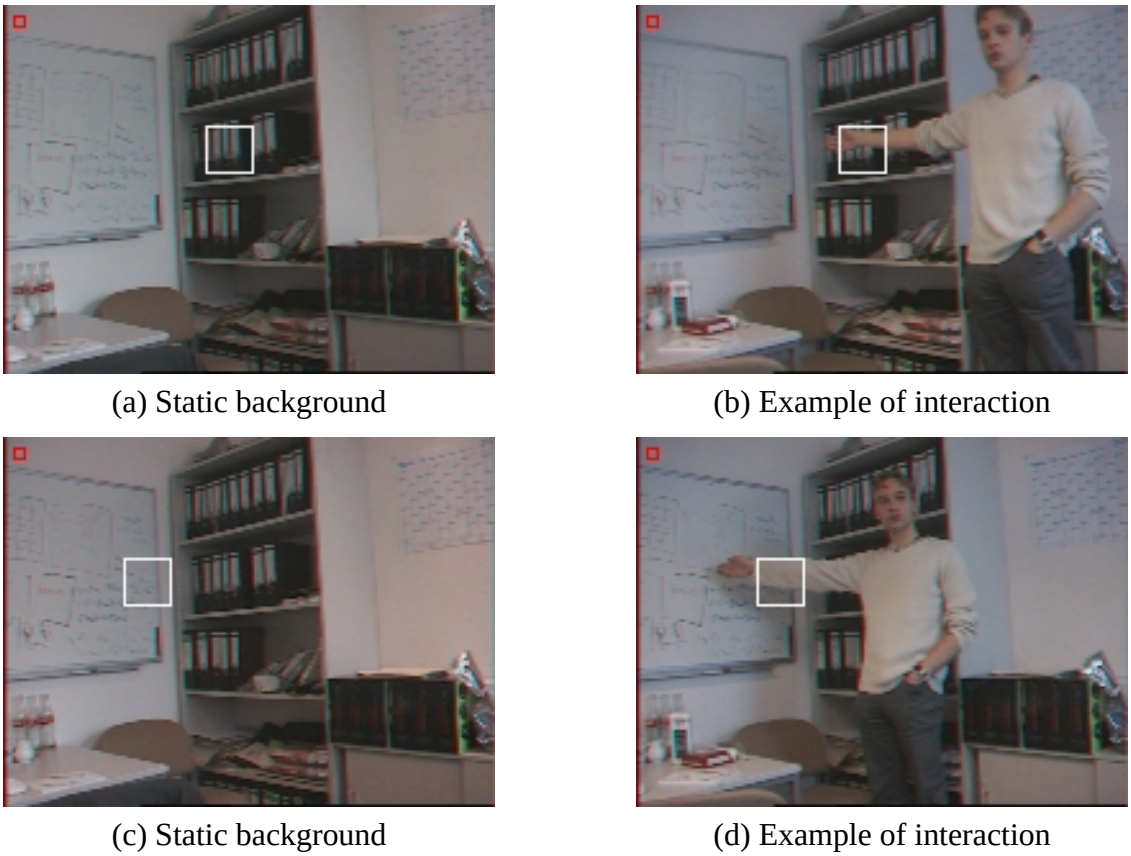
This rule base is applied for each detector. For the inference, we use the $\min$-function and for the aggregation within the evaluation of one detector the $\max$-function. Thus, we get a resulting fuzzy set $\mu_{i,r}$ for each detector. The N_D different $\mu_{i,r}$ are aggregated with the sum-function in order to weigh the preliminary results equally. For the logic operators *and, or* and

IF	μ_i = very high	AND	$\mu_{i,bgr}$ = (high OR very high)
	THEN $\mu_{i,r}$ = very high		
IF	μ_i = high	AND	$\mu_{i,bgr}$ = (high OR very high)
	THEN $\mu_{i,r}$ = high		
IF	μ_i = (high OR very high)	AND	$\mu_{i,bgr}$ = (high OR middle)
	THEN $\mu_{i,r}$ = high		
IF	μ_i = (high OR very high)	AND	$\mu_{i,bgr}$ = (low AND NOT very low)
	THEN $\mu_{i,r}$ = middle high		
IF	μ_i = (middle AND NOT low)	AND	$\mu_{i,bgr}$ = (high OR very high)
		AND	$\mu_{i,fgr}$ = (low OR very low)
	THEN $\mu_{i,r}$ = middle high		
IF	μ_i = (middle AND NOT high)	AND	$\mu_{i,fgr}$ = (high OR very high)
		AND	$\mu_{i,bgr}$ = (low OR very low)
	THEN $\mu_{i,r}$ = middle low		
IF	μ_i = (low OR very low)	AND	$\mu_{i,fgr}$ = (low AND NOT very low)
	THEN $\mu_{i,r}$ = middle low		
IF	μ_i = (low OR very low)	AND	$\mu_{i,fgr}$ = (high OR middle)
	THEN $\mu_{i,r}$ = low		
IF	μ_i = low	AND	$\mu_{i,fgr}$ = (high OR very high)
	THEN $\mu_{i,r}$ = low		
IF	μ_i = very low	AND	$\mu_{i,fgr}$ = (low OR very low)
	THEN $\mu_{i,r}$ = very high		

Table 2.1: Fuzzy rule base for the decision making process.

not, the standard functions are used as introduced in appendix A. We take the *center of area* approach for the defuzzification and use 0.5 as threshold to obtain a two-valued result. The classification result for the box area under consideration is ”background”, if the defuzzified result is above 0.5, or ”foreground”, if the defuzzified result is below 0.5.

The classification is performed for each box area by the mid-level processing stage. Pixels which have been misclassified as ”foreground pixel” by the low-level processing stage can be masked out in recognized ”background” box regions. Due to the identification of ”background” areas, the adaptive learning process of the low-level processing stage can be controlled. The details of this part of the feedback mechanisms are explained in chapter 2.4.



(a) Static background (b) Example of interaction

(c) Static background (d) Example of interaction

Figure 2.13: (a) and (b): Example images for first test area (white) with edgy background; (c) and (d) second area with smooth background.

2.2.3 Evaluation of Detectors

During the introduction of the change detectors in the equations (2.21), (2.27), (2.28), (2.30) and (2.31), the parameters δ_1, δ_2, δ_3, δ_4 and δ_5 were used as controlling factors. These parameters have to be specified in order to be used within applications. We performed an empirical evaluation to find suitable settings for the parameters and to test their applicability. A typical scene has been observed by a camera for 24 hours. After each minute, the current image was saved leading to a large set of images with various illumination conditions. Rather and completely dark images were not used for the empirical evaluation. Afterwards, a person performed some typical interaction movements in front of the camera, for example the arm and/or the hand were moved into the image area under consideration or the user simply stood in front of the camera while moving his arm.

We chose two different box areas, one with a rather smooth and monotone background and one with an edgy background (see figure 2.13), and empirically determined suitable parameter settings for these box areas, using the following box sizes and image metrics:

- 192×144 pixels image size, 8×8 pixels box size
- 192×144 pixels image size, 16×18 pixels box size
- 384×288 pixels image size, 8×8 pixels box size

- 384×288 pixels image size, 16×16 pixels box size
- 384×288 pixels image size, 32×32 pixels box size
- 768×576 pixels image size, 16×16 pixels box size
- 768×576 pixels image size, 32×32 pixels box size
- 768×576 pixels image size, 64×64 pixels box size

These combinations of metrics seem to be reasonable as far as the relationship between box size and image size is concerned.

Our first aim was to determine a suitable set of parameters δ_i, which on the one hand assigns high membership values to the semantic property "is similar" for the first part of the recorded images containing the static background under varying illumination conditions, and on the other hand assigns low membership values to the second part of images containing an interacting person in front of the area under consideration.

As we have already found for the low-level processing stage, the existence and appearance of edges in the background areas under consideration are important aspects for the successful application of our classification methods. Moreover, some of the change detection measures such as the *mean edge similarity* or the *sum of foreground edges* directly take changes within the edge values into consideration. As a consequence, the parameterization of the classification functions should be chosen with respect to the number of edges in the background. If a lot of edges exist in the background, most probably a small change is sufficient to cause low membership degrees of the similarity measures. On the contrary, if the background contains only few or even no edges, probably a larger change will be necessary to lead to low membership degrees that signal a foreground object. In order to cope with this obvious relationship, we try to determine two separate optimal parameter settings for both of our evaluation series illustrated in figure 2.13. However, first of all we need a measure for the edginess of the image area under consideration. For this purpose, we propose a simple threshold analysis that counts the number of edges above a predefined threshold:

$$\mu_{edginess} = \frac{1}{N_P} \sum_{i=1}^{N_P} \omega(\vec{p_i}) \tag{2.32}$$

where ω is

$$\omega(\vec{p}) = \begin{cases} 0 & : \max\limits_{i=1,\ldots,N_C} \bar{e}_i(\vec{p}) < \upsilon_s \\ 1 & : \max\limits_{i=1,\ldots,N_C} \bar{e}_i(\vec{p}) \geq \upsilon_s \end{cases} \tag{2.33}$$

with

$\upsilon_s \in [0,1]$	:	threshold parameter
N_P	:	number of pixels of box area under consideration
N_C	:	number of color channels
$\bar{e}_i(\vec{p})$	:	average edge value in color channel i at pixel $\vec{p}$ (see chapter 2.1.2).

The detectors were trained during the initial learning phase with 80 images. Adaptive updates of the detectors during the application phase were not performed. Thus, each frame of

the different background situations is evaluated only on the basis of the initially learned background knowledge. The number of available video frames for each of the situations is shown in table 2.2. After some trial executions of the classification process, the parameter settings shown in table 2.3, 2.4, 2.5, and 2.6 were found to be suitable. In order to generalize the found parameter settings, we assume a linear interpolation for the parameter domain $\mu_{edginess} \in [0, 1]$ as shown in figure 2.14. As already explained, we neither adaptively updated the knowledge bases, nor the trained mid-level detectors during this evaluation. However, while this is reasonable as far as the first four similarity measures are concerned, it does not make sense for the fifth *sum of foreground pixels* measures whose recognition capabilities depend on the low-level color knowledge bases (see section 2.1.1). Continuous updating with current image content is essential for this detector. Because of this reason, this is the only similarity measure, for which updating is performed during this evaluation. Suitable parameters, which have been found during the evaluation, are illustrated in table 2.7. As it could be expected, the *sum of foreground pixels* measure in equation (2.31) does not depend on the edginess of the area under consideration because it works on color information only. Thus, it is not necessary to change this parameter.

Due to the fuzziness of the membership functions, the presented parameter settings may be modified slightly without changing the results noticeably. Applying this parameter setting to our test series yields the results shown in table 2.8. In order to evaluate each similarity measure separately, we assume a simple threshold defuzzification scheme. Membership values with $\mu_i \leq 0.5$ lead to the result "is foreground" and membership values $\mu_i > 0.5$ to "is background". The semantic property "is background" is identical with the property "is similar" in this case because we measure the similarity between the initially learned background and current video images, assuming that similar images correspond to "background".

Additionally, it has to be kept in mind that the classification results are fuzzy. For this analysis, the results have been defuzzified using a threshold. In some cases, classification results may have been counted as wrong, although the fuzzy classification result may have been rather close to the applied threshold. Thus, the fuzzy classification may already imply an uncertain classification which is lost after applying the threshold because it is not employed within this analysis. If only a rather small part of the considered box area is covered by a foreground object, the decline in the membership values may be not severe enough to drop the membership values below the threshold. Although the tendency may be correct, the two-valued classification result may be wrong in our analysis. In order to incorporate the different fuzzy results for the determination of a final classification result, we have presented a general framework in section 2.2.2.1, which is based on a fuzzy rule base. Moreover, reliability indicators can be maintained, which influence the weighting of the similarity measures with respect to their former classification performance. For these series, we set the reliability indicators to 1.0 for all the detectors and did not influence them during execution in order to determine the quality of the fuzzy classification process itself. The overall results of the fuzzy rule base classification are shown in table 2.9. Depending on the metrics of both the boxes and the image, the amount of correctly classified images is between 97.8% and 99.9%. Comparing the final classification result to its associated similarity measure results shows that the correctness rate of the final result is usually well ahead of the different similarity measure results. Only for an image size of 384×288 pixels and a box size of 16×16 pixels, the *sum of foreground pixels* measure is slightly better than the final result (98.5% compared to 98.1%). Despite this case, the fuzzy rule base provides better classification results than the respective similarity measures,

Number of Frames	image size	box size	smooth	edgy
different background situations	192×144	8×8	2163	1097
person in front of background	192×144	8×8	185	1377
different background situations	192×144	16×16	2163	1097
person in front of background	192×144	16×16	185	1413
different background situations	384×288	8×8	1291	1291
person in front of background	384×288	8×8	240	435
different background situations	384×288	16×16	1291	1291
person in front of background	384×288	16×16	307	514
different background situations	384×288	32×32	1291	1291
person in front of background	384×288	32×32	383	619
different background situations	768×576	16×16	1291	1291
person in front of background	768×576	16×16	243	439
different background situations	768×576	32×32	1291	1291
person in front of background	768×576	32×32	304	503
different background situations	768×576	64×64	1291	1291
person in front of background	768×576	64×64	389	562

Table 2.2: Number of available image frames for each situation.

Empirically determined δ_1

image size	box size	δ_1 edgy	δ_1 smooth
192×144	8×8	50	2000
192×144	16×16	70	120
384×288	8×8	40	2750
384×288	16×16	28	400
384×288	32×32	23	400
768×576	16×16	60	5000
768×576	32×32	25	600
768×576	64×64	30	800

Table 2.3: Determined δ_1 for different metrics and background situations.

which are used as input for the rule base. The fuzzy rule based approach is obviously able to compensate wrong results of the similarity measures in many cases.

We have shown that the introduced similarity measures can be parameterized successfully. They are applicable to a wide range of different image and box metrics. Moreover, we have shown that the fuzzy rule based classification is a reasonable approach for the evaluation of the similarity measures and yields good overall classification results. An in-depth analysis of the classification results in the context of the whole segmentation system is presented in chapter 2.5.3.

Empirically determined δ_2

image size	box size	δ_2 edgy	δ_2 smooth
192×144	8×8	1000	3500
192×144	16×16	3500	5000
384×288	8×8	2000	100000
384×288	16×16	12000	30000
384×288	32×32	35000	30000
768×576	16×16	6000	120000
768×576	32×32	10000	50000
768×576	64×64	50000	30000

Table 2.4: Determined δ_2 for different metrics and background situations.

Empirically determined δ_3

image size	box size	δ_3 edgy	δ_3 smooth
192×144	8×8	500	20000
192×144	16×16	1500	2875
384×288	8×8	250	25000
384×288	16×16	600	15000
384×288	32×32	1600	10000
768×576	16×16	600	90000
768×576	32×32	900	90000
768×576	64×64	4000	120000

Table 2.5: Determined δ_3 for different metrics and background situations.

Empirically determined δ_4

image size	box size	δ_4 edgy	δ_4 smooth
192×144	8×8	5	1000
192×144	16×16	7	30
384×288	8×8	4.5	1500
384×288	16×16	4.5	600
384×288	32×32	6	130
768×576	16×16	7	500
768×576	32×32	5.5	800
768×576	64×64	7.5	1000

Table 2.6: Determined δ_4 for different metrics and background situations.

Empirically determined δ_5

image size	box size	δ_5 edgy/smooth
192×144	8×8	5
192×144	16×16	5
384×288	8×8	5
384×288	16×16	3
384×288	32×32	3
768×576	16×16	3
768×576	32×32	3
768×576	64×64	3

Table 2.7: Determined δ_5 for different metrics, independent of background situation.

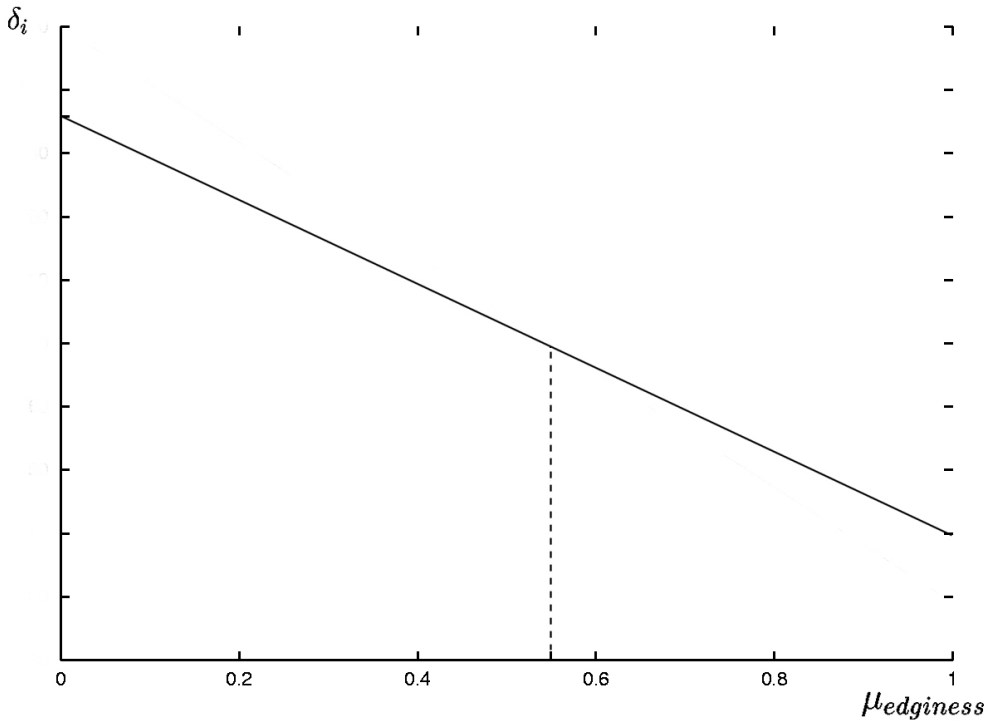


Figure 2.14: The parameters δ_i are calculated by linear interpolation of the empirically determined parameters.

Relative Edge-based Similarity Measure

Correctly Classified in [%]	8×8 boxes	16×16 boxes	32×32 boxes	64×64 boxes
192×144	98.4 %	99.9 %	-	-
384×288	96.9 %	96.9 %	94.7 %	-
768×576	-	96.5 %	98.2 %	96.5 %

Mean Color Similarity Measure

Correctly Classified in [%]	8×8 boxes	16×16 boxes	32×32 boxes	64×64 boxes
192×144	87.6 %	97.9 %	-	-
384×288	97.8 %	97.4 %	97.9 %	-
768×576	-	96.8 %	98.1 %	98.2 %

Mean Edge Similarity Measure

Correctly Classified in [%]	8×8 boxes	16×16 boxes	32×32 boxes	64×64 boxes
192×144	98.1 %	98.0 %	-	-
384×288	96.0 %	97.6 %	96.5 %	-
768×576	-	95.4 %	95.9 %	98.4 %

Sum of Edges Similarity Measure

Correctly Classified in [%]	8×8 boxes	16×16 boxes	32×32 boxes	64×64 boxes
192×144	98.8 %	98.7 %	-	-
384×288	97.5 %	96.0 %	95.6 %	-
768×576	-	95.8 %	96.4 %	95.3 %

Sum of Foreground Pixels Similarity Measure

Correctly Classified in [%]	8×8 boxes	16×16 boxes	32×32 boxes	64×64 boxes
192×144	98.1 %	98.5 %	-	-
384×288	96.7 %	98.5 %	98.0 %	-
768×576	-	97.4 %	97.6 %	97.9 %

Table 2.8: Classification results for the different similarity measures.

Final Classification Results

Correctly Classified in [%]	8×8 boxes	16×16 boxes	32×32 boxes	64×64 boxes
192×144	99.0 %	99.9 %	-	-
384×288	98.6 %	98.1 %	99.7 %	-
768×576	-	97.8 %	98.2 %	99.4 %

Table 2.9: Final classification results yielded by fuzzy classification framework.

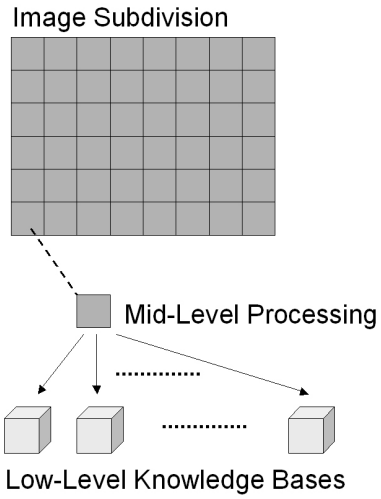


Figure 2.15: For each mid-level box, different low-level knowledge bases are administrated in order to cope with different background content.

2.2.4 Application of the Mid-Level Processing Stage

Within the segmentation process, the mid-level processing stage is used for two purposes. On the one hand, the current image content is compared to known background content enabling the recognition of foreground objects as large differences. On the other hand, foreground boxes are constantly observed as far as their dynamics and mobility is concerned. This is achieved by feeding a low-level knowledge base and a set of mid-level parameters with the content of the foreground box. In subsequent video frames, the pattern recognition is not only used to detect whether the box contains known background again, but also to measure the similarity of the formerly learned foreground box content with the current foreground content. A high similarity of learned foreground content and current content indicates that the object – as far as the respective box area is concerned – has not moved. This is an important feature for the tracking component, which is described in chapter 2.3.2, because it enables the recognition of permanent changes. If a box is classified as ”foreground” again and the similarity to formerly learned foreground content is low, the low-level knowledge base and the mid-level parameters are reset and newly initialized with the current content. Because of the continuous changes which are caused by a moving foreground object, it is not reasonable to maintain or store such short-dated information for longer time. If indeed a permanent change in the background is recognized by the subsequent processing stages, the low-level knowledge bases and mid-level parameters trained so far can simply be added to the set of background knowledge bases without any further training steps. This scheme is illustrated in figure 2.15.

The mid-level processing stage is able to classify the image content in order to find foreground objects in front of known background. Due to the automatic recognition of permanent changes, the amount of known background is extended by the system itself. Another major advantage of this approach is the considerable reduction of some of the typical problems of image processing. Especially, the influence of noise or aliasing effects of the cameras can be masked out nearly completely. Practical evaluations have shown that the mid-level processing stage is able to sufficiently cope with severely changing illumination conditions as well. For example the recognition remained stable, even if the light was switched on or off. This is

usually a difficult situation for segmentation algorithms based on background subtraction because the image colors change considerably in this case. An analysis and evaluation of the whole segmentation system is presented in chapter 2.5.

A further advantage of the mid-level processing stage is that its output is a reduced set of data. For example, we subdivided a $192 \times 144 = 27648$ pixels large image area into $12 \times 9 = 108$ box areas. Compared to the results of the low-level processing stage, the information, which has to be handled, is reduced to 0.3% of the original data set. However, the computational effort to calculate the five similarity measures and to perform the fuzzy classification process for several knowledge bases and each box area separately requires a state of the art processor as explained in chapter 1.3.

2.3 High-Level Processing Stage

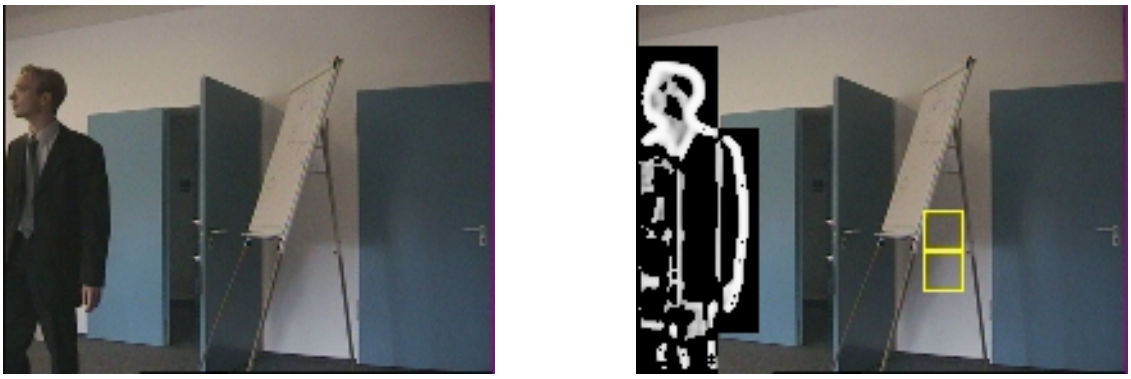
During the high-level processing stage, application-dependent knowledge is utilized to finally determine the objects of interest and to recognize possibly wrong classification results of the mid-level processing stage. The results of the mid-level processing stage – the boxes are classified either as foreground or as background – are employed to determine objects within the video images as connected components of foreground boxes. Based on given parameters as for instance the maximum number of expected foreground objects or their size, superfluous objects can be neglected. A typical example is illustrated in figure 2.16. In (a), the original video image is shown, in (b) the result of the mid-level processing stage. The person on the left is recognized as connected component of foreground boxes. For illustration purposes, the image data of the foreground boxes is replaced by the contour information which is provided by the low-level processing stage. Moreover, there are two wrongly classified foreground boxes in (b) on the right, indicated by yellow boxes. The high-level processing stage is able to utilize application-dependent knowledge, in this case for instance the expected size of objects, to identify the wrong classifications. Consequently, the results of the low-level segmentation are masked out in the wrongly classified box areas. Subsequent processing stages do not have to cope with these faults because they are recognized within the segmentation.

Moreover, the high-level processing stage contains a tracking component, which tries to track and follow objects over time. Based on this information, the visibility of objects is set and permanent background changes are recognized. Superfluous objects (as in the example) are observed and masked out as long as they do not fulfill the criteria.

2.3.1 Application-dependent Knowledge

Depending on the application, the aim of segmentation may be different. The high-level processing stage tries to benefit from these application-dependent aspects to identify the final objects of interest. Further objects which are possibly superfluous or may result from wrong classification results are masked out for subsequent processing steps.

An important aspect of the high-level processing stage is the fact that – after the pixelwise classification during the low-level processing stage and the box-wise classification during the mid-level processing stage – we are now classifying objects. As mentioned above, objects within the image are determined as connected components of foreground boxes. The following properties of these objects, together with application-dependent knowledge, are utilized in the high-level processing stage:



(a) Original image (b) Result of mid-level processing stage

Figure 2.16: Example of mid-level processing stage results. The person on the left is recognized as connected component of foreground boxes. For illustration purposes, the original image is replaced with the low-level contour segmentation in (b). The wrongly classified foreground boxes in (b) on the right (indicated by yellow boxes) can be recognized as ”wrong” by the high-level processing stage, for instance because of their size. As a consequence, the results of the low-level processing stage are masked out in these box regions.

- **Number of Foreground Objects**
  A first important aspect is the number of the expected foreground objects. If, for instance, just one foreground object is expected to be in front of the camera, further objects can be neglected. The treatment of superfluous objects is explained in section 2.3.2.2.

- **Size of Foreground Objects**
  Another feature of the expected foreground objects is their size. In the high-level processing stage, a minimum number of foreground boxes of which an object has at least to consist can be specified. Objects consisting of fewer boxes, which are consequently of a smaller size, can be neglected and are made invisible. Thus, they do not disturb further processing steps of subsequent applications. Moreover, a minimum object size can be specified for the detection of wrong classification results. Objects consisting of less boxes than this given size are considered not to be objects at all, but wrong classifications of the mid-level processing stage. As a result, such an object is not only made invisible, but is immediately learned as background. Additionally, it may be specified that this rule is not applied for objects located on the border of the image because in many applications objects of interest may enter the video image from one of the borders. In this case, it is not reasonable to learn such objects immediately as background, so this case can explicitly be excluded.

- **Shape of Foreground Objects**
  A complicated feature of objects is their shape. However, because of the fact that the objects are approximated using rectangular boxes the shape of the objects, which may be derived from the connected components of foreground boxes, is not meaningful at all. Nevertheless, depending on the expected foreground objects, the system can make use of the fact that objects appear as contiguous areas, which especially do not contain any holes. In that case, boxes possibly classified as ”background” can be recognized as ”foreground” nonetheless.

Using this application-dependent knowledge enables the high-level processing stage to extract the final aim of segmentation, if more than the expected number of objects is found. Furthermore, small detected objects, which may arise from faulty low-level or mid-level classification results, can be removed.

2.3.2 Tracking of Objects

While the operations we have described so far are applied to each video image separately, the tracking component of the high-level processing stage incorporates features which occur due to the time coherence of subsequent image frames. The tracking component also works on the mid-level results and thus has to deal with a small number of box classification results as well. The main task of the tracking component is the tracking of objects from frame to frame. Moreover, the decision which objects are visible and which are invisible is made by the tracking component. Additionally, the tracking of objects allows the observation of their behavior over time. This information may be utilized on the one hand to detect permanent changes in the background, for instance caused by an object moving in the background. On the other hand, the knowledge about the behavior over time allows the improvement of segmentation results. For instance, a box that belonged to an object for a significant time may have been classified wrongly, if it is suddenly recognized as ”background”, while the rest of the object did not move and is still ”foreground”. Thus, the tracking component is a reasonable supplementation of the segmentation approach introduced so far and improves the overall results by considering time coherent features.

2.3.2.1 Features of Objects

An important feature of the tracking component is the *dynamics* of objects, which is provided by the pattern recognition of the mid-level processing stage. As explained in chapter 2.2.4, foreground boxes are constantly learned into separate low-level knowledge bases and mid-level similarity measures. This mechanism allows the comparison of current with former foreground box content. A high similarity to formerly learned foreground content indicates little mobility and may be a hint for a permanent background change. The dynamics of an object i is determined as:

$$d_i = \frac{1}{s} \sum_{j=1}^{s} (1 - \mu_{j,fgr}) \tag{2.34}$$

where

s : size of object, counted in boxes
$\mu_{j,fgr}$: fuzzy similarity to learned foreground in j-th box of object.

Large membership degrees of d_i indicate a dynamically moving object, while low membership degrees indicate a more or less static object. The tracking component stores the last position of the object, i. e., the boxes that belong to the object. Moreover, for each object box the information is stored, how often the box has belonged to the object, calculated as running average over the last preceding images.

2.3.2.2 Determination of Visibility

In order to be visible for subsequent applications, objects have to consist of a predefined minimum number of boxes, which is an application-dependent parameter of the high-level processing stage. Smaller objects are simply masked out and made invisible. Moreover, the visibility of objects is limited by the specified maximum number of visible objects, which is another parameter of the high-level processing stage that is largely application-dependent. If more objects of sufficient size are detected than allowed, the high-level processing stage has to decide which of these objects are visible and which not with respect to the maximum number of visible objects. In this case, the likeliness of visibility is determined for each object by a weighted formula that takes into consideration the dynamics, the size, the age of the object and the number of frames since when the object has already been visible. The visibility v_i of object i is determined by:

$$v_i = \upsilon_1 \cdot d_i + \upsilon_2 \cdot s + \upsilon_3 \cdot a + \upsilon_4 \cdot f \tag{2.35}$$

where

d_i	:	dynamics of object (equation (2.34))
s	:	current size of object, counted in boxes
a	:	age of object (i.e. number of frames since first appearance)
f	:	number of frames since the object is visible
υ_j	:	weighting factor of aspects with $j = 1, ..., 4$.

The application of equation (2.35) provides an indicator for the choice which objects should be made visible and which not. Those objects i with the largest visibility indicators v_i are chosen as visible, whereas the rest – as far as there are more objects than the maximum allowed number – is made invisible. We experienced good results with the weighting factors $\delta = 0.2$, $\upsilon_1 = 0.2$, $\upsilon_2 = 0.25$, $\upsilon_3 = 0.15$ and $\upsilon_4 = 0.4$.

2.3.2.3 Detection of Permanent Background Changes

For the detection of permanent background changes, the dynamics of objects obviously plays an important role again. Static objects are likely to show no dynamics at all, especially when their static behavior can be observed for a longer time. The calculation of the dynamics of an object i at time t is explained in equation (2.34). In order to take into consideration the amount of time the object remains static, we measure the preceding dynamics d_i^t of object i at time t, which is averaged over time ($t > 0$):

$$d_i^t = \delta \cdot d_i + (1 - \delta) d_i^{t-1} \tag{2.36}$$

For $t = 0$, the dynamics d_i^0 is set to:

$$d_i^0 = d_i \tag{2.37}$$

where

δ	:	weighting factor of current and preceding dynamics
d_i	:	current dynamics of object i, see equation (2.34).

The decision, whether a detected object is a permanent background change or an object of interest, is made depending on the current dynamics d_i, the preceding dynamics d_i^t and the

number n of frames since when the object has remained static. Using the thresholds υ_{d_i}, $\upsilon_{d_i^t}$ and υ_n, a permanent change is recognized, if the following conditions are fulfilled:

$$d_i > \upsilon_{d_i} \quad \wedge \quad d_i^t > \upsilon_{d_i^t} \quad \wedge \quad n > \upsilon_n \tag{2.38}$$

Each time the current dynamics d_i of an object is below the specified threshold ($d_i < \upsilon_{d_i}$), the object most probably has moved and thus is not a permanent change in the background. In this case, the counter n is set to 0 again. Good results have been achieved with $\delta = 0.2$, $\upsilon_{d_i} = 0.7$ and $\upsilon_{d_i^t} = 0.7$. In order to be independent of the actual calculation time for one image frame, we choose υ_n with respect to the actual number of processed frames per second. Thus, υ_n corresponds to the number of seconds during which the object remains static. Depending on the subsequent application, ten to fifteen seconds should be a good choice for υ_n.

2.3.2.4 Improvement of Segmentation Results

The tracking component collects information about the position, the preceding positions, the dynamics and the preceding dynamics of objects. Besides the detection of permanent background changes, as explained in section 2.3.2.3, this information can also be utilized to observe the behavior of objects. For instance, if a rather static behavior of an object is detected, most probably the same boxes as in preceding images will belong to the object again. As a consequence, a box that is classified as ”background”, but has belonged to the object during the preceding images, is reconsidered. For this purpose, the median classification result of the mid-level processing stage of the neighboring boxes, to which the box under consideration is finally set, is determined. That means, if the box that is supposed to be classified wrongly is surrounded by ”foreground” boxes, it will be set to ”foreground” as well. Otherwise, the ”background” classification is kept. Due to this mechanism, possibly wrongly classified boxes are identified by incorporating knowledge about the behavior of objects and their time coherent features.

2.3.3 Results of High-Level Processing Stage

The high-level processing stage yields a set of results and directives for the subsequent feedback mechanisms (section 2.4). The mid-level processing stage distinguishes between the classes ”background” and ”foreground”, which are provided as input for the high-level processing stage. Depending on the results, which are obtained using the described methods, the classes of the boxes may be modified by the high-level stage.

In addition to the ”background” and ”foreground” classification, we introduce two new classes, namely ”border” and ”new background”. We classify boxes as ”border” that are neighboring to a ”foreground” box (considering only the 4-neighborhood relationship). The usage of a ”border” class is reasonable because small parts of a foreground object may stick out into neighboring boxes, which may have been classified as ”background” because the change within their area is not severe enough. The introduction of the ”border” class is partly a result of the empirical evaluation, which has shown that small differences in the image content may lead to a small decline in the change detection measures only (see section 2.2.3). The ”border” class allows the algorithm to perform a specific handling of these boxes possibly classified

wrongly. For example, the content of these ”border” boxes may not be considered for learning. If a permanent change in the background is detected, the high-level processing stage will classify these boxes as ”new background”.

Depending on the evaluation of the visibility calculation, the high-level processing stage provides information for each foreground box, whether it is visible, i. e., the low-level processing stage is advised to work out object pixels. Otherwise, it is invisible, which means that the box area is masked out eliminating possibly wrongly classified pixels of the low-level stage.

In order to track the dynamics of objects, which is achieved by learning the foreground content in separate knowledge bases, it is important to reset the low-level knowledge bases and mid-level parameters of the pattern recognition system, when the object moves. This directive is obtained during the evaluation, whether the object is a permanent background change as explained in section 2.3.2.3. Depending on this evaluation, the high-level processing stage decides to either reset or not to reset the foreground box content learned so far. Resetting these knowledge bases is not problematic because the observation of moving objects provides short-dated information, which is not reasonable to store for a longer time.

2.4 Feedback Mechanisms

Performing a classification for each box as described in section 2.2 yields an assignment of either ”background” or ”foreground” to each box as a result of the mid-level processing stage. However, the high-level processing stage provides an extended set of classification results and further directives that may be even contradictory to the mid-level results. In this section, we present methods that, on the one hand, solve possible conflicts of the results and, on the other hand, aim at improving the segmentation process for subsequent images.

2.4.1 Adaptive Learning

Basically, we have to distinguish between eight different combinations of the mid- and high-level processing stage. The results of the mid-level processing stage are either ”background” or ”foreground”. Additionally, the high-level assignments ”background”, ”foreground”, ”border” and ”new background” have to be considered as explained in section 2.3. The possible combinations are illustrated in figure 2.17. For each of the eight possible combinations (indicated by arrows in figure 2.17), a separate treatment of the respective box content is performed:

- **”Background” and ”Background”**
  Fortunately, both mid-level and high-level processing stage consider the box content to be background in this case. Consequently, the mid-level change detectors are adaptively updated with the current box content. The currently active low-level knowledge bases are advised to learn the color and edge information of the box (see chapter 2.1). If the low-level processing stage has determined foreground object pixels, these wrongly classified pixels would be removed.

- **”Background” and ”Foreground”**
  The mid-level processing stage wrongly classified the current box content as ”background”. A new instance of mid-level change detectors is generated and initially learned

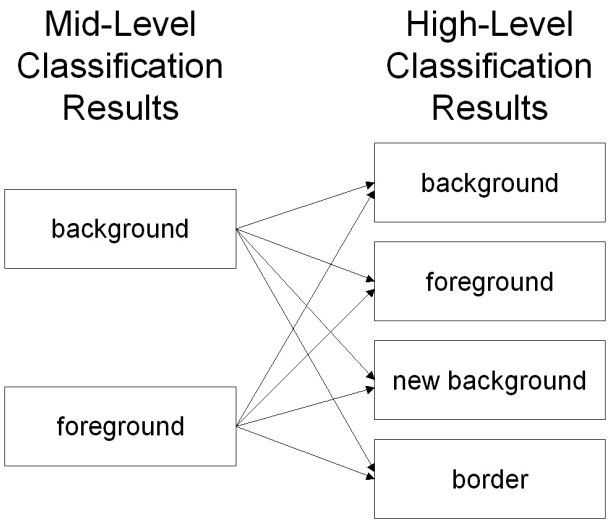


Figure 2.17: There are eight possible combinations of mid- and high-level classification results.

with the current box content. Furthermore, new low-level knowledge bases are created and trained on the current content. The parameters of the old mid-level detectors are stored together with the old low-level knowledge bases in order to be reactivated in the case that the old background situation occurs again.

Depending on the directive whether the box content is visible foreground or not, the low-level processing stage is advised to work out foreground contour pixels of the object.

- **”Background” and ”Border”**
  This combination of classification results is not necessarily a contradiction because the current box is only neighboring to a ”foreground” box. Small parts of a foreground object may or may not stick out into the box under consideration. As a consequence, the low-level processing stage is advised not to work out foreground object pixels and both the mid-level parameters and the low-level knowledge bases are not trained on the current box content in order to avoid learning of any wrong content.

- **”Foreground” and ”Foreground”**
  In this case, the high-level processing stage supports the classification of the mid-level processing stage. In the case that the current box has been ”background” so far, new instances of mid-level change detectors and low-level knowledge bases are created and initialized with the current box content.

  If the box has been ”foreground” before, the additional information provided by the high-level stage, whether the object is dynamic and moving or whether the object remains static, is considered. In the case that the object has remained static, the low-level knowledge bases and mid-level parameters, which have been instantiated for the foreground box content, are learned into the existing data base. Otherwise – the object has moved – the knowledge bases and parameters are reset and newly initialized. Then the current content is learned into the knowledge bases.

  Depending on the directive, whether the box content is visible foreground or not, the low-level processing stage is advised to work out foreground contour pixels of the object.

- **”Foreground” and ”Background”**
  The decision to classify the current box content as ”foreground” was wrong. Consequently, the detectors have been too sensitive to possible changes. In order to adapt the parameters and the knowledge bases, the current image data is learned more intensively.

- **”Foreground” and ”Border”**
  The ”border” classification indicates that the high-level processing stage has modified the original class ”foreground” to ”background”, which has become ”border” because of its spatial neighborhood to other foreground boxes. However, the usage of the ”border” class indicates that it is quite difficult to correctly classify the border regions between object and background. Thus, neither parameters are adapted nor any content is learned. The low-level processing stage is advised, not to work out any object pixels.

- **”Foreground” and ”New Background”**
  The high-level processing stage has decided that the object, to which the box under consideration belongs, is a permanent change in the background. Thus, the low-level knowledge base together with the trained mid-level parameter setting is added to the set of background knowledge bases. The low-level processing stage is advised, not to work out any object pixels. In subsequent images, the object should immediately be recognized as ”background”.

2.4.2 Adjustment of Parameters

During experimental tests, we have experienced a large dependency between existing background edges and potential segmentation failures. We have already taken this effect into consideration as far as the low-level edge segmentation is concerned (see section 2.1.2) by introducing the function $\eta(\vec{p})$ that reflects existing background edges in the surroundings of pixel $\vec{p}$. Because of the same effect, we parametrize the mid-level similarity measures differently with respect to the edginess of the respective box area (see section 2.2.3). For the same reason, we adjust the filter parameter υ_e of equation (2.13) in section 2.1.2 to the edginess of the respective box area. Basically, we distinguish two different cases, either the background box area is smooth and does not contain any edge at all, then a low threshold υ_e is chosen, or the area contains background edges, then a higher threshold is applied.

Considering the different combinations of mid-level and high-level results, we are able to deduce some information about the mid-level classification results. As explained in section 2.2.2, we use five different similarity measures, which are combined to a final result using a fuzzy logic rule base. Depending on the respective case, one or more of the detectors may have provided a poor classification result. We use the high-level results in order to reconsider the reliability indicators of the similarity measures. The indicators of those detectors, which have provided a wrong classification result, are decreased, while the others are increased. For subsequent classifications, this method helps improving the classification results due to the inclusion of the reliability indicators within the fuzzy rule base.

When a detector provides poor classification results over a long period of time, its parameters are presumably not chosen sufficiently well. Each of the presented fuzzy membership functions in section 2.2.1 is influenced by a controlling factor δ_i (with $i = 1, ..5$). Principally, the influence of the parameters is identical for each membership function. A large value of δ_i tends to result in lower membership degrees of the respective membership function, while

a small value of δ_i tends to result in larger membership degrees. In section 2.2.3, we have manually determined δ_i as a result of empirical evaluations. While the determined parameters work fine for the empirical testing, it is unlikely that they are universally valid. Thus, a more general approach for the determination of the controlling factors is appropriate. As a part of the feedback mechanisms, we employ the reliability indicators, which are utilized and maintained for the determination of an overall result of the mid-level processing stage (see section 2.2.2.1). The indicators $\mu_{i,bgr}$ and $\mu_{i,fgr}$ reflect the classification quality of each similarity measure, both for low membership degrees ($\mu_{i,fgr}$) and for high membership degrees ($\mu_{i,bgr}$). For the adaptive manipulation of the controlling factors, we consider the reliability indicators and possible disproportions between them. If for one and the same similarity measure the reliability indicator $\mu_{i,bgr}$ for high membership degrees is rather low and $\mu_{i,fgr}$ for low membership degrees is rather high, the controlling factor is obviously not chosen sufficiently well. In this exemplary case, the classification quality can be improved by increasing δ_i because a low reliability indicator $\mu_{i,bgr}$ indicates that the box content has often been classified as "background", although it has been "foreground". Increasing δ_i should adjust this imbalance by leading to lower membership degrees of the similarity measure.

Going into more detail, we continuously perform the adaption of parameters after a predefined amount of frames. We compare the reliability indicators $\mu_{i,fgr}$ and $\mu_{i,bgr}$ for each similarity measure i. If $|\mu_{i,fgr} - \mu_{i,bgr}| > \theta$ holds, the controlling factor δ_i is manipulated considering two cases. If $\mu_{i,fgr} < \mu_{i,bgr}$, then

$$\delta_i = [1 - (\mu_{i,bgr} - \mu_{i,fgr}) \cdot \iota] \cdot \delta_i \tag{2.39}$$

Vice versa, if $\mu_{i,fgr} > \mu_{i,bgr}$, then

$$\delta_i = [1 + (\mu_{i,fgr} - \mu_{i,bgr}) \cdot \iota] \cdot \delta_i \tag{2.40}$$

where $\theta \in [0, 1]$ is a threshold specifying the maximally allowed difference between $\mu_{i,fgr}$ and $\mu_{i,bgr}$ before the controlling factors δ_i are manipulated, and $\iota \in \mathbb{R}$ is a factor that regulates the impact which indicates how much the controlling factors are increased or decreased, respectively.

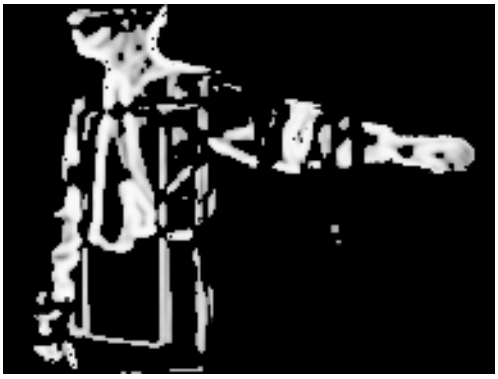
2.5 Evaluation of the Segmentation Process

In this chapter, we have introduced a multi-level segmentation approach with internal feedback mechanisms so far. We have described a lot of ideas and methods to improve the segmentation process. In this section, an objective evaluation is taken in order to underline the performance of the system.

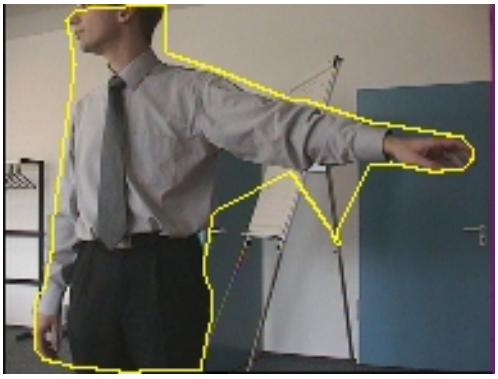
2.5.1 Necessity of the Multi-Level Approach

Principally, the low-level processing stage offers a segmentation approach that is able to perform the determination of foreground object pixels as a kind of background subtraction approach. Thus, a legitimate question is, why we still need a multi-level segmentation approach? On the one hand, there are some qualitative capabilities that are related to the possibilities of the utilized pattern recognition. Due to the mid-level and high-level recognition capabilities, the system is able to detect permanent changes in the background. Moreover, different

(a) Original video image

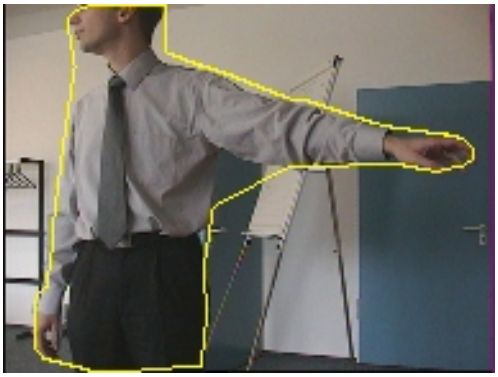


(b) Low-level segmentation only



(c) Resulting faulty contour approximation

(d) Multi-level segmentation



(e) Resulting correct contour approximation

Figure 2.18: Typical example for improvements due to multi-level approach.

low-level knowledge bases can be administrated and background knowledge can be stored and recalled for several situations. This is a qualitative advantage over common background subtraction approaches. On the other hand, there are also quantitative improvements of the segmentation process as far as the final segmentation results are concerned. Due to the autonomous determination of foreground areas by the mid- and high-level processing stage, the low-level segmentation is only applied to image regions, where foreground objects are actually located. Thus, recognized background regions are directly masked out for the low-level segmentation making failures on the pixel level in these regions impossible.

A typical example for the improvements related to the multi-level approach is shown in

figure 2.18. The failures of the low-level segmentation in figure 2.18(b) are avoided due to the multi-level approach in (d). The image was taken out of an application that enables interaction with back-projection walls using arm gestures. This system is explained in detail in chapter 4. The application of the contour approximation algorithm (see chapter 3) produces an outlier in figure 2.18(c) because of the wrongly classified pixels below the arm of the user. The subsequent application which determines the pointing direction of the arm may be negatively affected by the outlier. This problem is avoided due to the better results of the multi-level segmentation in figure 2.18(d) and (e).

2.5.2 Comparison to Common Background Subtraction

A common background subtraction approach works on gray-level images [BAL82]. During an initial learning phase, the static background has to be presented to the system. The simplest background model assumes that the gray-level values of each pixel vary independently, according to normal distribution. Similar to our approach, the mean gray-level value $g(\vec{p})$ at each pixel position $\vec{p}$ is calculated during the learning phase:

$$m(\vec{p}) = \frac{1}{N_S} \sum_{t=1}^{N_S} g_t(\vec{p}) \tag{2.41}$$

where $N_S > 0$ is the number of sample images and $g_t(\vec{p}) \in [0, 1]$ is the gray-level value in the background image at time t at pixel $\vec{p}$. Moreover, the standard deviation at each pixel position is determined:

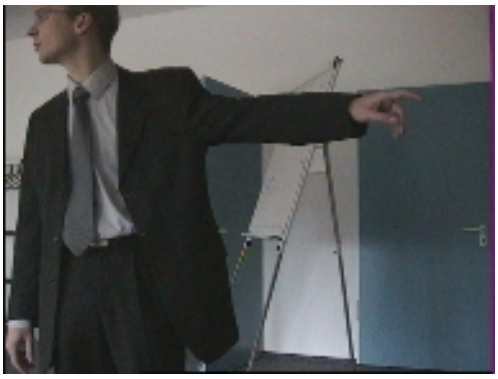
$$\sigma(\vec{p}) = \sqrt{\frac{1}{N_S} \sum_{t=1}^{N_S} \left(g_t(\vec{p}) - m(\vec{p})\right)^2} \tag{2.42}$$

During the application phase, the segmentation is performed by considering the gray-level values of the pixels, the learned mean values, and the determined standard deviations:

$$f(\vec{p}) = \begin{cases} 0 & : \quad |m(\vec{p}) - g(\vec{p})| \leq \upsilon_s \cdot \sigma(\vec{p}) \\ 1 & : \quad |m(\vec{p}) - g(\vec{p})| > \upsilon_s \cdot \sigma(\vec{p}) \end{cases} \tag{2.43}$$

where $\upsilon_s \in \mathbb{R},\ \upsilon_s > 0$ is a threshold parameter. The threshold function f classifies the current input pixels into ”foreground” and ”background” depending on their similarity to the learned background images. Often, background subtraction approaches are combined with adaptive learning mechanisms (e. .g. [WRE95, KIL94]). For the described approach, the gray-level values are updated using the running average for the recognized background areas. We have implemented such a common background subtraction approach using the Open Computer Vision Library [OCV02] in order to compare its performance and results to our system. In order to remove noise, we additionally apply an averaging threshold filter that is used for our segmentation approach as well (see equation (2.4) in section 2.1.1). Moreover, we realized an adaptive updating of recognized background regions by calculating an enlarged convex hull polygon for the detected foreground object pixels. Outside this convex hull, the mean gray-level values are updated using the running average.

First of all, we experienced some difficulties to specify a suitable threshold parameter υ_s that, on the one hand, yields a sufficient amount of foreground object pixels and, on the other hand, suppresses as much noise as possible. While the common background subtraction

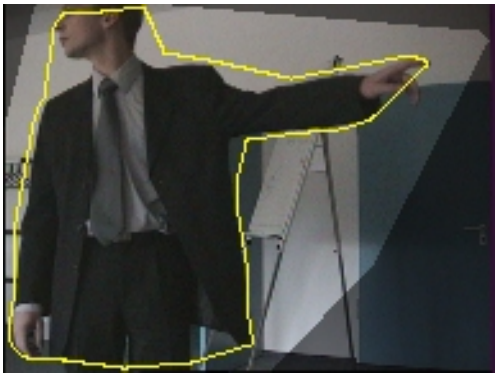


(a) Original video image

(b) Common background subtration

(c) Resulting faulty contour approximation

(d) Our low-level segmentation approach



(e) Resulting contour approximation

Figure 2.19: Typical example for better performance in comparison to common background subtraction.

approach worked sufficiently well for some scenarios, it nevertheless failed almost completely in other scenarios as for instance illustrated in figure 2.19. The scenario is difficult because the background colors change significantly, when the person enters the scene. These changes are most probably related to the slight shadow the user is casting. Moreover, the contrast of the images is quite low, so some parts of the user's dark clothing appear to be identical to parts of the background regions. As can be seen in figure 2.19(b), the threshold cannot be increased because some parts of the user are already masked out. However, there are large parts of the background regions that are segmented as foreground on the lower right of the image. While our low-level approach (figure 2.19(d) and (e)) does not work perfect all the time –

Figure 2.20: Typical image of evaluated series.

some frames contain failures as for instance shown in figure 2.18(b) – it nevertheless performs considerably better than the common background subtraction approach. The application of the multi-level segmentation improves the overall performance of our introduced approach as shown in section 2.5.1.

2.5.3 Objective Classification Quality

In section 2.2.3, we have empirically determined a suitable parameter setting for the mid-level box classification. The classification process and the employed feature extraction methods are working well, if they are efficiently parametrized. However, the linear interpolation of the parameter domain as proposed in section 2.2.3 has not been evaluated yet. In order to test the introduced classification process, we took a series of images, which has been recorded in front of a back-projection wall. Using the segmentation system for the figure/ground separation, a system was realized that enables the interaction of users with a back-projection wall using arm gestures. This system is explained in more detail in chapter 4. The sequence consists of 1162 images showing a user who points at the wall with his arm. On the application level, the mouse cursor on the back-projection wall is set to the position the user is aiming at. During the initial learning phase, 80 images showing the mere background were employed for the self-learning process. While recording the series, the electric light was switched off leading to a severe change in the illumination conditions. The images have a size of 192×144 pixels and are subdivided in 12×9 box areas. An exemplary image is shown in figure 2.20. In order to have an objective measure, we classified $2/3$ of the box areas of all images manually. Four columns on the right of the image area have not been classified manually because of the mirroring effect of the back-projection wall as can be seen in figure 2.20 on the right. On the one hand, the mirrored arm is something we do not intend to find as far as the application level is concerned. On the other hand, its appearance is a severe change in the learned background that should be detected by our method. For the evaluation of the classification process, it is not important at all. Thus, we concentrate on the $2/3$ of the image area on the left.

We applied the mid-level pattern recognition system to the test series and compared the results of the pattern recognition to our manually determined results. We counted as ”wrong”, if a foreground box was labeled as ”background” and vice versa. Based on this information, we examined the impact of the continuous adaptiveness of low-level knowledge bases and mid-level parameters and of the reliability indicators of the fuzzy rule base classifications. First

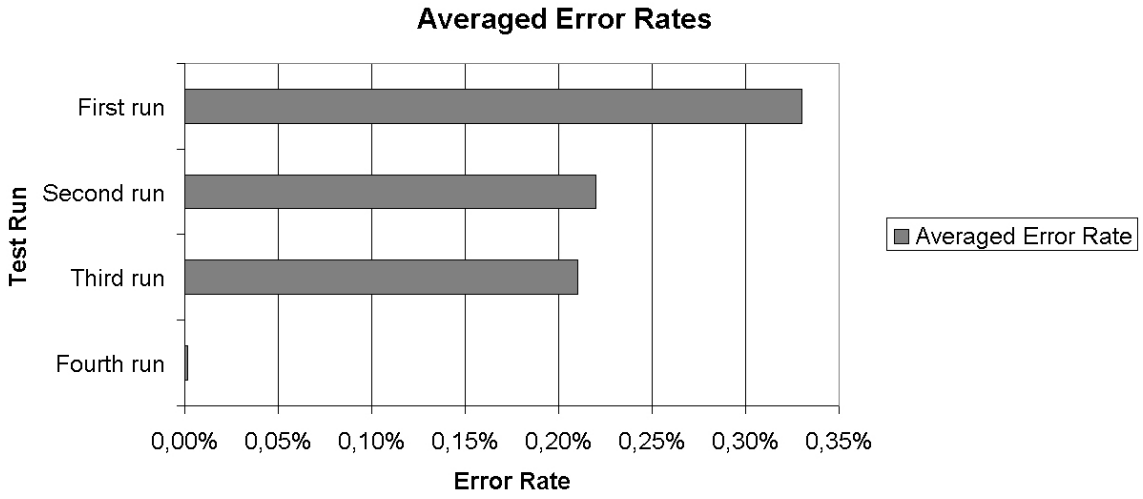


Figure 2.21: Classification error rates in dependency of improving methods. First run without adaptive updates and without influencing reliability indicators, second run with adaptive updates, third run with adaptive updates and reliability indicators, fourth run after applying high-level processing stage.

of all, we ran the test series with neither adaptive learning nor influencing of the reliability indicators. The average of the error rates for all the box areas under consideration was 0.33%. (see figure 2.21). Depending on the box area, the error rate of the mid-level classification results was between 0.0% and 3.49% in the respective box area. For the second run of the test series, we activated the adaptive learning of identified background content (see section 2.1.4) and the adaptive updating of the mid-level parameters (see section 2.2.1). Because of the definition of the *sum of foreground edges* and *sum of foreground pixels* similarity measures – these measures take directly into consideration the low-level results - these steps are not analyzed separately. The overall error rate averaged out at 0.22% (see figure 2.21). Compared to the first run, this is a decrement of wrong classifications of 32.3%. Depending on the box area, the error rate was between 0.0% and 3.23% for respective box areas. For the third run, we activated both the adaptive learning and the influencing of the reliability indicators for the classification process (see section 2.2.2). While the maximum error rate for the different box areas did not improve – still between 0.0% and 3.23% – the overall error rate slightly decreased to 0.21% (see figure 2.21). Compared to the second run, this is an improvement of further 8.0% and to the first run of 37.6%.

While the comparisons so far have taken into consideration the mid-level classification results, we finally analyze the high-level results as well. Within the high-level processing stage, application-dependent knowledge such as size or shape is utilized to find wrong classification results of the mid-level stage. As we have described in section 2.3, the high-level processing stage has an extended set of classification results. For instance, a ”border” class is introduced that indicates the neighborhood of the box under consideration to a foreground object. For the evaluation of the high-level processing stage, we have counted an error, if a ”background” box was classified as ”visible foreground” and if a ”foreground” box was classified as ”background”. Thus, a ”border” classification was not counted as wrong. Following this scheme, there was exactly one wrong classification result in the whole series, averaging out an error rate of 0.0013% of all classification results (see figure 2.21). This result indicates that especially border boxes are inclined to be wrongly classified. Consequently, it seems to be reasonable

(a) (b) (c)

Figure 2.22: Wrongly classified pixels are presumably close to the identified main object.

to treat ”border” boxes separately, for example by learning their content neither as foreground nor as background.

However, interpreting these promising results does not mean that the segmentation results are perfect. Merely the mid-level and high-level classifications are correct, or at least more or less correct as far as the ”border” boxes are concerned. Nevertheless, the pixelwise segmentation performed in the low-level processing stage may work out wrong foreground pixels within the ”visible foreground” boxes. An example is illustrated in figure 2.22. The box area below the arm of the user is principally classified correctly as ”visible foreground”, but some pixels of this box are classified wrongly as foreground pixels (figure 2.22(b)). This may lead to an inaccurate contour approximation (chapter 3), shown superimposed in yellow in figure 2.22(c). The fact that wrong pixels can mainly be found close to the objects of interest may be considered as an advantage.

Summarizing the results of this objective comparison allows the conclusion that 99.79% of the classification results of our pattern recognition system have been identical to the human classifications. Moreover, the introduced mechanisms which realize a continuous adaptiveness to current circumstances lead to significant improvements of the classification results. Moreover, we have analyzed a sequence of 1162 images only. Considering the fact that most of the improving methods are conceived for a long time of application, the benefits of these methods may be even more significant than it has become obvious in our relatively short test sequence.

2.5.4 Analysis of Knowledge Bases

During the mid-level processing stage, several low-level color and edge knowledge bases may be instantiated for a box region, if a permanent change in the background is detected. The pattern recognition system of the mid-level stage administrates these knowledge bases and recognizes known background situations by choosing appropriate knowledge bases. However, the color knowledge bases are able to store detailed histogram information for each pixel of a box region separately in order to cope with slight changes and varying illumination. Thus, we analyze the necessity to have several knowledge bases by exploring the content of the color and edge knowledge bases. As a matter of fact, our multi-level segmentation system requires a significant amount of memory for the knowledge bases (see chapter 1.3). Thus, it is difficult to examine the color knowledge bases for every pixel in detail because an image consisting of 192×144 pixels already requires at least 27648 color knowledge bases. Moreover, if a permanent change in the background is detected, the instantiation of new knowledge bases for

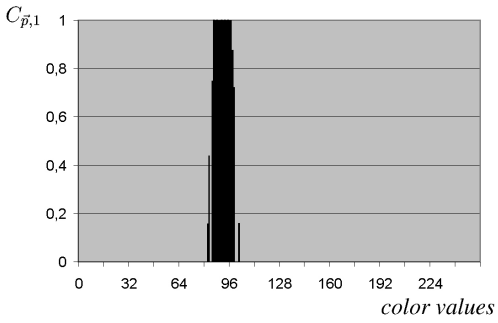
Figure 2.23: A typical image of the parking lot.

one 16×16 pixel large box area will require additional 256 color knowledge bases. Therefore, it is nearly impossible to examine the necessity of several knowledge bases in detail. As a consequence, we illustrate the impact of the mid-level processing stage for an example that was taken from the parking lot surveillance application (see chapter 4.4). The parking lot surveillance is an outdoor application that has to cope with the typical illumination changes which occur, e. g., because of moving clouds. A typical image of the parking lot is shown in figure 2.23.

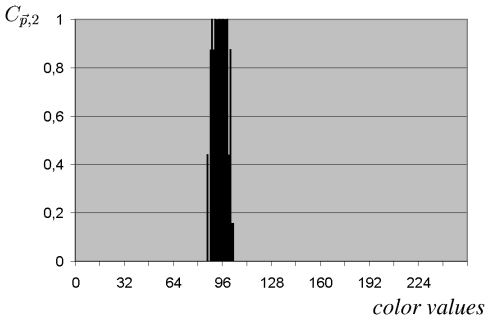
The application was executed for approximately 16 hours for this analysis. During this time, 811001 images were processed. The size of the box regions is 16×16 pixels and the RGB color space is used. On average, 5.8 knowledge bases were instantiated for each box region, the maximum was 21 knowledge bases. The decision of the mid-level processing stage to instantiate new low-level knowledge bases depends both on the results of the color and of the edge segmentation. Because of this reason, we illustrate the average edge values of the edge knowledge base for a typical box region, for which the average of six knowledge bases was instantiated. In figures 2.24(a) – 2.29(a), an enlarged clipping of the different edge knowledge bases is illustrated for the 16×16 pixels large box area under consideration. The content of the edge knowledge base has been shown for a complete image in figures 2.7(b) and 2.8(b) on page 28 and 29. Moreover, the color knowledge bases for an exemplary pixel of the box region are shown as a typical example in (b), (c) and (d) of the figures 2.24–2.29.

As can be seen in the figures 2.24 – 2.29, the average edge values of the knowledge bases are more or less significantly different. Only the fourth and the fifth edge knowledge base seem to be quite similar in figure 2.27 and 2.28, but the size of the disc in figure 2.27 is a bit larger than in figure 2.28. The visualization of the color knowledge bases illustrates the fuzzy membership degrees of the color values of the R, G, and B-channel to the linguistic term ”is background” for the exemplary pixel position (see chapter 2.1.1). As can be seen, the intervals of high membership degrees, which denote background colors, are small. Thus, foreground object colors just have to differ slightly in order to be recognized by the segmentation algorithm. Considering the first three color knowledge bases, the necessity of using different color knowledge bases becomes obvious because the intervals of high membership degrees are completely different. Summarizing this content in just one knowledge base would lead to severe disadvantages. On the one hand, a large interval of high membership degrees limits the recognition of foreground colors because colors within the interval will be recognized as background and are consequently not recognized correctly, if they occur within a foreground object. On the other hand, color knowledge may disappear because of the aging process (see

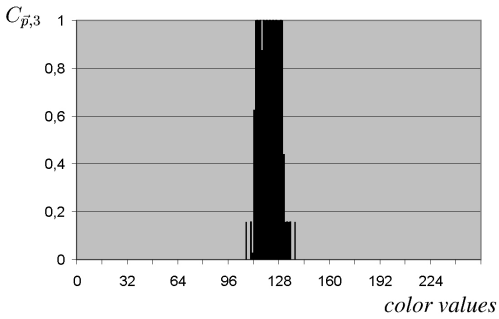
(a) Average edge values.



(b) R-channel



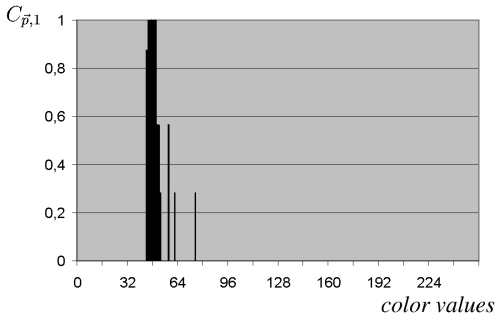
(c) G-channel



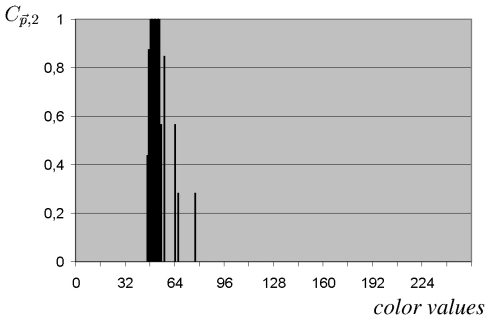
(d) B-channel

Figure 2.24: First edge knowledge base in (a) and corresponding color knowledge base for one pixel as an example in (b), (c) and (d).

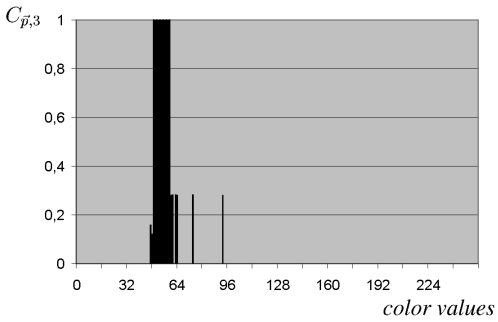
(a) Average edge values.



(b) R-channel

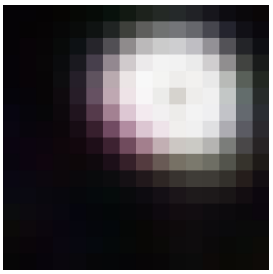


(c) G-channel

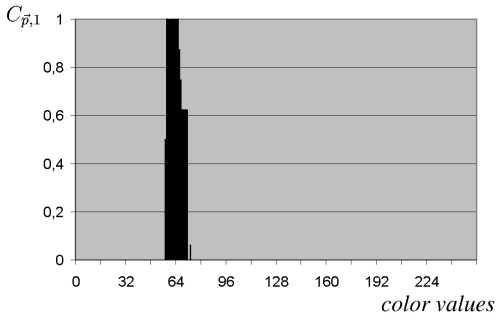


(d) B-channel

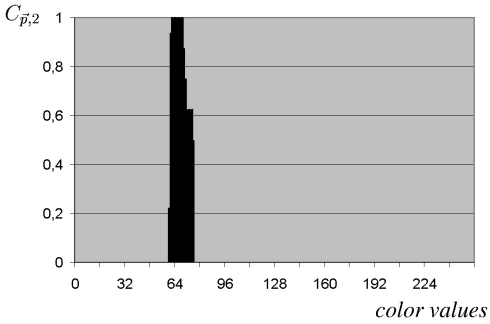
Figure 2.25: Second edge knowledge base in (a) and corresponding color knowledge base for one pixel as an example in (b), (c) and (d).



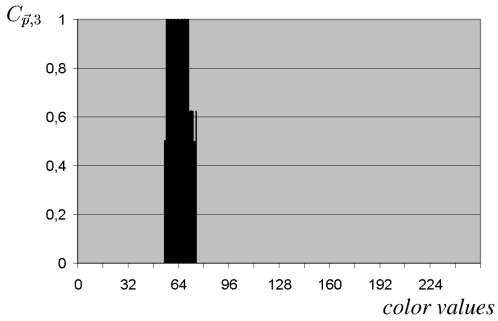
(a) Average edge values.



(b) R-channel

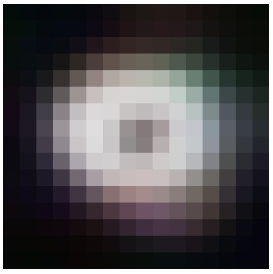


(c) G-channel

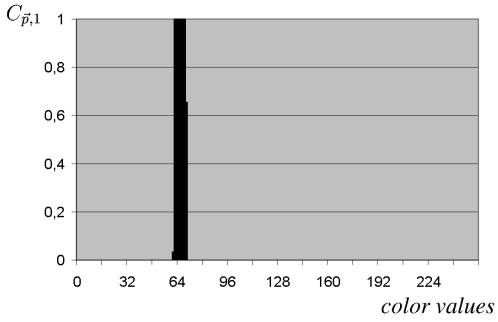


(d) B-channel

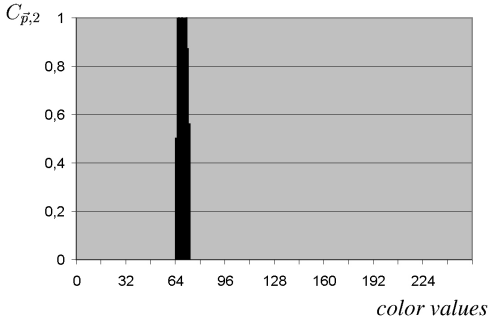
Figure 2.26: Third edge knowledge base in (a) and corresponding color knowledge base for one pixel as an example in (b), (c) and (d).



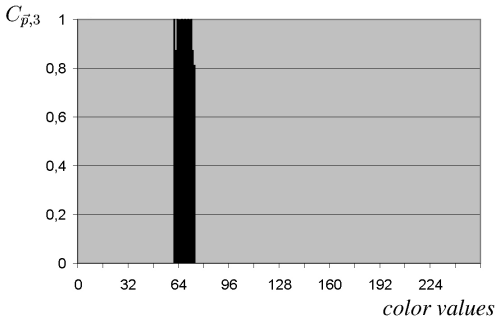
(a) Average edge values.



(b) R-channel

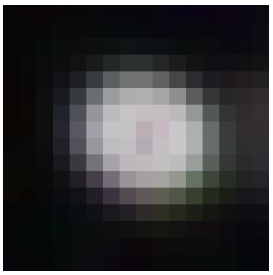


(c) G-channel

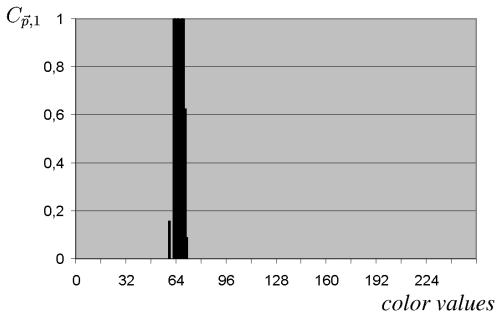


(d) B-channel

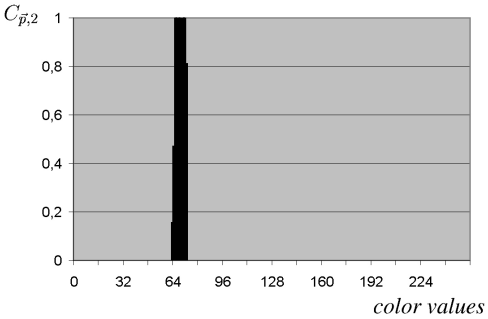
Figure 2.27: Fourth edge knowledge base in (a) and corresponding color knowledge base for one pixel as an example in (b), (c) and (d).



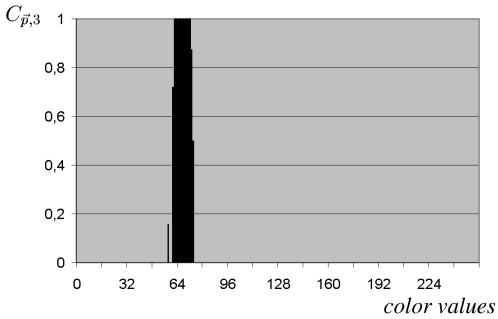
(a) Average edge values.



(b) R-channel



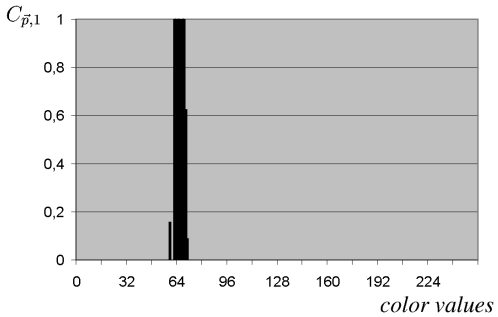
(c) G-channel



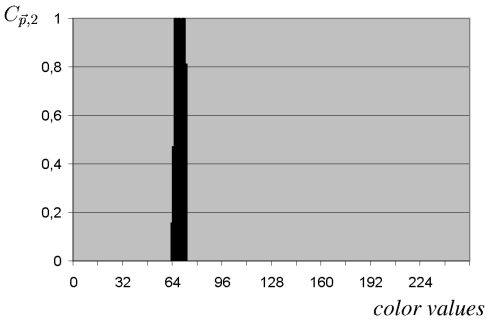
(d) B-channel

Figure 2.28: Fifth edge knowledge base in (a) and corresponding color knowledge base for one pixel as an example in (b), (c) and (d).

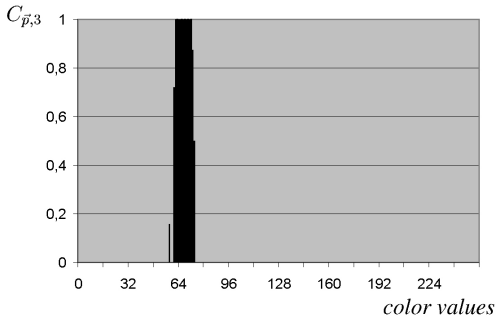
(a) Average edge values.



(b) R-channel



(c) G-channel



(d) B-channel

Figure 2.29: Sixth edge knowledge base in (a) and corresponding color knowledge base for one pixel as an example in (b), (c) and (d).

chapter 2.1.1) after a permanent change in the background. As a consequence, the learned color knowledge would be lost and unavailable, if a known situation reoccurs. Due to our approach to have several knowledge bases in combination with a pattern recognition system, the system is able to store and to administrate completely different background content and to recall this knowledge, if former situations reoccur. The last three color knowledge bases show similar intervals for all the color channels. However, as already explained, this is just the visualization for one out of 256 pixels of the box region. As indicated by the average edge values in figure 2.27, 2.28 and 2.29, the content of the background has more or less significantly changed, although the color values of the exemplary pixel under consideration may not be affected.

Although a detailed analysis of the thousands of color knowledge bases is not possible, the example illustrates the necessity and the advantages of the approach. Both color and edge knowledge bases show significant differences, underlining the importance of using several knowledge bases.

2.5.5 Evaluation on Application-Level

In section 2.5.2, we have shown that our combination of color- and edge-based segmentation, referred to as low-level segmentation, performs better than the common background subtraction approach that uses a dynamic reference image with mean gray-level values and the standard deviation. Moreover, we have demonstrated in section 2.5.3 that the results of the mid-level processing stage are almost identical to human classifications, averaging out at 99.79% identical classifications. In section 2.5.4, we have demonstrated the necessity and benefits of having several knowledge bases in order to administrate completely different background content.

Unfortunately, it is not possible to compare the performance of our approach to other state of the art methods we have surveyed in chapter 1.2 because none of them is available as open source. Nevertheless, in order to show the capabilities of our approach, we analyze the performance of our system in more detail on the application level in chapter 4 on the basis of very different applications, comprising for instance both indoor and outdoor environments.

2.6 Performance Improvements due to Parallelization

Experiments have shown that the presented segmentation system requires remarkable computational power (see chapter 1.3) in order to achieve a frame rate of 15-20 frames per second (fps), which may be considered as realtime. On a Pentium-III 1100 MHz system, the frame rate of our system is approximately 7-9 fps. However, dual processor systems have become more and more popular and moreover affordable in recent years. Due to the subdivison of the image area into boxes within the low- and mid-level processing stage, the system is highly suitable for parallel computing. Thus, we have implemented the system as multithreading solution that is able to principally distribute the computational effort on n threads, which may consequently run on n processors. However, only the low- and mid-level processing stage are suitable for parallelization. The final evaluation of the obtained results within the high-level processing stage has to be performed singlethreaded before learning in the mid- and low-level

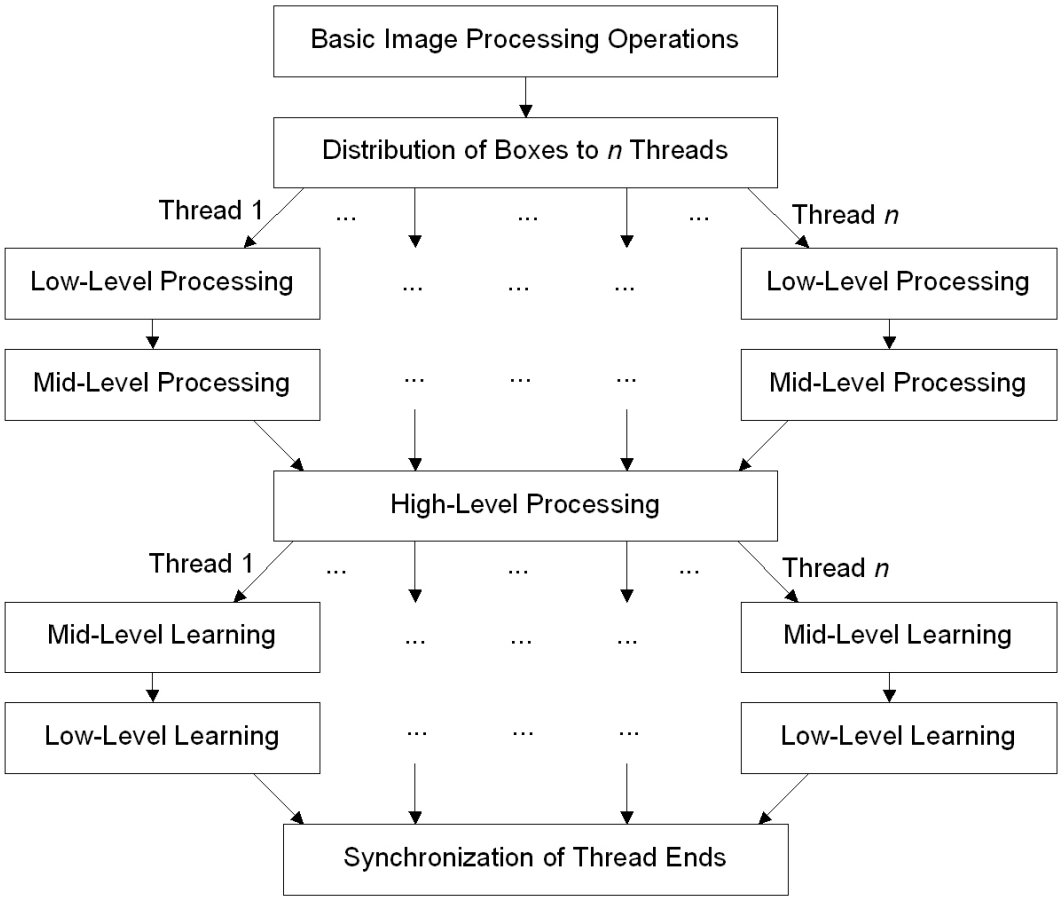


Figure 2.30: Multithreading structure for multiprocessor systems.

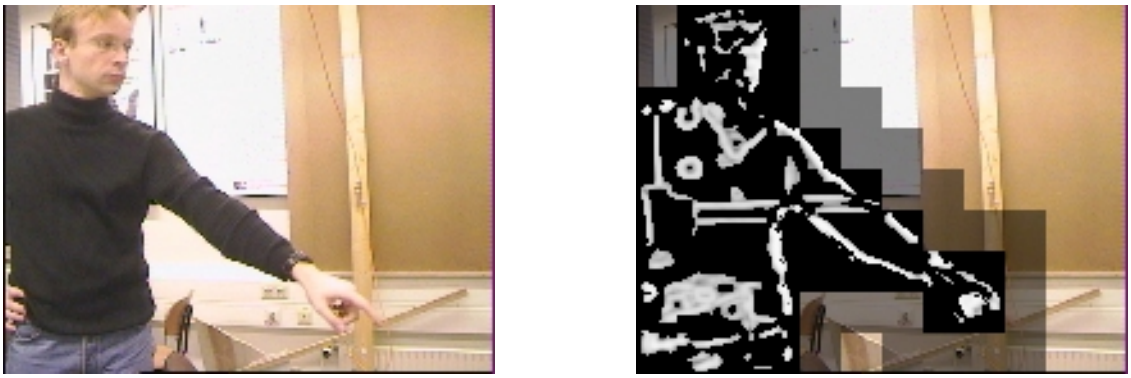
stage can be executed multithreaded again. This structure is illustrated in figure 2.30. Compared to the singlethreaded execution of the segmentation system, the frame rate improved to approximately 12-14 fps on a Dual Pentium-III 1100 MHz system.

2.7 Summary of Segmentation Approach

We have presented a three-level architecture for segmentation based on background subtraction. The low-level processing stage works pixelwise and determines contour pixels of foreground objects. The mid-level processing stage subdivides the image area in small rectangular box-shaped regions. For each box region, a pattern recognition system is trained that allows the administration and recognition of different background content. The features of the pattern recognition system are designed to be insensitive against varying illumination conditions. The mid-level processing stage determines foreground regions, in which contour pixels of the objects are worked out, and background regions, in which possibly wrongly classified pixels of the low-level processing stage are masked out. Moreover, additional similarity measures can be incorporated easily into the classification process due to the fuzzy rule-based classification process. The high-level processing stage builds objects as connected components of foreground boxes. Based on the objects found, application-dependent knowledge is applied in

Low-level	*Equation*	*Impact*
σ	(2.2)	learning in color knowledge base
υ_c	(2.4)	filter parameter for color segmented results
ρ	(2.5)	aging of color knowledge
α, β	(2.11)	classification of edge values
υ_e	(2.13)	filter parameter for edge segmented results
υ_a	(2.15)	filter parameter for combined results
γ	(2.16)	adaptive learning of edge knowledge
Mid-level	*Equation*	*Impact*
δ_1	(2.21)	edge-based mutual similarity measure
δ_2	(2.27)	mean color similarity measure
δ_3	(2.28)	mean edge similarity measure
δ_4	(2.30)	sum of foreground edges measure
δ_5	(2.31)	sum of foreground pixels measure
High-level	*Equation*	*Impact*
$\upsilon_1, \upsilon_2, \upsilon_3, \upsilon_4$	(2.35)	determination of visibility
δ	(2.36)	dynamics of objects over time
$\upsilon_{d_i}, \upsilon_{d_i^t}, \upsilon_n$	(2.38)	detection of permanent background change
Feedback	*Equation*	*Impact*
θ, ι	(2.39), (2.40)	adjustment of parameters

Figure 2.31: Parameters of the multi-level segmentation approach.



(a) Video image (b) Result of segmentation

Figure 2.32: Exemplary result of application.

order to remove objects which do not fit the criteria of a subsequent application. Moreover, the high-level processing stage decides which objects are visible or invisible and whether a region previously classified as object is a permanent change in the background. Finally, internal feedback mechanisms are employed to continuously adapt to slight changes in the background and to adjust parameters.

During the definition of the multi-level segmentation system, we have introduced several parameters, which are summarized in tabular 2.31. Besides these parameters, application-dependent knowledge such as the size of objects or their expected number (see chapter 2.3.1)

may be incorporated.

In figure 2.32, an example of application of the multi-level segmentation system is illustrated. The original video image is shown in figure 2.32(a), the result in figure 2.32(b). The recognized object, in this case the person, is found as connected component of foreground boxes. For illustration purposes, the original image content of the foreground boxes is replaced by the result of the low-level processing stage. The ”border” boxes are indicated by a darkened box area. Recognized background regions are not changed in figure 2.32(b). Further processing steps, as for example the polygonal contour approximation that is explained in chapter 3, can be applied based on this result.

Chapter 3

Polygonal Contour Approximation

The contour is considered to be one of the most important features of an object. There are a lot of techniques in applied machine vision systems, which rely on contour information, such as object localization, recognition, and classification, to name just a few. A common approach to the calculation of contour pixels of an object in a digital image is spatial edge detection. A variety of edge detection operators such as the *Sobel* or the *Prewitt* operator can be found in [GON92]. Unfortunately, these methods are normally affected by discontinuities within the edge pixels. Thus, an edge linking algorithm has to be applied as a second step in order to obtain a closed contour in, e. g., the sense of 8-neighborhood of pixels. Another important aspect in this context is the detection of connected components. The algorithm has to decide, which pixels belong to one object and which pixels belong to different objects. The calculation of closed boundaries of objects is important for the determination of regions and can also be employed for pattern recognition purposes [SON98].

This chapter focuses on this second processing stage of an already preprocessed video image. We assume that the first step has already performed a segmentation, which has found pixels of the object contour. This is for example the result of the multi-level segmentation system presented in chapter 2. We propose a new method for the derivation of a complete polygonal contour from this information. A very simple solution for this purpose is the calculation of the convex hull polygon of the given set of object pixels. The convex hull polygon encloses the object pixels and is able to close even large gaps within the boundary of objects. However, as a severe disadvantage of the convex hull polygon, concave parts of objects are not approximated sufficiently. We propose a rather simple, but intriguingly powerful approach to contour approximation, based on the calculation of convex hull polygons for subdivided parts of the object pixels. The convex hull polygons and their intersection points are merged into a graph-based structure, which is employed to derive approximated boundaries of the sets of input pixels. Due to this approach, even concave parts of objects are approximated with adjustable accuracy in realtime. Moreover, the algorithm does not need any prior knowledge about the structure of an object in order to work properly. The determination of the number of objects within the image as well as the grouping of pixels is performed automatically, based on the assumption of the maximum length of contour gaps. The relationship between the choice of parameters and the ability of the algorithm to close gaps of respective lengths is analyzed theoretically. The application of the proposed method is independent of the presented multi-level segmentation system and works with other segmentation approaches due to an abstraction of input data as well.

In section 3.1, we briefly survey the state of the art as far as contour approximation methods are concerned. The scenarios of application for our algorithm are explained in more detail in

section 3.2. In section 3.3, we define a common starting basis in order to make our approach independent from specific preprocessing operations. The detection and grouping of different objects is explained in section 3.4. Algorithms for the determination of the approximated hull polygon are given in section 3.5. In section 3.6, we report on problems which result from the discretization of pixel positions and propose methods to solve them. In section 3.7, we analyze mathematically the influence of the parameters of our algorithm on the ability to close gaps in the contour and to distinguish between different objects within the image. For the integration of time coherence, an additional algorithm is introduced in section 3.8. In section 3.9, we present some examples of application of our contour approximation algorithm. Finally, we compare our results to other state of the art algorithms in section 3.10.

3.1 State of the Art

Often, edge-based segmentation approaches are utilized to determine boundaries and borders of objects in video images. However, the results of typical edge detection operators are normally not suitable for the immediate derivation of a closed boundary because the found contour pixels do not automatically yield a closed contour in the sense of a 4- or 8-neighborhood relationship of pixels. In gray-level images, on the one hand discontinuities in the edge image may affect the proper identification of boundaries. On the other hand, any kind of thresholding that is applied to the edge values may lead to gaps in the boundaries of objects because weak edges may be eliminated. Thus, further supplementary processing steps have to be appended in order to combine separated edge segments and to yield a closed contour. Moreover, the found edge segments have to be grouped to objects. For this purpose, a decision has to be made, in which case the distance between edge segments is large enough to assign those segments to the same instead of separated objects.

There are several well-known techniques which deal with these problems. Basically, these methods differ on the one hand in the strategy that leads to the final border construction and on the other hand in the amount of prior information that is integrated into the process (compare [SON98], p. 134-176). First of all, edge segments have to be determined. A common approach is *edge thresholding* that is applied to edge magnitude images. This approach is based on the assumption that noise and other disturbing influences lead to non-significant edge values, which can be removed by simple thresholding. However, the basic problem is to find a global appropriate threshold providing a result which is neither ”over-thresholded” nor ”under-thresholded” [KUN87]. A more sophisticated approach is *thresholding with hysteresis* that employs a lower and an upper threshold, which are chosen according to an estimated signal-to-noise ratio [CAN86]. Borders which have been determined by one of these methods are often missing important parts or may be affected by noise. Taking into consideration the spatial neighborhood and context of edges may improve the edge detection results. For instance, a weak edge between two strong edge pixels is most probably also a part of an edge segment. Techniques following this scheme are normally refered to as *edge relaxation* methods [ROS76, HAN90, HAN78]. While these edge relaxation methods try to close gaps of the detected edges, they do not guarantee to yield closed boundaries of objects. An approach that is more comparable to our technique is *border tracing*. Common border tracing techniques require a preprocessing of the video images that has already determined contiguous regions within the image that have been labeled respectively. Based on these regions, the

border is traced pixelwise, for instance by clockwise searching for neighboring edge pixels [PAV77]. This approach is very similar to our introduced technique. However, border tracing is only applicable, if a closed boundary around objects or more generally around regions already exists. Thus, the main advantage of our approach is its ability to close gaps and to approximate the contour of objects which may have been segmented incompletely. Other border tracing methods work directly on gray-level images and try to find paths of high-gradient edge pixels [DUD76, BAL82]. The usage of graph structures is also a very common approach in border detection methods. Often, additional knowledge such as known starting or end points of the boundary is required for these approaches [MAR72, NIL82]. Moreover, graph-based approaches are often based on spatial edge directions and vertices of the graph are neighboring pixels, so the required input data is different compared to our approach. As an advantage, general problem-solving methods for graph structures such as cost minimization algorithms, can be applied. However, often discontinuities in the edge images require heuristics in order to close small gaps [SON98]. An extensive survey of border detection and contour approximation methods can be found in [SON98]. Because of the close relationship between the extraction of contour pixels and the linking of the found edge pixels, these topics are often treated together, e.g [SAB97]. A well-known approach to contour approximation problems are *active contour models* or so-called ”snakes” [KAS88]. Snakes are defined as energy-minimizing splines and are matched as a deformable model to an image with respect to energy minimizing functions. An important aspect for the application of snakes is the existence of a priori knowledge or some kind of image understanding process that specifies a starting position and then pushes the snake towards an appropriate result. Snake-line solutions work well and have been analyzed and improved continuously. For the realization of active contours, fast algorithms have been proposed [WIL92]. We present a comparison of our approach to active contours based on the *greedy* snake algorithm [WIL92] and the *gradient vector flow* snake [XU97, XU02] in section 3.10.

Another well-known approach in the context of contour approxiamtion are Hough transforms which are often used for the detection of straight lines, circles or ellipses [HOU62]. Hough transforms are well suited for combining separated edge segments. However, the obtained result is not necessarily suitable as contour approximation for arbitrary objects. Nevertheless, the results of a Hough transform may be utilized for the initialization of active contour models [LAI94]. A relatively simple method to determine an approximated closed contour of a given set of pixels is the calculation of the convex hull polygon [PRE85]. The convex hull polygon closes even large gaps within the edge segments of objects. However, as a severe disadvantage, concave parts of objects are approximated insufficiently. Our algorithm extends the capabilities of the convex hull polygon, so even concave objects are approximated sufficiently.

3.2 Scenarios of Application

Our contour approximation algorithm is able to approximate the contour of a given set of object pixels. This requires the determination of these object pixels during some preprocessing operations, for example by using a background subtraction approach as introduced in chapter 2. In additon, there are further possible input sources. For instance, a human hand can be segmented using skin color information. Then the shape of the hand can be approximated using the introduced approach. Such a process normally requires some kind of thresholding

to finally distinguish between object and background pixels. As already shown in section 3.1, discontinuities in edge images are a common problem in this context because weak edges may be eliminated by thresholding operations leading to gaps in the boundaries. The algorithm introduced here is not suitable to supplement the edge detection process in video images in general. It is applicable only, if sets of object pixels are already isolated. Based on these isolated sets of pixels, an approximated closed contour for the objects is determined. Moreover, the algorithm expects a two-valued input image and cannot utilize gray-level edge images. Thus, some kind of thresholding has to be performed before applying our approach.

The algorithm is able to recognize different objects, if there is a sufficiently large distance between them. More exact information about this aspect as far as the parameterization and the size of the distance is concerned is deduced mathematically in section 3.7. However, our algorithm is not able to determine or to detect any kind of occlusion of objects. The detection of different objects is based on the spatial neighborhood of object pixels only. Our approach is based on the calculation of convex hull polygons within smaller box-shaped image areas. The chosen size of the boxes is essential for the capability of the algorithm to close gaps of specific lengths. On the one hand, a small box size leads to a closer approximation of the boundaries, but, on the other hand, larger gaps may not be closed. In this case, the determined contour cuts into the object. A larger box size leads to a more inexact contour approximation, but larger gaps are more likely to be closed. An automatic determination of box size parameters would be preferable in order to yield optimal results for arbitrary input images. However, this would require some kind of image understanding process or prior knowledge about the objects, which we do not assume to have. The relationship between the chosen box size and the capability of the algorithm to close gaps is analyzed theoretically in section 3.7.

3.3 Starting Basis and Goal

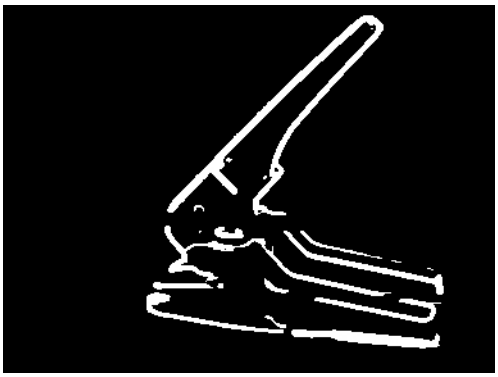
The application of our algorithm requires some preprocessing operations, which ideally yield a large amount of object pixels, for instance the multi-level segmentation system introduced in chapter 2 may be applied. However, the application of our contour approximation algorithm does not depend on any specific preprocessing method. In order to have a common starting basis and to abstract from any specific segmentation algorithm, we assume a general representation f of an input image, which assigns a binary number to every pixel (x, y):

$$f : (x, y) \longrightarrow \begin{cases} 1\text{: pixel belongs to an object} \\ 0\text{: affiliation of pixel is unclear} \end{cases} \tag{3.1}$$

The definition of equation (3.1) explicitly takes into account that a ”0” as input value does not necessarily imply that the pixel belongs to the ”background”. On the contrary, such a pixel may either belong to a weak edge that has been removed by some kind of thresholding operation, to one of the objects or indeed to the background. Figure 3.1 shows an image of a punch on a desk, which has been segmented by our system. As can be seen, the contour of the punch has not been found completely. A canonical approach such as border tracing is not applicable because the 4- or 8-neighborhood relationship of pixels cannot be utilized for the determination of a closed contour.

The specific task of our algorithm is to approximate the boundaries of an arbitrary number of foreground objects represented by $f(x, y) = 1$ for every pixel $\vec{p} = (x, y)$. The result of

(a) Exemplary input image



(b) Segmented image

Figure 3.1: Exemplary video image (a) and required input for contour approximation (b).

an execution of the algorithm is a set of closed polygonal chains (CPC), exactly one for each foreground object. The closed polygonal chains are represented by a list of pixel positions. Succeeding positions in the list denote an edge segment. The last and the first element of the list finally close the polygonal chain. The CPCs should meet the following requirements:

1. The CPCs are closed.

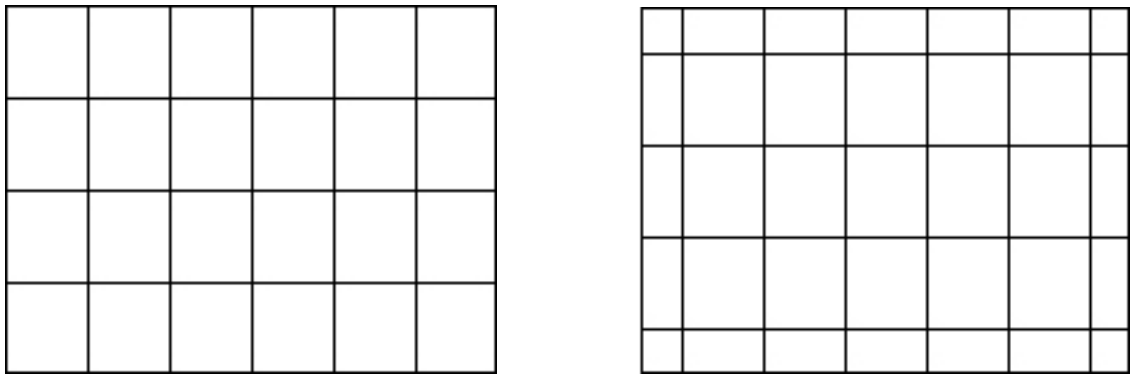
2. Each polygon encloses the whole foreground object. As a consequence, each pixel (x, y) with $f(x, y) = 1$ lies within the polygon and none lies outside. The pixels of the CPC are considered to belong to the object.

3. The CPCs close small gaps within the contour of the objects, which may be caused by faulty or inaccurate segmentation results.

4. The CPCs should not enclose too many background pixels.

Requirement 4 is rather subjective because of the unclear affiliation of the ”0” input pixels. As a matter of fact, the requirements 1, 2, and 3 are fulfilled by the convex hull polygon of the set of foreground edge pixels. The calculation of the convex hull polygon is a well-known approach to shape representation and description, see for example [SON98], and can be calculated in $O(N \log N)$ time, where N is the number of points. For algorithms and run-time considerations see for example [PRE85]. Nevertheless, the main disadvantage is that non-convex foreground objects and, accordingly, concave parts of foreground objects are not approximated sufficiently, in contrast to requirement 4. Thus, we propose a suitable way to combine the advantages of the convex hull properties with the ability to approximate non-convex objects and thus fulfilling all the requirements of a sufficient contour approximation. This is also the reason, why the punch has been chosen as an exemplary input object: It is a rather concave object. As we will show in our comparison in section 3.10.2, our requirement 2 is a particular problem for snake-based algorithms because they tend to cut off parts of the objects.

The basic idea of our algorithm is to subdivide the image area into two layers of rectangular grids of boxes as shown in figure 3.2. Every box in the first layer is overlapped by four boxes of the second layer and vice versa, except for the smaller border boxes from the second layer



(a) First layer of boxes (b) Second layer of boxes

Figure 3.2: Two overlapping box grids (a) and (b) are utilized to subdivide the image area.

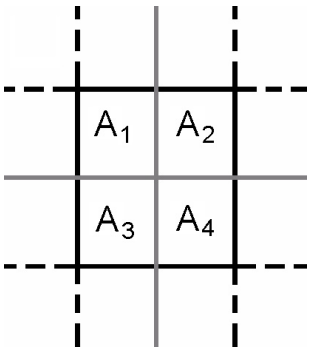


Figure 3.3: The intersection areas A_i of the box layers have an almost identical size. The first layer is depicted in black, the second in gray.

(see figure 3.2). The placement of the second layer of boxes is intended to be symmetrical, so the resulting intersection areas A_1, A_2, A_3 and A_4 have an almost identical size (see figure 3.3). We have relaxed this requirement to "almost identical" because truncation effects may lead to small differences in the sizes. We use the following notations for the size of the boxes (counted in pixels):

- $N_{B,hor}$: horizontal width of boxes
- $N_{B,ver}$: vertical height of boxes
- $N_{O,hor}$: horizontal width of overlapping intersection area of boxes
- $N_{O,ver}$: vertical height of overlapping intersection area of boxes

In figure 3.4, the punch is used as exemplary input again, with both layers of boxes superimposed. We require neighboring to have a common border line within the same layer. As a consequence, object pixels located on a border line are considered for operations within both boxes. Let H_1, H_2 be the number of horizontal boxes and V_1, V_2 the number of vertical boxes for layer 1 and 2, respectively. H_2 and V_2 depend on the parameters of the first layer:

$$H_2 = H_1 + 1 \quad \text{and} \quad V_2 = V_1 + 1.$$

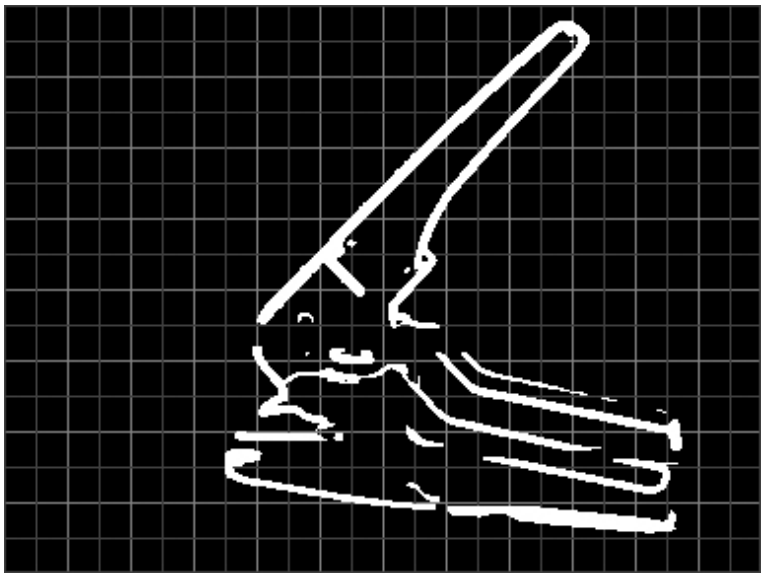


Figure 3.4: Exemplary input image with box layers superimposed. The first layer is shown in bright gray and second layer in dark gray.

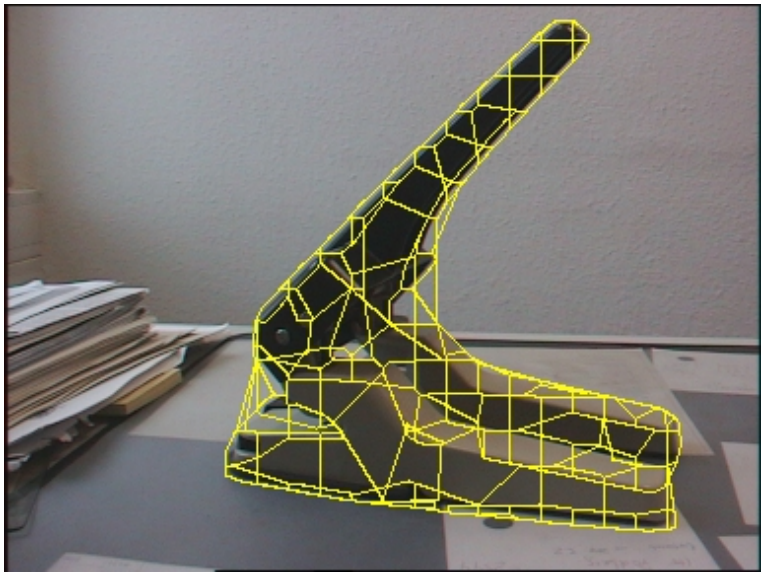


Figure 3.5: Convex hull polygons (yellow) of both layers superimposed on the original image.

In summation, there are

$$N_B = H_1 \cdot V_1 + H_2 \cdot V_2 \tag{3.2}$$

different boxes.

The central idea of our approach is to calculate the convex hull for all the pixels $f(x, y) = 1$ of the input image (see equation (3.1) at the beginning of this section) in each of the N_B boxes in both layers separately. If the convex hull contains less than three points, the box will be treated as empty. As a result, this method yields a large number of convex CPCs that are shown superimposed in figure 3.5. The convex CPCs approximate the concave object rather

well as far as the ambient boundary is concerned (the example image has a size of 384×288 pixels and the box size is 32×32 pixels). For our algorithm, only the points and the implicitly given edges of the so found N_B convex hull polygons are important leading to a reduction of the total number of foreground pixels under consideration. Our algorithm determines the ambient boundary of the object by calculating intersection points of neighboring convex hull polygons and tracing the boundary in either clockwise or counterclockwise order around the object. The details are explained in the following sections where we present algorithms for the detection of disjoint objects and for the derivation of an approximated contour of the detected objects.

3.4 Grouping and Detection of Disjoint Objects

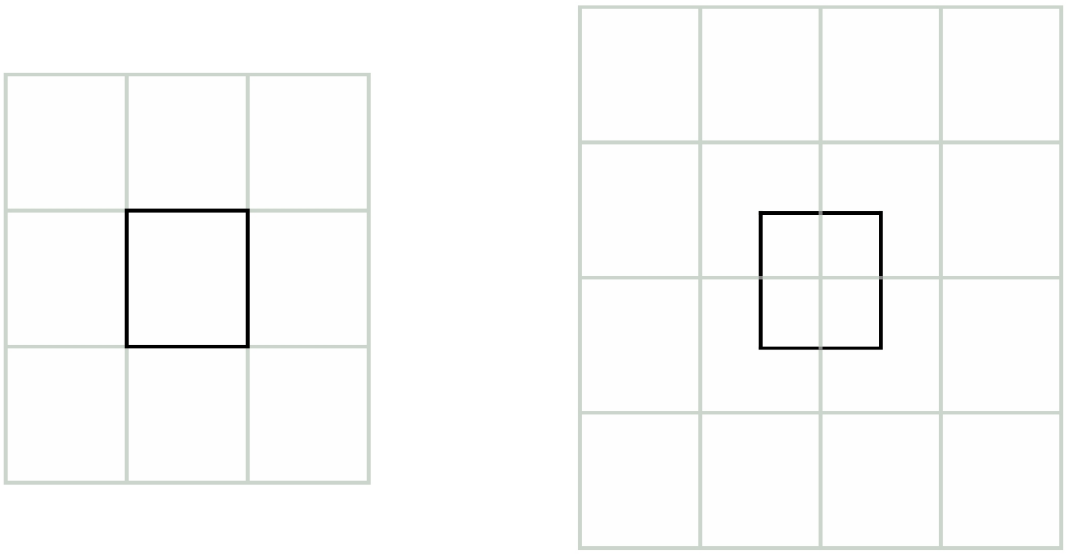
First of all, we determine which areas of the image (represented as boxes) belong to the same foreground object. We simplify this problem by considering our previously defined boxes instead of separate pixels. Two boxes i and j are considered to belong to the same foreground object, if at least one foreground pixel is situated in their intersection area. This can be realized using a *Union-Find* data structure. The Union-Find data structure represents sets of elements. Its operation "Find" returns the set, to which an element belongs. The "Union" operation replaces two existing sets with their union. In our case, the elements are boxes. Initially, every box defines its own set. During execution of the algorithm, sets of boxes will be merged into one set, if two of its boxes i and j have at least one foreground pixel belonging to the same foreground object, i. e., there is at least one foreground pixel in their intersection area. In algorithm 1, the Union-Find data structure is utilized to yield sets of boxes. After execution of the algorithm, each set of boxes covers exactly one foreground object. For more information about the Union-Find data structure see for example [AHO82].

Algorithm 1 Find disconnected foreground objects

```
Generate a set for each box i (i = 1, ..., N_B) that contains a convex CPC.
for each box i with 1 ≤ i ≤ N_B do
  for each box j that shares an intersection area A_I with box i do
    if there is at least one foreground pixel within A_I then
      perform a Union operation for the sets of box i and j.
    end if
  end for
end for
Let N be the number of sets (i.e. objects)
for each object 1 ≤ i ≤ N do
  Assign object number i to each box of set i.
end for
```

3.5 Determination of Approximated Hull Polygon

Running the algorithm presented in section 3.4 determines the number of foreground objects in the image and assigns an object number to each box. Boxes that do not belong to any foreground object are marked as background.



(a) Neighboring boxes in first layer (b) Intersecting boxes in second layer

Figure 3.6: Neighboring boxes of the same layer (a) and of the respective other layer (b), which have to be considered for intersection points with the current box (black).

The following algorithm approximates the contour of exactly one object. Further objects are processed in the same manner. Without loss of generality, we assume to find a clockwise circulation around the object and that the calculated convex hull CPCs are available as a list of points in clockwise order as well. The counterclockwise case can be treated analogously. In order to find a contour approximation for an object, we first need two arbitrary starting points $\vec{p}_0$ and $\vec{p}_1$ belonging to the contour. These points can be found by calculating the convex hull of the whole object and choosing two arbitrary succeeding points as $\vec{p}_0$ and $\vec{p}_1$. For this purpose, only the points of the already determined convex CPCs of the object are used. The point $\vec{p}_a = \vec{p}_1$ is taken as starting position for the circulation around the object and $\vec{p}_b = \vec{p}_0$ is assumed to be a predecessor of $\vec{p}_a$.

The following steps are repeated until $\vec{p}_1$ is reached again as current position: The current position $\vec{p}_a$ is added to the list L of ambient hull points. For $\vec{p}_a$, the boxes b_x are determined, in which $\vec{p}_a$ occurs. For each box b_x that contains $\vec{p}_a$, the succeeding point $\vec{p}_x$ of the convex hull for this box is taken. The straight line $\overline{\vec{p}_a\vec{p}_x}$ is considered to be possibly part of the ambient hull. But at first, it is determined, whether there are intersection points $\vec{p}_i$ of $\overline{\vec{p}_a\vec{p}_x}$ and another convex hull line of one of the neighboring boxes. If an intersection point $\vec{p}_i$ is found, it will be inserted into both hull polygons and $\vec{p}_x$ is set to $\vec{p}_x = \vec{p}_i$. In order to find intersection points, the hull polygons of the neighboring boxes of the same and of the respective other layer have to be considered. See figure 3.6 for the boxes under consideration.

In figure 3.7, an example is shown. The current positions may for example be $\vec{p}_b = (1)$ and $\vec{p}_a = (4)$. Then $\vec{p}_x = (3)$ is chosen as one of the next possible points. But first it is determined, whether there are intersection points $\vec{p}_i$ on $\overline{\vec{p}_a\vec{p}_x}$. Such a point $\vec{p}_i$ (encircled in the figure) indeed exists because of the intersection of $\overline{\vec{p}_a\vec{p}_x}$ and $\overline{(5)(7)}$. Thus, $\vec{p}_x = \vec{p}_i$ is set.

Having determined and inserted possible intersection points into the CPCs, the clockwise angle φ_x between $\overline{\vec{p}_b\vec{p}_a}$ and each $\overline{\vec{p}_a\vec{p}_x}$ for each box b_x is calculated. The point $\vec{p}_n = \vec{p}_x$ with the smallest angle φ_x is taken as next ambient hull point of the approximated contour. Then

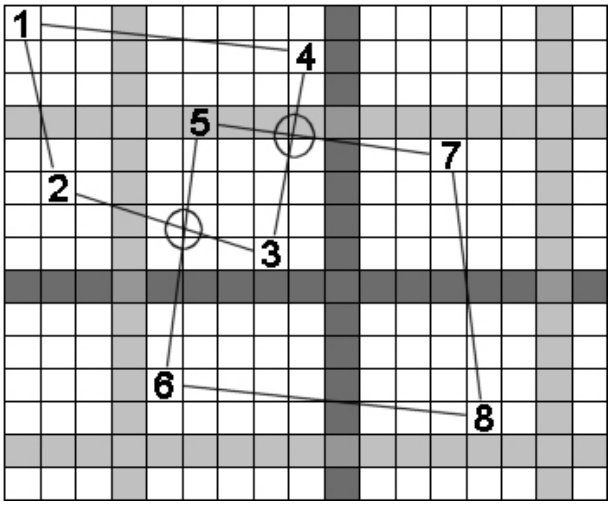


Figure 3.7: Determination of intersections points of CPCs (encircled).

$\vec{p}_b = \vec{p}_a$ and $\vec{p}_a = \vec{p}_n$ is set. These steps are repeated until $\vec{p}_1$ is reached again as current position. In order to clarify this method, it is summarized in algorithm 2.

Algorithm 2 Find object contour

Calculate convex hull of the whole set of CPCs of the boxes belonging to the foreground object.
Take two arbitrary succeeding points $\vec{p}_0$, $\vec{p}_1$ of this convex CPC (w.l.o.g. in clockwise order) as starting line with $\vec{p}_a = \vec{p}_1$ and $\vec{p}_b = \vec{p}_0$.
repeat
  Append $\vec{p}_a$ to the list L of points for the ambient CPC.
  for each box b_x that contains $\vec{p}_a$ **do**
    Determine succeeding point $\vec{p}_x$ of $\vec{p}_a$ in hull CPC of box b_x.
    repeat
      Find possible intersection points $\vec{p}_i$ of $\overline{\vec{p}_a\vec{p}_x}$ with hull lines of neighboring boxes.
      Insert $\vec{p}_i$ in hull CPCs.
      Set $\vec{p}_x = \vec{p}_a$.
    until all intersection points are found.
    Redetermine succeeding point $\vec{p}_x$ of $\vec{p}_a$.
    Calculate clockwise angle φ_x by regarding $\vec{p}_b$, $\vec{p}_a$ and $\vec{p}_x$ as a triangle.
    Find the vertex $\vec{p}_n = \vec{p}_x$ that has the smallest angle φ_x.
  end for
  Set $\vec{p}_b = \vec{p}_a$ and $\vec{p}_a = \vec{p}_n$.
until $\vec{p}_a = \vec{p}_1$
Return list L of contour points

The algorithm yields a list L of points that define the final approximated hull polygon as a CPC for the object under consideration. By always choosing the smallest possible angle, it is ensured that only ambient points are added to the list, otherwise the graph would not have been constructed properly.

Summarizing our technique leads to algorithm 3, which yields as result the number of recognized foreground objects and determines a list of contour points as CPC for each object.

Algorithm 3 Contour approximation

```
Find and determine number N of disconnected foreground objects (algorithm 1)
for 1 ≤ i ≤ N do
    Find object contour (algorithm 2)
    Store returned list of contour pixels as L_i
end for
return N and L_1, ..., L_N
```

3.6 Problems Resulting from Discretization

In this section, some problems are treated, which result from the discretization of pixel positions, especially of the calculated intersection points. These intersection points are generally rounded to discrete pixel positions simplifying the determination of the ambient hull significantly. Unfortunately, under unfavorable conditions the discretization may lead to problems, if two lines of the same hull polygon intersect. That may be the case after rounding and inserting an intersection point. Similarly, it is possible that the insertion of a rounded intersection point violates the clockwise order of a CPC. Both cases may lead to a deadlock in the algorithm, when calculating the ambient circulation around the object. In order to cope with this severe problem, two additional methods are incorporated in the algorithm. First, we examine after each insertion of intersection points, whether the new line segments intersect with one of the other line segments of the current CPC. In this case, this additional intersection point will be inserted into the CPC. Second, the CPC of this box is recalculated with respect to the required clockwise order, which may have been violated because of the insertion.

Another problem caused by the discretization may be an intersection point laying at a position that would have been reached from the previous position during the *repeat*-loop of the algorithm, if the point had existed before. In this case, if the new point meets the smallest angle condition, the algorithm may choose it as ambient hull point, leading to a deadlock, as well. In order to cope with this problem, the current position is removed from the list of the approximated contour points and the algorithm recalculates the angles of the previous position. If appropriate, a new contour point may be chosen.

3.7 Performance Analysis

The convex hull computed for a locally restrained area within an image has the property to close small gaps in the contour of an object. This property is considered to be important and was required in section 3.3 because image preprocessing most probably does not provide closed boundaries of objects as explained in section 3.1. Our approach to calculate the convex hull in box-shaped areas limits the edge-linking capabilities of the convex hull, if a gap in the boundary is incidentally located on the border between two boxes. In order to illustrate this problem, figure 3.8 shows an enlarged clipping of figure 3.4.

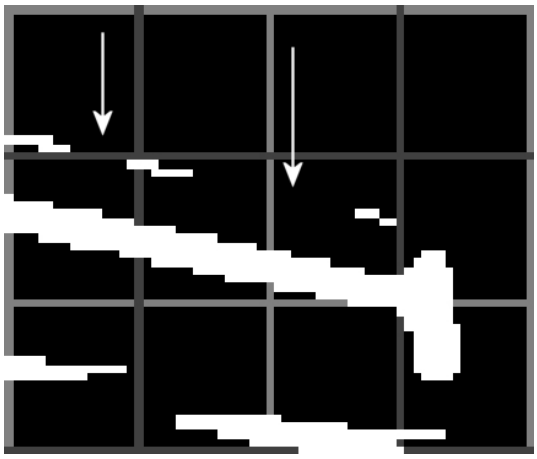


Figure 3.8: Gaps in the contour located between boxes (right arrow) are found by an overlapping box.

The convex hull polygon of the upper left box (in bright gray) closes the first gap (indicated by left arrow) very well, but is not able to close the second gap (right arrow) that is located between the upper left and the upper right box. Our concept to have a second layer of symmetrically offset boxes solves this problem. The gap is closed by the lower middle box of the second layer (in dark gray) because the convex hull calculated for this box provides a CPC closing the gap. Keeping this idea in mind, the close relationship between the handling of gaps and the disjointness of objects becomes obvious. In order to gain some reliable information about the behavior of our technique depending on the chosen box size, the parameters $N_{B,hor}$, $N_{B,ver}$, $N_{O,hor}$ and $N_{O,ver}$ are analyzed. These parameters and the given image size automatically lead to the number of boxes H_1, V_1, H_2 and V_2.

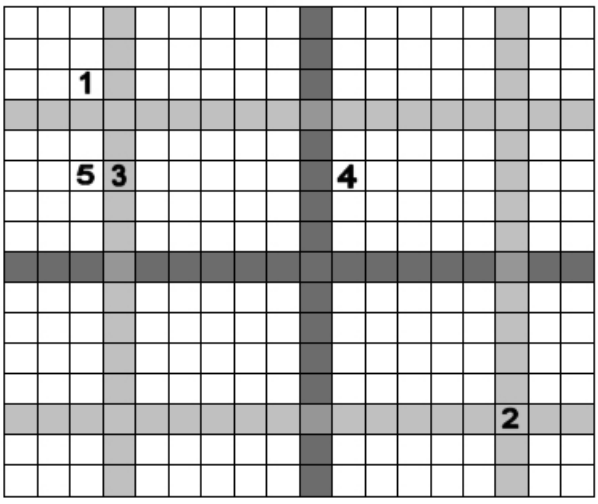


Figure 3.9: Extreme cases for the recognition of connected components

We have to point out that the position of a contour gap has a certain influence as far as its detection and handling by our method is concerned. If for example only the foreground pixels (4) and (5) in figure 3.9 are given, none of the boxes will produce a hull line closing the

gap because these points do not appear in the same box. If the points (4) and (5) are shifted one pixel position to the left or to the right, the gap will be closed. Thus, the behaviour of our algorithm depends on the position of both the boxes and the foreground objects in relation to each other. In order to illustrate the relationship, the extreme cases are analyzed. For example, for two points laying on the same horizontal position the minimum distance required to ensure that these points are not connected is $N_{B,hor}$ (see points (1) and (2) in figure 3.9). Generalizing this rule for two points with arbitrary positions yields:

Disconnectivity Criterion

The distance

$$D_{disconnect} = \sqrt{{N_{B,hor}}^2 + {N_{B,ver}}^2} \tag{3.3}$$

is the minimum distance between two points $\vec{p}_1$ and $\vec{p}_2$ for which $\vec{p}_1$ and $\vec{p}_2$ are considered to be disjoined. Two disconnected objects O_1 and O_2 will be recognized as two objects by the algorithm, if for two arbitrary points $\vec{p}_1 \in O_1$ and $\vec{p}_2 \in O_2$ the condition

$$||\vec{p}_1 - \vec{p}_2|| > D_{disconnect} \tag{3.4}$$

holds.

See points (1)-(2) in figure 3.9 as an example. The opposite point of view leads to the

Connectivity Criterion

The distance

$$D_{connect} = \min(N_{O,hor}, N_{O,ver}) \tag{3.5}$$

is the maximum distance between two points $\vec{p}_1$ and $\vec{p}_2$ for which $\vec{p}_1$ and $\vec{p}_2$ are considered to belong to the same object. An object O will be recognized as one object, if the distance between an arbitrary point $\vec{p}_1 \in O$ and its closest neighbor $\vec{p}_2 \in O$ is

$$||\vec{p}_1 - \vec{p}_2|| \leq D_{connect}. \tag{3.6}$$

See points (3)-(4) in figure 3.9 as an example. Taking into account the *Connectivity* and *Disconnectivity Criterion*, it is possible to predict the behavior of the algorithm. Nevertheless, for distances d with

$$D_{connect} < d < D_{disconnect} \tag{3.7}$$

the result, whether a gap is closed or not, is not predictable because it depends on the position of the object relative to the box layers.

In order to measure the calculation time, the algorithm was applied to a series of 950 images. The series was taken from the interaction application which is presented in chapter 4.2, an exemplary image is shown in figure 4.2 on page 104. The contour approximation algorithm was applied to the images of the sidewards camera. Images showing merely the background were left out for this evaluation. Each image had a size of 192×144 pixels and was tested for $N_B = 413$ ($N_{B,hor} = N_{B,ver} = 12$) and $N_B = 636$ ($N_{B,hor} = N_{B,ver} = 10$) boxes. The number of boxes is the crucial factor and not the size of the image as far as the calculation time of the algorithm is concerned. The whole number of pixels within each box area has to

be considered only once, namely during the calculation of the convex hull polygon of this box area. Afterwards, the number of points of the convex hull polygons is the decisive factor as far as the calculation time is concerned. Thus, the calculation time of our algorithm is largely independent of the actual image size, but depends on the chosen box size and the resulting number of boxes.

Measured value	413 boxes	636 boxes
Contour points	76.6	97.2
Considered points	205.7	270.9
Intersection points	15985.4	20120.9
Calculation time	0.026 s	0.032 s

Table 3.1: Performance for $N_B = 413$ and $N_B = 636$ boxes.

For each image, the number of points in the returned contour approximation list, the number of considered points for the result list, the number of calculated intersection points, and the calculation time[1] were determined, which are shown in table 3.1. While the number of calculated intersection points is impressive, the algorithm works for both parameter settings in realtime and thus can be considered to be suitable for interaction applications.

3.8 Time Coherence

Principally, the introduced contour approximation algorithm may be applied both to still and to live video images. The algorithms presented so far utilize one input image only. Thus, in live video applications, every image is processed as a still image, neglecting additional information about previously determined object boundaries. In chapter 4, where several examples of application are presented, the segmentation may sometimes provide faulty results, which lead to outliers of the contour approximation. An example of an outlier is illustrated in figure 4.9(b) on page 109. As we have shown in section 3.7, the calculation time of our approach allows for realtime applications, in which objects move relatively slow, leading to slight changes from frame to frame only. For this kind of applications, we have developed a further algorithm which incorporates time coherence and leads to smoothed approximation results over time. As a consequence, sporadic outliers of the approximated boundaries are significantly reduced.

Basically, the smoothing algorithm maintains a second boundary for each object that is the result of previous smoothing operations. After the first appearance of an object, this second boundary is initialized with the current boundary. In each new frame, the contour approximation algorithm 3 (see section 3.5) is applied and the obtained boundaries are combined with the second, preceding boundary polygon. We assume that both boundary polygons are given as lists L_p and L_q of succeeding edge points. This is exactly the result of the presented contour approximation algorithm. The determination of a current smoothed boundary polygon L'_p is performed as illustrated in figure 3.10. Let the dotted, upper polygon (points depicted by stars and enumerated as $\vec{p}_1, ..., \vec{p}_n$) denote the former boundary L_p of preceding frames,

[1] On a Pentium-III 700 MHz based system, considering the contour approximation algorithm without any previous segmentation only.

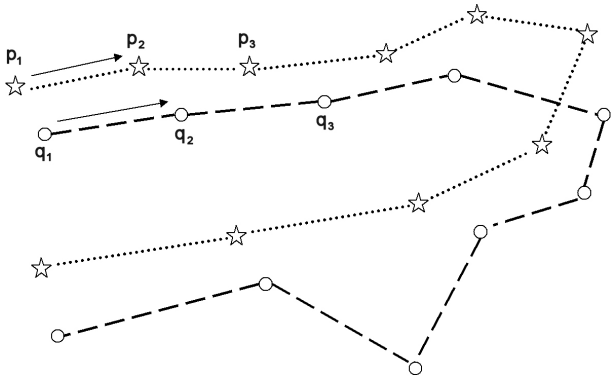
and the second, lower polygon denote the current polygon L_q (enumerated as $\vec{q}_1, ..., \vec{q}_n$). As can be seen, the current polygon L_q has an outlier at the bottom, in comparison to the former smoothed polygon L_p. Both polygonal chains are illustrated in figure 3.10(a). The arrows depict the clockwise orientation of the polygonal chains, which are provided by the contour approximation algorithm.

Beginning with the first element $\vec{q}_1$ of the current boundary, the closest neighbor of the former boundary L_p is determined on the basis of the Euclidean distance. For initialization purposes, the whole list L_p of points has to be considered. In our example, the corresponding point for $\vec{q}_1$ is $\vec{p}_1$. The algorithm starts with $\vec{p}_a = \vec{p}_1$ and $\vec{q}_b = \vec{q}_1$ and stores the new smoothed boundary in L'_p. The following steps are repeated: First, the closest neighbor of $\vec{q}_b$ in L_p is determined. For this purpose, not the whole list L_p is considered (as for the initialization), but just those successors of $\vec{p}_a$, for which the distance to $\vec{q}_b$ becomes smaller. The stop condition for this loop is reached, when the distance of a successor is larger than the distance of its predecessor to $\vec{q}_b$. If $\vec{p}_a$ is the last element in the list, $\vec{p}_a$ is set to the first element of L_p. We denote the closest successor of $\vec{p}_a$ to $\vec{q}_b$ with $\vec{p}_{best}$. Between $\vec{q}_b$ and $\vec{p}_{best}$ a line segment is constructed and $\vec{q}_b$ is shifted along $\overline{\vec{q}_b\vec{p}_{best}}$ according to a predefined factor γ:

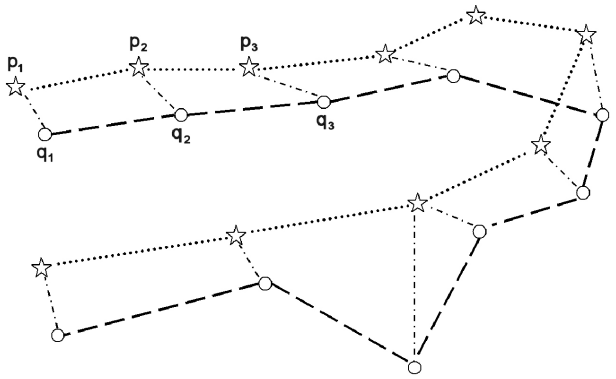
$$\vec{r}_b = \vec{q}_b + \gamma \cdot (\vec{p}_{best} - \vec{q}_b) \tag{3.8}$$

with $\vec{r}_b, \vec{q}_b, \vec{p}_{best} \in \mathbb{R}^2$ and $\gamma \in [0, 1[$. The choice of γ influences the impact of the current boundary on the smoothed polygon. Large values of γ (within the interval $[0, 1[$) lead to a low influence, i. e., the smoothed polygon slowly adapts to the current boundary. Low values of γ lead to a minor smoothing effect, $\gamma = 0$ immediately duplicates the current boundary as smoothed result. In figure 3.10(b), the line segments between the elements of L_q and their closest neighbor in L_p are indicated by the dashed lines. In figure 3.10(c), the new positions which result from the shifting operation are illustrated. The shifted position $\vec{r}_b$ is added to the list L'_p. Moreover $\vec{p}_a = \vec{p}_{best}$ is set and $\vec{q}_b$ is set to its successor. The loop is repeated until $\vec{q}_b$ does not have a successor and the end of list L_q is reached. In our example, the resulting smoothed boundary polygon is illustrated in figure 3.10(d) as thick line. As can be seen in the example, the outlier can partly be compensated by our smoothing algorithm. However, while γ is approximately 0.5 in the example, a larger γ would smooth the boundary considerably better. Figure 3.10 is just an illustration and is not based on exact calculations.

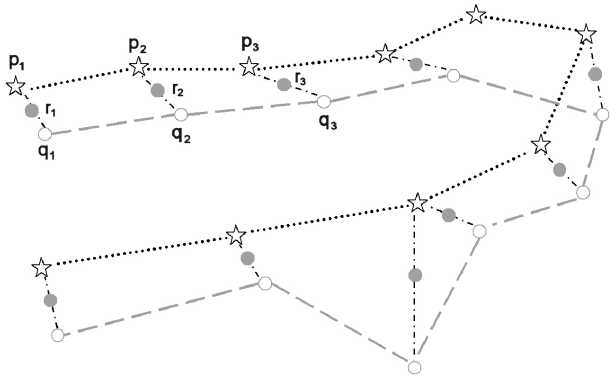
While our smoothing algorithm works well, if the approximated contour polygons move and change slightly, it may fail, if a fast or sudden movement of the objects in the image takes place. For this purpose, we propose an analysis of the current boundary polygon and the calculated smoothed boundary. In the case of a sudden movement, we newly initialize the smoothed boundary with the current contour approximation to react to significant changes in time. In order to decide whether to keep the smoothed contour or to simply duplicate the current contour and to take it as a new starting basis for subsequent images, we average the distances between the corresponding points of the current and smoothed contour and apply a threshold. The whole smoothing algorithm is shown as algorithm 4. Some examples of application of the smoothing algorithm are illustrated in section 3.9.



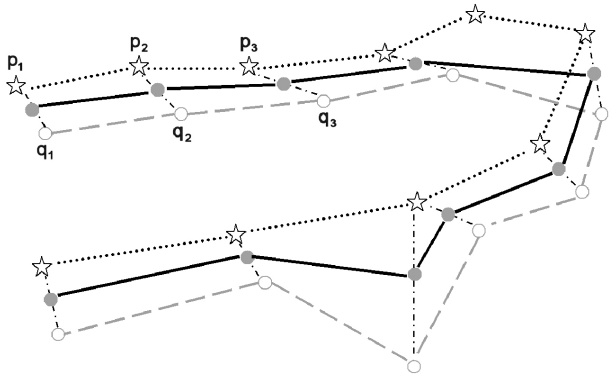
(a)



(b)



(c)



(d)

Figure 3.10: Smoothing of approximated contour.

Algorithm 4 Smoothing Algorithm

Input: Approximated contour polygon L_q
if L_p is empty, i. e., first appearance of boundary **then**
  Initialize L_p with current boundary polygon L_q.
else
  Set $\vec{q}_b$ to the first element $\vec{q}_1$ of list L_q.
  Determine closest neighbor $\vec{p}_a$ for $\vec{q}_b$ by considering all the elements of L_p.
  repeat
    $r_{best} = \infty$.
    repeat
      Set $\vec{p}_i = \vec{p}_a$.
      Calculate Euclidean distance r between $\vec{q}_b$ and $\vec{p}_i$.
      if $r < r_{best}$ **then**
        Set $r_{best} = r$ and $\vec{p}_{best} = \vec{p}_i$.
      else
        Stop condition for this loop is met.
      end if
      Set $\vec{p}_i$ to its successor. If the end of L_p is reached, set $\vec{p}_i$ to the first element of L_p.
    until Stop condition has been met.
    Construct a line segment $\overline{\vec{p}_{best}\vec{q}_b}$ and determine new position $\vec{r}_b$ by shifting $\vec{q}_b$ along this segment according to predefined factor γ (see equation (3.8)).
    Add $\vec{r}_b$ to list L'_p of smoothed boundary polygon.
    Sum up shifted distance between $\vec{q}_b$ and $\vec{r}_b$ in r_{shift}.
    Set $\vec{p}_a = \vec{p}_{best}$.
    Set $\vec{q}_b$ to its successor.
  until End of list L_q is reached, i. e., $\vec{q}_b$ does not have a successor.
  Average out shifted distance r_{avg} by dividing r_{shift} by the number of points in L_q.
  if $r_{avg} < r_{threshold}$ **then**
    Take L'_p as new smoothed boundary polygon.
  else
    Take L_q as new boundary polygon for initialization purposes.
  end if
end if

3.9 Examples

In order to demonstrate the capabilities of our polygonal contour approximation algorithm, some examples with different parameters are presented. Each example is represented by two images, the input image with the box layers superimposed and the resulting image with the original image and the calculated ambient hull polygon (in white) superimposed. For an analysis of the impact of different choices for the box size $N_{B,hor}$ and $N_{B,ver}$, the punch is used as an example in figure 3.11. On the left, the presegmented input image and on the right the original image with the approximated contour as result is shown. The first example in (a) and (b) uses a width and height of $N_{B,hor} = N_{B,ver} = 32$ pixels and $N_{O,hor} = N_{O,ver} = 16$. As a result, for these parameters $D_{connect}$ is determined as

$$D_{connect} = \min(N_{O,hor}, N_{O,ver}) = 16.$$

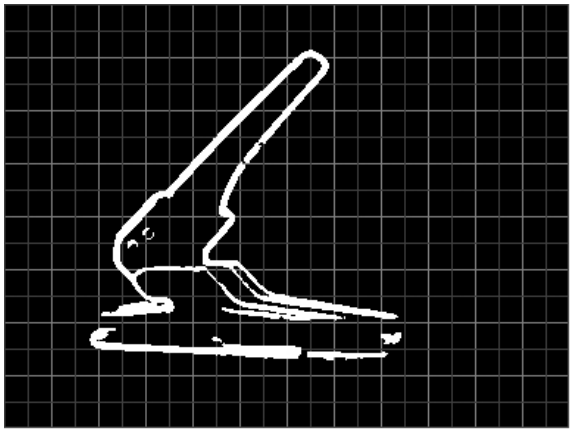
This means that even in the worst case szenario as far as box boundaries and object position are concerned gaps within the contour of the foreground objects will be closed, if they are not larger than 16 pixels. $D_{disconnect}$ is calculated with the maximum appearing sizes

$$D_{disconnect} = \sqrt{32^2 + 32^2} \cong 45,$$

which means that the distance between arbitrary points of two objects has to be at least 45 pixels to ensure that these objects are not recognized as one. The algorithm already yields promising results for these parameters as can be seen in figure 3.11(a) and (b) . The contour approximation can be improved by decreasing the size of the boxes to $N_{B,hor} = N_{B,ver} = 16$ in figure 3.11(c) and (d). Nevertheless, at the lower left side of the object a small piece of the punch is not enclosed. A close look at the input image (c) shows that the contour gap at this position is responsible for this behaviour. As a matter of fact, the results of the contour approximation algorithm depend on the quality of the presegmented input images. In figure 3.11(e) and (f), the size of the boxes has been decreased to $N_{B,hor} = N_{B,ver} = 11$. Here, the contour gaps of the input image (e) are too large to be closed. Thus, the algorithm is not able to approximate the object's contour correctly.

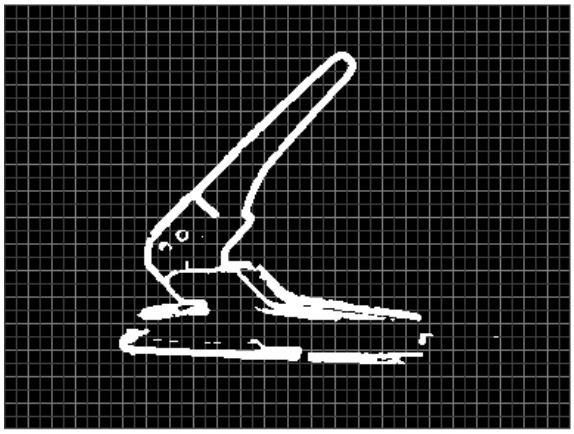
In figure 3.12, two objects with only a small distance between them are shown (with $N_{B,hor} = N_{B,ver} = 19$). The determined contour polygons approximate the objects very well. Moreover, the algorithm correctly recognizes two separate objects because there is no box which contains pixels of both objects.

In section 3.3, we underlined the independence of the contour approximation algorithm from specific preprocessing operations. Especially, it is not necessary to employ the introduced multi-level segmentation system (chapter 2) for the determination of foreground object pixels. In figure 3.13(a) the set of object pixels was determined by the dedicated search for skin color in the normalized RGB color space. This is a common approach for skin color detection, see for instance [YAN98]. In order to remove noise and wrongly classified pixels, the color segmented image has been postprocessed with subsequent morphological *erode* and *dilate* operations [GON92], leading to figure 3.13(a). The image has a size of 192×144 pixels, the box metric was chosen $N_{B,hor} = N_{B,ver} = 9$. Despite the postprocessing with morphological operators, the determined hand region contains some smaller gaps, which are for instance caused by the ring the user wears and wrongly classified pixels. Although the gaps are not too large, they impair techniques like border tracing (see section 3.1). The result of the application of the introduced contour approximation approach is shown in figure 3.13(b). As can be



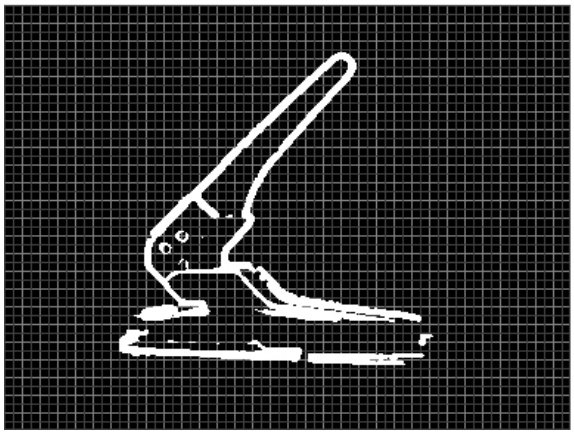
(a) $N_{B,hor} = N_{B,ver} = 32$ pixels

(b) Resulting contour approximation

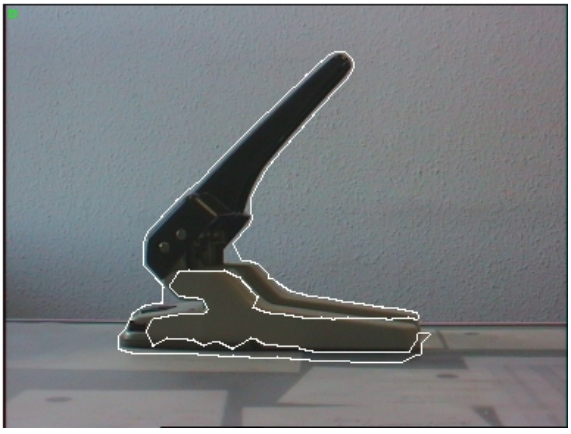


(c) $N_{B,hor} = N_{B,ver} = 16$ pixels

(d) Resulting contour approximation

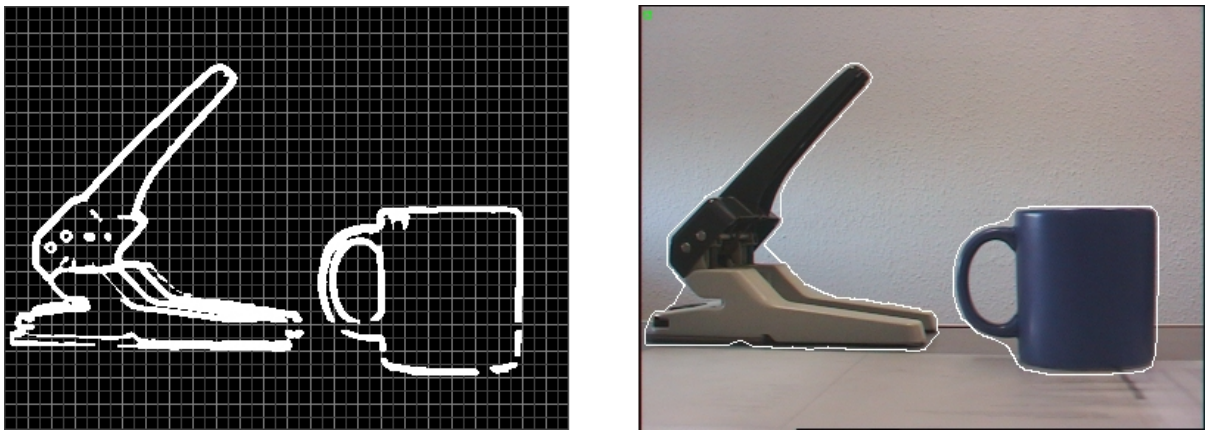


(e) $N_{B,hor} = N_{B,ver} = 11$ pixels



(f) Resulting contour approximation

Figure 3.11: Example for the effects of different box sizes. Promising results can be seen in (a) and (b), which can be improved by decreasing the size of the boxes in (c) and (d). The contour gaps will not be closed anymore, if the size of the boxes is decreased once more in (e) and (f). The relationship between the box size and the length of contour gaps is theoretically analyzed in section 3.7.



(a) Input image (b) Resulting contour approximation

Figure 3.12: Example for two objects with small distance between them.

(a) Input image (b) Resulting contour approximation

Figure 3.13: Input image resulting from dedicated search for skin color.

seen, the calculated boundary closes the gaps sufficiently well and yields a very good contour approximation.

In section 3.8, we have presented a smoothing algorithm that is able to utilize time coherence, which may, for example, be important in live video applications. For this kind of applications, the image and consequently the object boundaries are expected to change slightly from frame to frame. Thus, sporadic failures within the segmentation stage, which lead to sudden outliers of the contour approximation, can be smoothed significantly with the introduced algorithm. As an example of application, we apply the smoothing algorithm to a series of images, which was taken from an interaction application (see chapter 4.2). In this application, the pointing direction of the user's arm is determined. Within the series, the light is switched off leading to some faulty segmentation results (see figure 4.4 on page 105). In figure 3.14 and figure 3.15, the impact of a sudden occurence of an outlier on the smoothed contour is illustrated. Both figures illustrate eight succeeding images, before the occurence of the outlier in (a) and afterwards in (b)-(h). The smoothing factor is set to $\gamma = 0.8$ in figure 3.14, and to $\gamma = 0.9$ in figure 3.15. The blue polygon depicts the current, the yellow polygon the smoothed boundary. As can be seen in both figures, the smoothing algorithm is

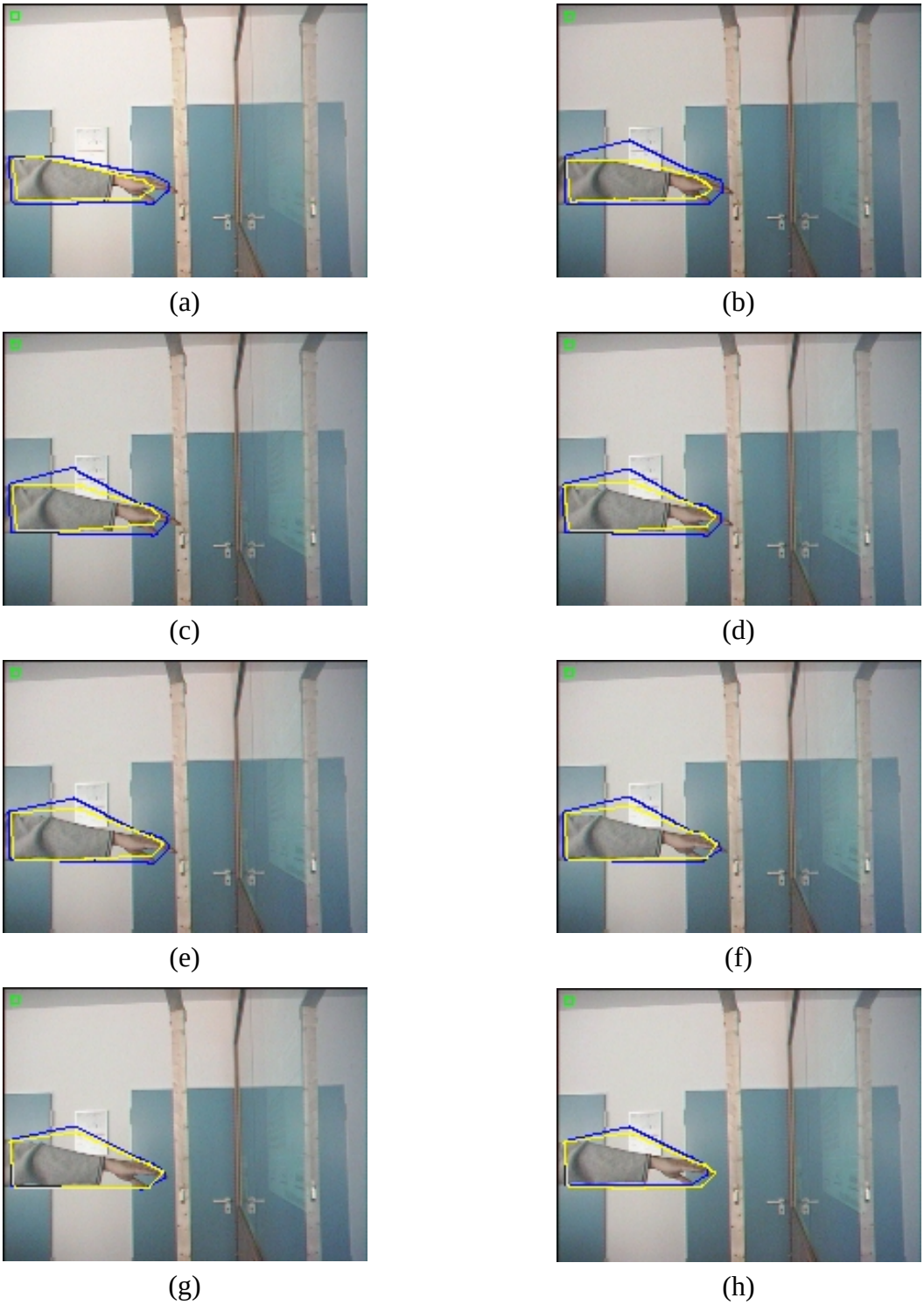
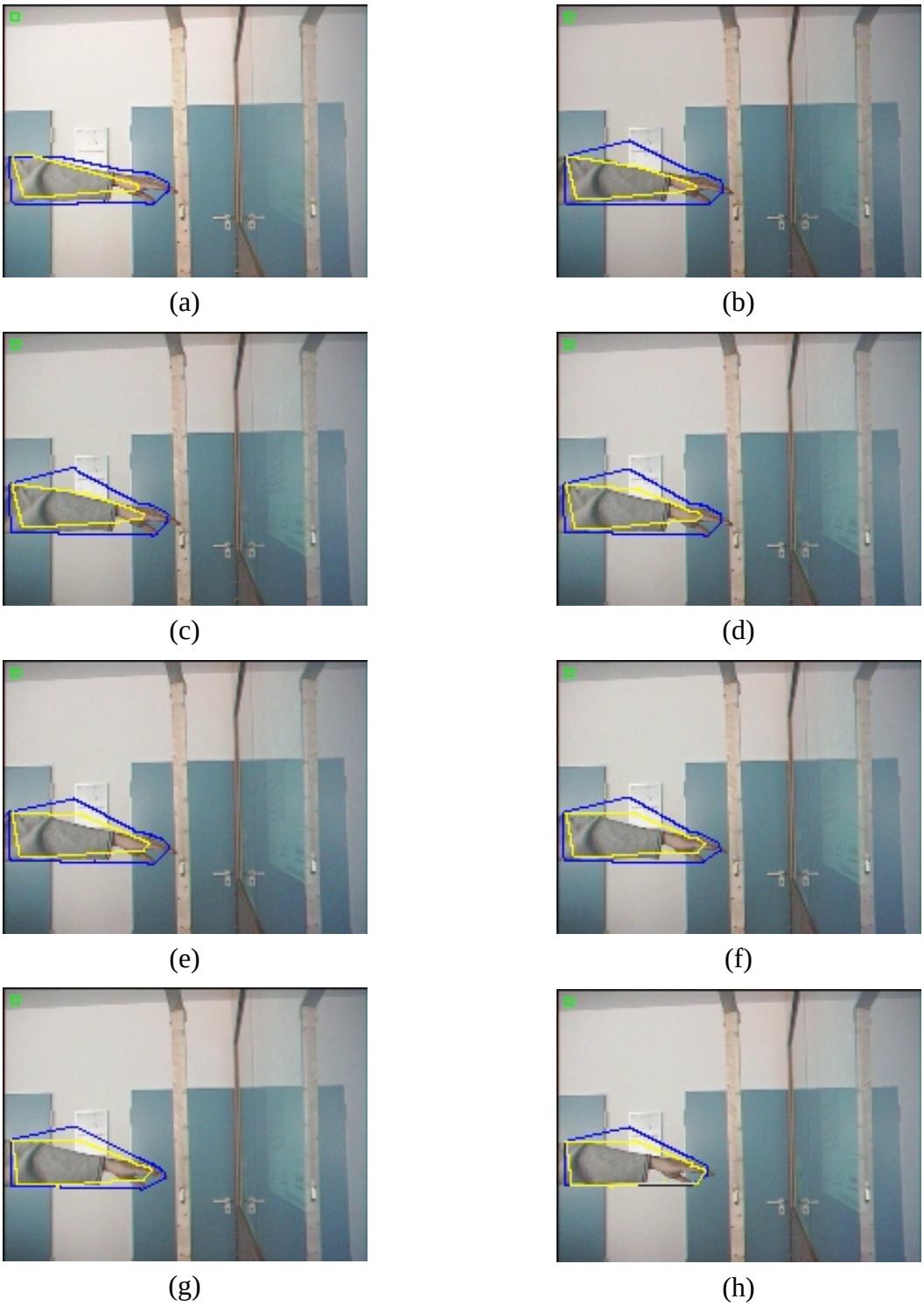


Figure 3.14: Example of application of the smoothing algorithm. The smoothing factor is set to $\gamma = 0.8$, the blue polygon depicts the current approximated contour and the yellow polygon depicts the smoothed contour. In (a), the current and the smoothed contour are more or less identical until the light is switched off in (b). As can be seen, the smoothed contour is slowly adapting to the outlier above the user's arm in (c)-(f) until it finally becomes more or less identical in (g) and (h).



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 3.15: The smoothing factor is set to $\gamma = 0.9$ and applied to the same series of images as shown in figure 3.14. Again the blue polygon depicts the current approximated contour and the yellow polygon depicts the smoothed contour. As can be seen in (b)-(h), the smoothed contour adapts significantly more slowly to the outlier. Thus, in (h) the smoothed result is more ore less identical to figure 3.14(c).

able to compensate sudden occurences of outliers sufficiently well. Depending on the chosen smoothing factor γ, the smoothed boundary adapts faster or more slowly to the current contour approximation.

3.10 Comparison to Other Approaches

In order to evaluate the introduced contour approximation approach, we compare the results of our algorithm to other approaches. In section 3.10.1, we apply both the convex hull and our algorithm to some exemplary images and demonstrate the improvements of our method, which is also based on calculating convex hull polygons. In section 3.10.2, we use the same exemplary images for a comparison to the active contour models (”snakes”) based on the greedy snake algorithm and the gradient vector flow snakes.

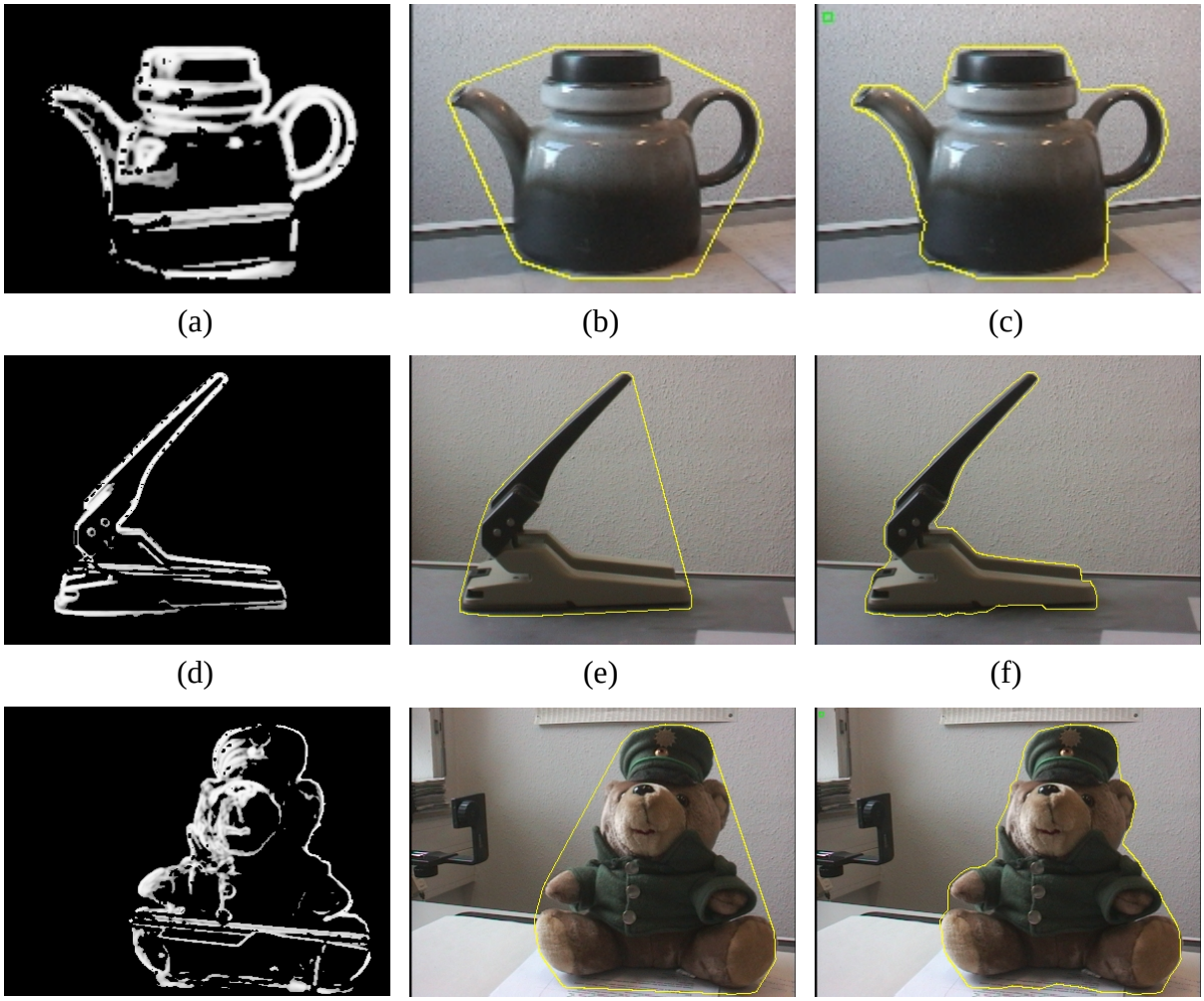
3.10.1 Convex Hull

Because of the close similarity of our approach to the calculation of the convex hull polygon, we present some examples of application in order to underline the improvements of our technique as far as the contour approximation capabilities are concerned. For each example, we calculate both the convex hull polygon and an approximated contour with the introduced approach. As an indication for the capabilities of the contour approximation, we count the number of pixels of the enclosed areas. In figure 3.16, three examples of application are illustrated. In the first row (figure 3.16(a), (b) and (c)), a tea pot is shown. The image size is 192×144 pixels, the box size is 11×11 pixels. The set of input pixels for our algorithm is shown in (a). The convex hull polygon in (b) encloses an area of 13867 pixels, compared to 12007 pixels of our approach in (c). The enclosed area is consequently shrunken to 86.6% of the convex hull area.

In figure 3.16(d), (e) and (f), the punch is shown again. The image has a size of 384×288 pixels, the chosen box size is 11×11 pixels. The set of input pixels is shown in (d). The area enclosed by the convex hull polygon in (d) consists of 35957 pixels, compared to 20019 pixels yielded by the introduced approach in (e). The determined object area is shrunken to 55.7% of the convex hull size.

In figure 3.16(g), (h) and (i), a teddy bear is taken as exemplary object. The image has a size of 384×288 pixels, the chosen box size is 24×24 pixels. The set of input pixels is illustrated in (g). The convex hull polygon in (h) encloses an area of 46608 pixels, compared to 42740 pixels yielded by our approach in (i). The enclosed area could be shrunken to 91.7% of the convex hull area.

As a proof of quality, the visual inspection of the results in figure 3.16 may be considered. Depending on the amount of concave parts of the input objects, the enclosed object area could be shrunken (between 55.7% and 91.7% of the convex hull polygon size) without cutting off parts of the object.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 3.16: The set of input pixels is shown on the left. The convex hull polygon in the middle, superimposed in yellow, does not approximate concave parts of the objects sufficiently. Our introduced approach approximates the contour much better on the right.

3.10.2 Active Contours

Active contour models or so called ”snakes” have been introduced in [KAS88] and evolved enormously since then. The original work [KAS88] currently yields 898 citations in the ResearchIndex [RES02]. A snake is a controlled continuity spline which is typically influenced by image forces and external constraint forces. The image forces attract the snake to salient image features such as lines or edges, the external constraint forces may be directed by some higher level image understanding mechanism. Snakes are defined by an energy function reflecting these forces. An approximated contour is determined by minimizing the energy functional. Let $\mathbf{v}(s) = (x(s), y(s))$ denote the snake points, then the energy functional can be written as

$$E^*_{\text{snake}} = \int_0^1 E_{\text{snake}}(\mathbf{v}(s))ds = \int_0^1 \Big(E_{\text{int}}(\mathbf{v}(s)) + E_{\text{image}}(\mathbf{v}(s)) + E_{\text{con}}(\mathbf{v}(s)) \Big)\, ds \quad (3.9)$$

where E_{int} represents the internal energy of the snake due to bending, E_{image} reflects the image forces and E_{con} the external constraint forces. For our comparison, we utilize the greedy algorithm [WIL92], which is provided as open source in C++ in the Open Computer Vision Library [OCV02] and the *gradient vector flow* (GVF) snake, which is provided as MatLab source code [MAT02, XU02] by the authors of [XU97]. The quantity being minimized by the greedy algorithm is

$$E_{\text{snake}} = \int \Big(\alpha(s)E_{\text{cont}} + \beta(s)E_{\text{curv}} + \gamma(s)E_{\text{image}} \Big)\, ds \quad (3.10)$$

where E_{cont}, weighted by the factor $\alpha(s)$, denotes ”continuity” and is a first-order term which makes the snake act like a membrane. E_{curv} denotes ”curvature” and is weighted by the factor $\beta(s)$. Curvature is derived as second-order term and makes the snake act like a thin plate. Both E_{cont} and E_{curv} correspond to E_{int} in equation (3.9). The E_{image} term is the same as in equation (3.9). External constraints are not included in [WIL92]. The parameters α, β and γ are utilized to balance the relative influence of each of the terms. The greedy algorithm itself works iteratively and examines the neighborhood for each point in order to find a new location providing the smallest value for the energy term. For details of the specification of the different terms see [KAS88, WIL92]. Basically, the GVF snake introduces an external force that overcomes the problem of poor convergence to concave parts of objects of active contour models [XU97]. The gradient vector flow is computed as a diffusion of the gradient vectors of the image, forcing the active contours into concave parts of objects.

For our comparison, we use the same input images as in section 3.10.1. The segmentation has already been performed by our multi-level segmentation approach (chapter 2), so a two-valued image is given as input for both the greedy and the GVF snake algorithm shown in figure 3.17(a), 3.18(a) and 3.19(a). The choice of the parameters – α, β, γ and a spatial neighborhood of pixels, which limits the maximal movement of a snake point per iteration, for the greedy algorithm [WIL92] and α, β, γ and κ for the GVF snake [XU97, XU02] – is important for the approximation performance of the snake algorithm. In order to have an initialization for the snake, we calculate an enlarged convex hull polygon of the input image for the greedy snake algorithm. Moreover, we interpolate additional snake points between larger line segments of the convex hull polygon. Both the initial contour and the resulting snake are

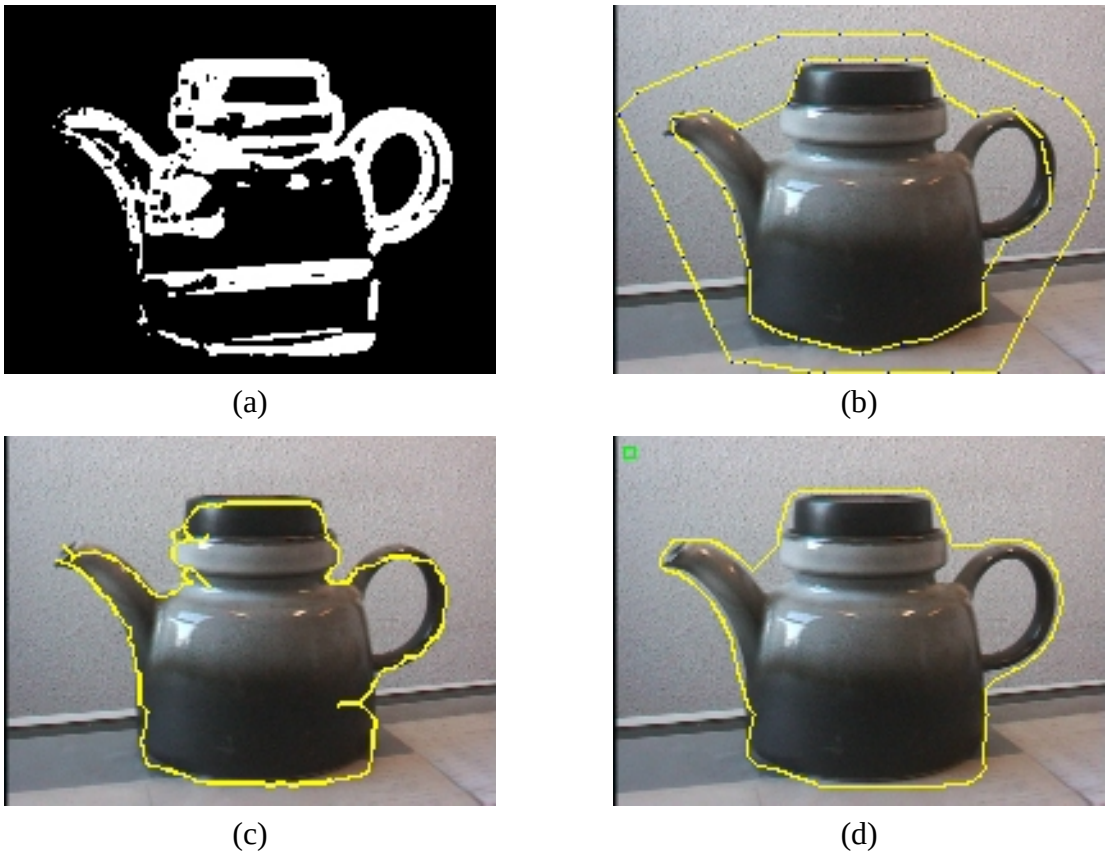


Figure 3.17: Input image in (a). Application of greedy snake algorithm to the tea pot image in (b) and the GVF snake in (c). The result of our approach is shown in (d).

shown superimposed in yellow in the examples, the points of the snake are depicted by blue dots. For the initialization of the GVF snake, we use an ellipsoid which encloses the object. The resulting GVF snake is superimposed on the original image for the comparison.

The results for the first example – the tea pot – are shown in figure 3.17. In (a), the input image for the algorithm is shown, which has been obtained by our multi-level segmentation algorithm. The input image has a size of 192×144 pixels and the initial snake contour consists of 32 points. The application of the greedy snake algorithm with a neighborhood of 11×11 pixels, $\alpha = 0.1$, $\beta = 0.2$, and $\gamma = 1.0$ yields the result shown in (b) after 16 iterations. The application of the GVF snake is illustrated in (c) with $\alpha = 0.05$, $\beta = 0$, $\gamma = 1$ and $\kappa = 0.5$. The contour approximation achieved by our algorithm is shown in (d). As can be seen, the results of the active contour algorithms are quite good for the chosen parameter settings. The greedy snake algorithm in (b) slightly cuts into the object on the right and has problems to approximate the concave parts of the object on the upper left. The GVF snake approximates the concave parts of the tea pot very accurately in (c). Nevertheless, it cuts more clearly and more often into the object. The result of our approach is somewhere in between in (d). Our approximated contour neither cuts into the object, nor is the approximation of the concave part of the tea pot on the upper left and the upper right as exact as the GVF snake in (c).

The second exemplary image – the punch – consists of 384×288 pixels. Again, the greedy snake and the GVF snake algorithm are applied to the input image in figure 3.18(a). For the greedy snake algorithm, the parameters $\alpha = 0.1$, $\beta = 0.1$, $\gamma = 0.8$ yield the contour illustrated

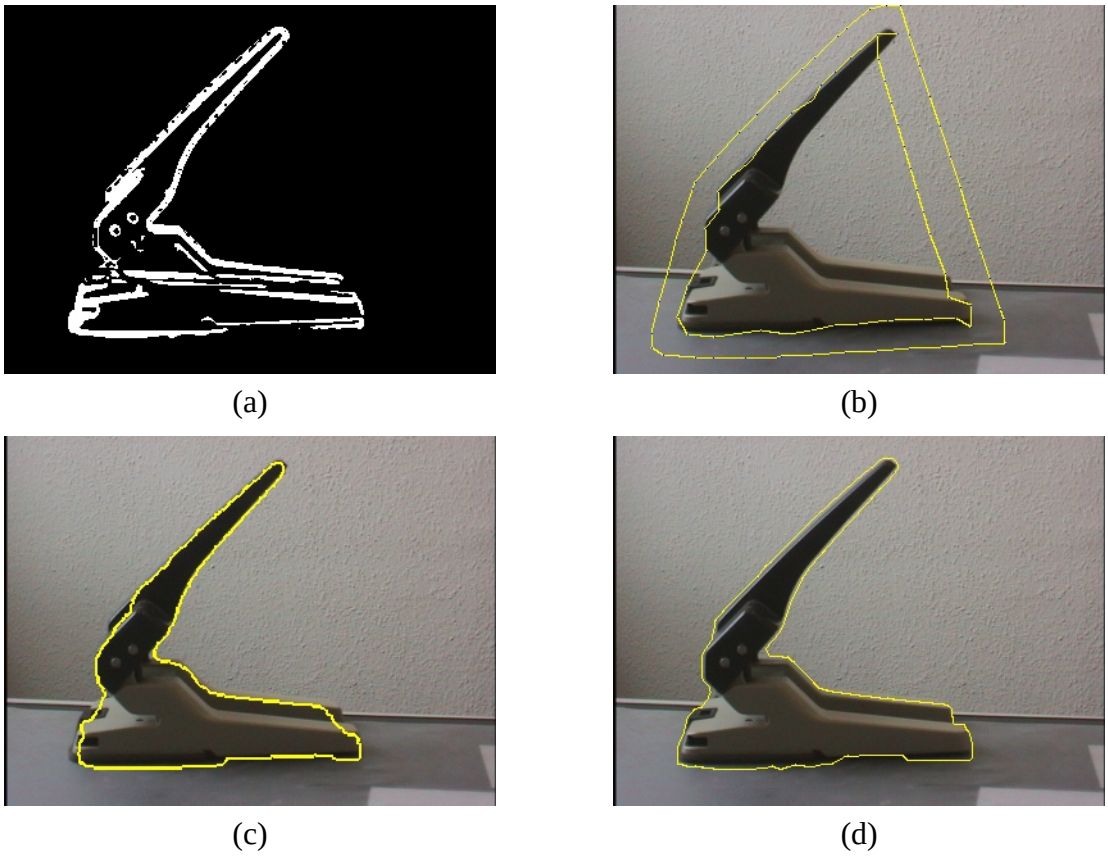


Figure 3.18: Input image in (a). Application of greedy snake algorithm to the tea pot image in (b) and the GVF snake in (c). The result of our approach is shown in (d).

in (b) after 28 iterations. The application of the GVF snake algorithm with $\alpha = 2$, $\beta = 0$, $\gamma = 1$ and $\kappa = 2$ is shown in (c), the result of our approach in (d). As can be seen, the greedy snake algorithm completely fails as far as the approximation of the concave part of the punch is concerned in (b). The GVF snake approximates the punch very well in (c), but cuts slightly into the object on the left and at the bottom right. The results of our approach in (d) are very similar to the GVF snake. Yet, our contour does not cut into the object in (d).

In figure 3.19, the teddy bear is used as input image again. The initial snake contour consists of 49 points. Figure 3.19(b) shows the greedy snake algorithm after 22 iterations, the applied parameters are $\alpha = 0.1$, $\beta = 0.1$, $\gamma = 0.8$ and a neighborhood of 15×15 pixels. In (c), the setting $\alpha = 2$, $\beta = 0$, $\gamma = 1.0$ and $\kappa = 2$ were chosen for the GVF snake algorithm. In comparison to our approach in (d), the results of both the greedy and the GVF snake algorithm are rather similar. The GVF snake performs bettern than the greedy snake as far as the approximation of the concave parts of the teddy bear on the right are concerned. Yet, the GVF snake more clearly cuts into the object on the left. Again, the results of our approach and of the GVF snake are very similar. Once more, our contour approximation does not cut into the object.

Summarizing the results of the comparion of our approach to active contour models, we have shown that our algorithm provides considerably better results than the greedy snake algorithm because of the significant problems of the greedy snake as far as the convergence to concave boundaries is concerned (especially in figure 3.18). This is a well-known problem in

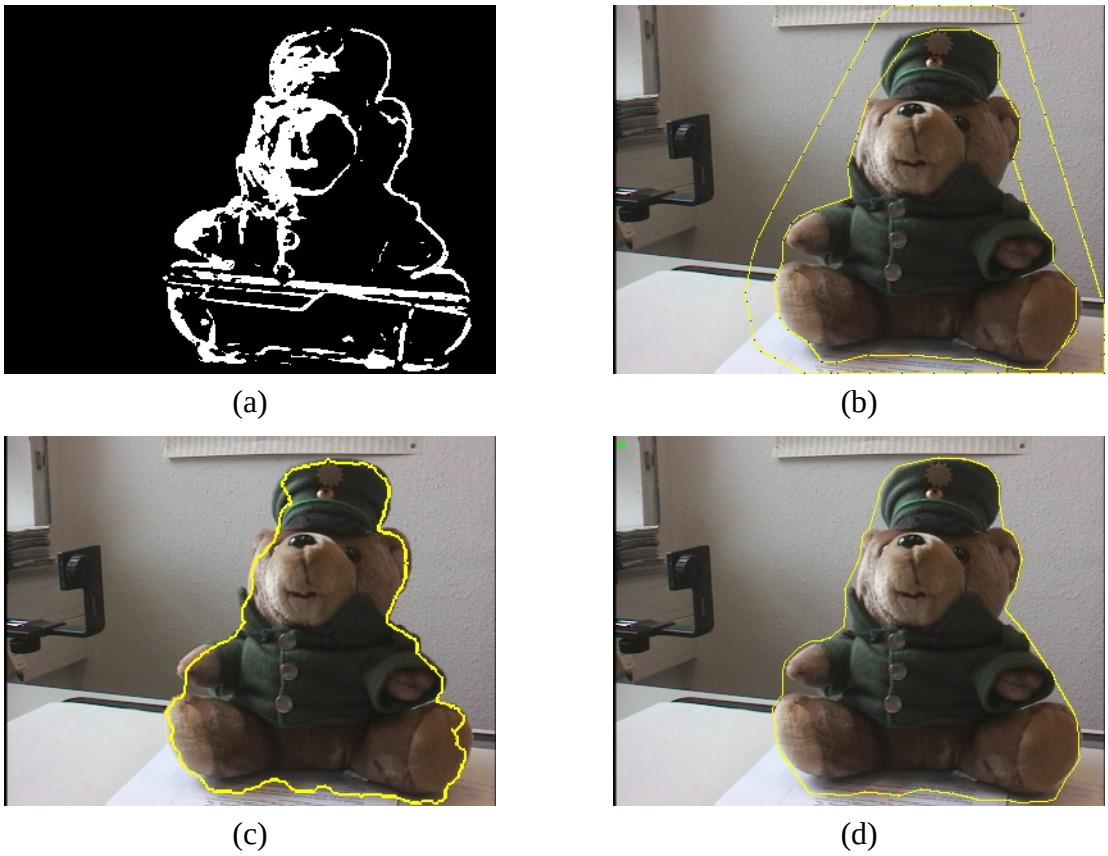


Figure 3.19: Input image in (a). Application of greedy snake algorithm to the tea pot image in (b) and the GVF snake in (c). The result of our approach is shown in (d).

literature and possible solutions and improvements have been discussed, e. g., the gradient vector flow snake [XU97]. The results of the GVF snake and our approximation are very similar as shown in the examples. However, our algorithm provides a contour that does not cut into the object which is the case in every example for both the greedy and the GVF snake. Moreover, the snake algorithms have to be extensively parametrized in order to yield meaningful results. Our algorithm requires just one assumption, namely about the maximum length of possible contour gaps.

However, the snake approach is more flexible than our algorithm. While our approach requires a presegmented set of input data for which an approximated contour is determined, the snake algorithm is applicable on original video images, which may be strongly affected by noise, as well. Misclassified pixels of the segmentation stage, which lead to outliers in the contour approximation of our approach, will presumably not disturb the snake algorithm. Nevertheless, the contour approximations yielded by the snake algorithms are not as geometrically exact as our approach. This seems to be especially disadvantageous, when the input data is a clear set of object pixels as it is the case in our scenario. In order to make our algorithm more robust against sporadic outliers, we are able to incorporate time coherence (see section 3.8) to smooth the approximated contours. Another advantage of active contour models is their ability to profit by any kind of higher level knowledge which may be available in other scenarios of application and which should direct the snakes to better results [COH91]. A comparison of the required calculation time of each of the approaches is difficult because the

greedy snake algorithm provides considerably worse results than our approach and the GVF snake is available as MatLab source code only [MAT02, XU02]. Thus, a direct comparison of the required calculation time of the GVF snake and our algorithm which works in realtime (see section 3.7) is not possible.

As a conclusion, we see the following main advantages of our approach. The parameterization of our approach is considerably simpler and comes across vividly. Snakes have to be extensively parameterized, otherwise the results of the approximation may be poor. Our algorithm is based on geometrical calculations which is a completely different approach to contour approximation. The visual inspection of the results indicates that our algorithm works geometrically more exact than the snake algorithm by not cutting off parts of the objects as far as the assumption about the maximum length of contour gaps is correct. If a clear set of input data is given, geometrically exact results are required, and the results have to be obtained in realtime, our algorithm may be preferable against snake-based algorithms.

Chapter 4

Applications

In order to test and to evaluate the introduced framework for segmentation purposes in computer vision systems, we incorporated the system in different applications. In chapter 4.2, we present an interaction system for a back-projection wall with arm gestures. In chapter 4.3, we integrate our system in the *Zyklop* hand posture recognition system for prelocalization purposes. Finally, we use our framework for a parking lot surveillance in chapter 4.4. While the arm gesture and hand posture recognition are more or less indoor applications, the parking lot surveillance has to cope with outdoor illumination conditions, which are significantly more unstable and, thus, more difficult to cope with.

4.1 Evaluation of Segmentation Performance

As explained in chapter 2.5, it is rather difficult to compare and to evaluate the achievements and benefits of our approach with other state of the art approaches because these are normally not available as open source code. Thus, we demonstrate the capabilities of our multi-level segmentation approach on the basis of typical applications. As an important aspect, we like to emphasize that the parameters of the low-level and mid-level processing stage are the same for all the presented applications. Only the high-level processing stage, which incorporates application-dependent features, is configured using knowledge about the application and the goal of segmentation, for example the expected size of objects. By considering the different types of applications, it becomes evident that the self-learning and adaptation methods contribute enormously to a reliable and stable segmentation system that is moreover comfortable to use.

For each of the applications, we have recorded series of video images enabling us to evaluate the results on a per frame basis. As a measure of quality, we consider the visual inspection of the results of the polygonal contour approximation (see chapter 3). Failures in the multi-level segmentation system will likely lead to insufficient contour approximations, which probably disturb further processing steps on the application level. For instance, there may be wrongly classified pixels, which belong to the background and should not have been found by the segmentation. In this case, the contour approximation most probably will not approximate the contour of the objects sufficiently well. As a consequence, one or more outliers of the approximated contour become obvious. In our point of view, outliers are the main problem as far as failures in the segmentation system are concerned. The aim of the interaction application presented in section 4.2 for instance is to derive the pointing direction of the user's arm.

Outliers of the contour approximation are thus a far more severe problem than a cut off finger or an approximation that slightly cuts into the user's arm as far as this application is concerned. This aspect, of course, is a matter of parameterization. Depending on the subsequent application, a shift of the noise filtering parameters could be advantageous. On the one hand, the objects (in this case the user) are more likely to be completely segmented as a consequence. On the other hand, it has to be taken into account that outliers then become more likely. For the arm gesture application, we consider an avoidance of outliers to be preferable. For the hand posture recognition, it may be more important not to cut off fingers of the segmented hand. However, in order to show the capabilities of our approach, we did not change the parameters for the respective application. Thus, we use the same parameters, which have been mentioned in the respective chapters, for each of the applications. As a consequence, a slight cut into the foreground objects of the contour approximation is not judged as an error of the segmentation system. In chapter 3.8, we have presented a smoothing algorithm for this kind of applications that is able to smooth the determined contour polygons based on time coherence. This algorithm significantly reduces the negative impact of outliers as shown in chapter 3.9. However, in order to evaluate the capabilities of the multi-level segmentation system itself, we do not apply the smoothing algorithm here. In cases of permanent changes, we will evaluate the capabilities of our algorithm to recognize and to adapt to such a situation and, moreover, to recall respective knowledge, when a formerly learned situation reoccurs.

4.2 Interaction with Arm Gestures

As a first example of application which employs the segmentation framework, a computer-vision-based interaction system is presented that enables natural interaction with a back-projection wall.

4.2.1 Introduction

Video projection is in widespread use for multimedial presentations in classrooms and at conferences. It also plays an important role in group meetings for visualization purposes. Usually, interaction is performed at a standard keyboard/mouse computer the screen content of which is additionally directed to a video beamer. This type of interaction limits the possibilities of group meetings because the interaction has to be performed at the computer, although it would be more natural to interact directly at the back-projection wall. For that purpose, special displays augmented with sensors have been developed such as the SmartBoard [STR94]. Another recent development is to use classical laser pointers by capturing the laser point on the projection screen by video cameras. Versions for front- and back-projection have been implemented based on that idea [KIR98, WIS01].

Another step further is the user pointing directly to the projection wall, without any additional pointing tool. The idea is to observe the user with video cameras in order to recognize his arm and to calculate the three-dimensional pointing direction of the arm. There has been and is an increasing number of projects concerning tracking of the human body such as the *Pfinder* and *Spfinder* of the MIT [AZA96, WRE95] for human-computer interaction. Most of the projects emphasize on the recognition of symbolic static or motion gestures while the precise determination of locations or directions is less often treated. Examples concerning

pointing are *Visualization Space* and *Dream Space* by IBM [IBMDS, IBMVS]. These systems and others define the pointing direction by a line connecting the head and the hand of the pointing arm. According to our experience, the postures that have to be performed to hit the desired goal are somewhat unnatural and thus inconvenient. Another approach is to use a computer-internal kinematic 3D-model of the human body, which is controlled by the user by mapping body features recognized in the camera images to features of the model. Then the arm direction of the model can be taken as the desired pointing direction [HOC99]. The advantage of the model-based approach is its robustness. Its precision, however, still depends on the precision of the 3D-information acquired from the images.

One basic idea of our solution is to define a slightly restricted scenario of pointing to the wall. In our opinion, this scenario is still natural and does not restrict the user too much. Our system uses two standard video cameras, under the ceiling and sidewards, which observe the space in front of the back-projection wall. The user directly interacts with the graphical user interface by pointing with his stretched arm to the wall. The arm has to be in a defined region of about 1 to 1.5 meters of depth in front of the wall. Further dynamic objects such as other parts of the body of the user are not allowed in this region. Typically, the cursor of the application is displayed at the intersection of the pointing line with the wall and can freely be moved by moving the arm. Further instructions, for example initiating a mouse button click, can be given by natural voice commands using a wireless headset microphone.

4.2.2 Usage of Segmentation Framework

In order to detect the presence and the position of the user and especially the arm of the user, the segmentation system is utilized. After the determination of the pointing arm within the video images, further processing steps have to be performed. For the transformation into three-dimensional world coordinates, a camera calibration technique has to be applied and the pointing direction has to be calculated. These steps are described in detail in [LEU01]. The segmentation framework is utilized within the interaction system to perform the first processing steps, namely to work out contour pixels of the user. We recorded different series of images in front of different back-projection walls. The first and the second series were recorded in front of the back-projection wall that is shown in figure 4.1.

All the series were recorded in an indoor environment. Nevertheless, the first and the second scenario are affected by large windows which are causing varying illumination conditions. Moreover, shadows are casted by further persons in the room, which do not appear in front of the camera. On the one hand, these conditions are unfavorable. But on the other hand, their impact is realistic as far as actual applications are concerned. The third scenario is recorded in a mere indoor environment without any windows.

4.2.3 First Scenario

The first series contains 1162 images with a size of 192×144 pixels. The initial learning phase was performed with 80 images. For the mid-level processing stage, the image was subdivided in 12×9 box areas. The high-level processing stage was adjusted to extract exactly one object with the minimum size of five boxes. Some exemplary video images of both cameras, sidewards and under the ceiling, are shown in figure 4.2.

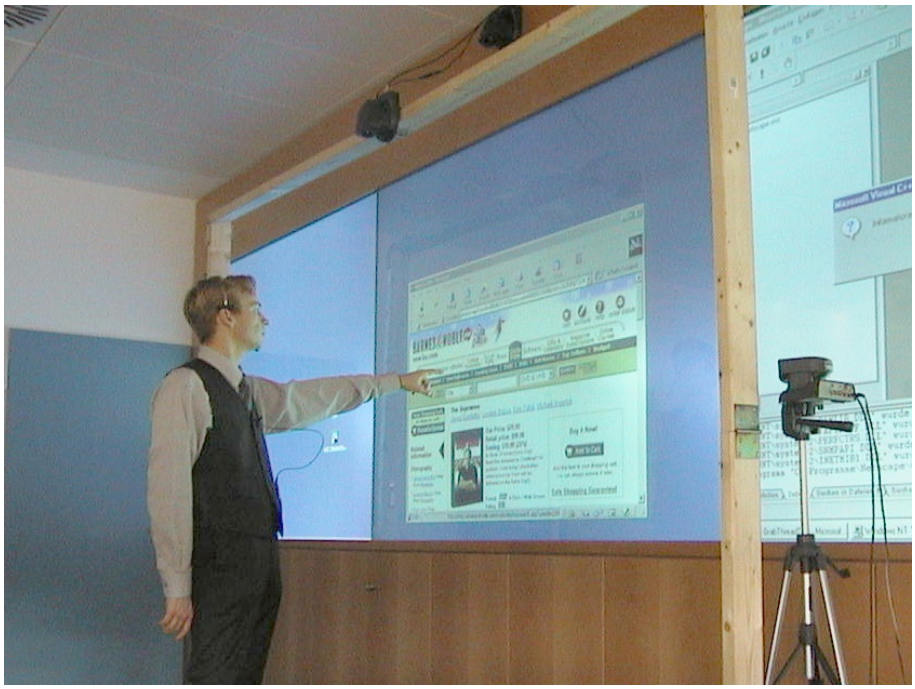


Figure 4.1: User interacting with back-projection wall. Two cameras observe the arm of the user. The mouse cursor is moved to the position the user is aiming at.

(a) Sidewards camera

(b) Ceiling camera

Figure 4.2: Exemplary video images of first scenario.

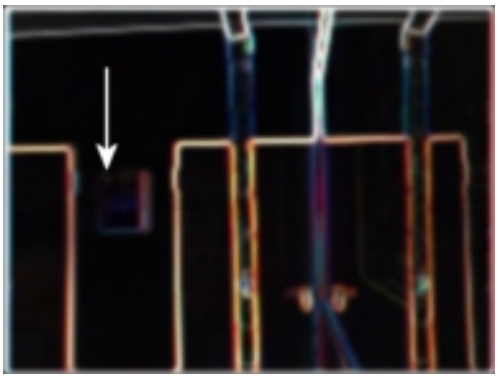
After the initial learning phase, the segmentation system works perfectly and provides good results for two different users. A typical example is illustrated in figure 4.3. After 605 frames, the electric light is switched off leading to a considerable change in the illumination conditions, while a user is interacting with the system. As a consequence, the left boundary of the picture frame on the left becomes visible. The situation is illustrated in figure 4.4.

However, when the user's arm is not near the picture frame, the mid- and high-level processing stage correctly classify the area as "background", so the changes in the color and edge values are learned into the low-level knowledge bases quickly. Altogether, we counted nine frames as "false", whereas the example in figure 4.4 is one of the worst results. Moreover, we could not find any further unfavorable effects which are related to the illumination change. In

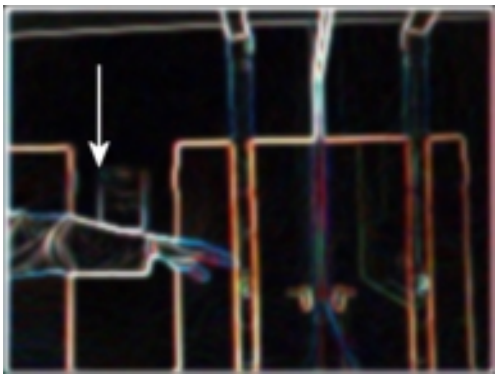
(a) Result of segmentation

(b) Result of contour approximation (blue)

Figure 4.3: Excellent results after the initial learning phase.

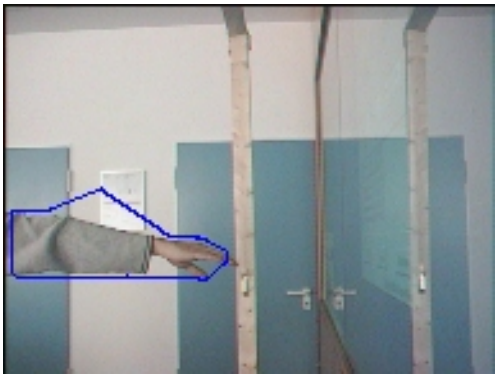


(a) Average edge values in edge knowledge base



(b) Edges in current image

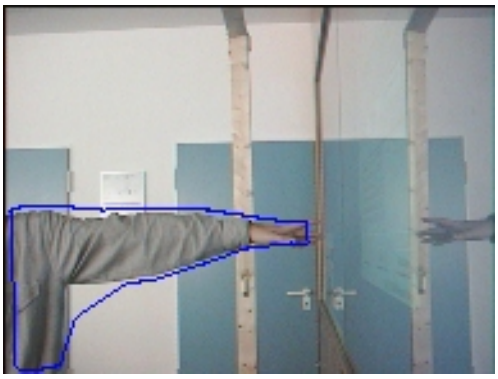
(c) Result of segmentation



(d) Faulty contour approximation (blue)

Figure 4.4: After switching off the electric light, the left boundary of the picture frame becomes visible, indicated by arrows in (a) and (b). The differences are detected by the low-level segmentation in (c) and lead to a faulty contour approximation in (d).

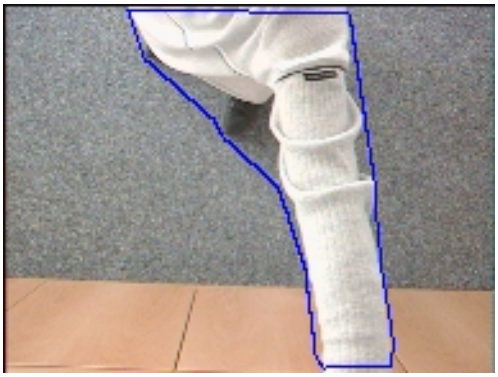
(a) Result of segmentation



(b) Result of contour approximation (blue)

Figure 4.5: Excellent results after automatically adapting to changed background situation.

(a) Result of segmentation



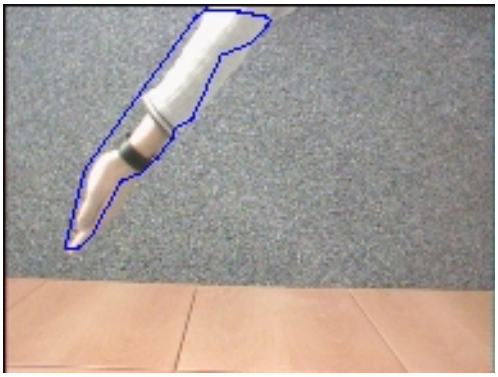
(b) Result of contour approximation (blue)

Figure 4.6: Perfect segmentation results for the images of the ceiling camera.

chapter 3.8, we have presented a smoothing algorithm, which is able to compensate the sudden occurrence of outliers based on time coherence. We have applied the presented algorithm to this situation in chapter 3.8. In figure 3.14 and figure 3.15 on the pages 91 and 92 the results of the smoothing algorithm are illustrated. As can be seen, the quality of the results can be improved significantly by the smoothing algorithm. However, we do not take into consideration the smoothed results for this evaluation in order to show the basic capabilities of the introduced segmentation approach. After the changed background is visible for a short time, the system adapts to the changes, so the results are good again, illustrated in figure 4.5.

As an overall result, there were nine out of 1082 images (or 0.8%) – the images of the initial learning phase are not counted – the results of which were not sufficient at all, leading to a correctness rate of 99.2%. The results for the ceiling camera were even better, the segmentation delivered very good results for the entire series. The segmentation system did not produce any outlier during the whole series leading to very good results of the contour approximation. An example of application is shown in figure 4.6. However, in some images the contour approximation cuts slightly into the arm of the user, as illustrated in figure 4.7. In the sense of our correctness definition, we do not count these inaccuracies as errors because the subsequent application is not affected. The more or less smooth background seems to be advantageous as far as the varying illumination conditions are concerned. Thus, the correctness ratio is 100%.

(a) Result of segmentation



(b) Result of contour approximation (blue)

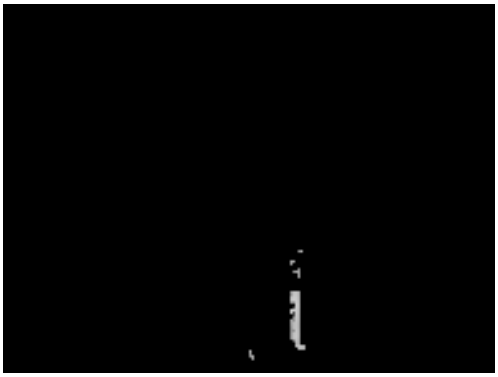
Figure 4.7: In some images, the contour approximation cuts slightly into the user's arm. The subsequent application is not affected by such inaccuracies.

4.2.4 Second Scenario

The second series of images was recorded in front of the same back-projection wall. This series consists of 3782 images, 80 images are employed for the initial learning phase. Again, a sidewards camera and one under the ceiling is employed to observe the arm gestures of the user (compare figure 4.1). However, the setup was slightly changed. As a consequence, the back-projection wall is not seen by the sidewards camera anymore. The scenario is slightly more complicated because further persons in the room sometimes cast shadows that lead to changes in the background. Sometimes, these changes are significant enough to lead to wrong classification results as for instance illustrated in figure 4.8.

While the segmentation worked reliably most of the time, in rare cases failures occured. For instance, a couple of pixels classified wrongly lead to an outlier of the contour approximation in figure 4.9. Evaluating the whole series of 3702 images, we counted 41 images or 1.1% with poor segmentation results (as for example shown in figure 4.9). In seven images or 0.19% of the series a non-existing object was provided by the segmentation. Counting both of these effects averages out an error rate of 1.3%. Consequently, for 98.7% of the entire series the segmentation provided good results as for instance shown in figure 4.10. Again, the results of the ceiling camera are slightly better because of the favorable background. A typical example of application is shown in figure 4.11.

During the series, a chair is moved into the scene. The chair is not seen by the sidewars camera because it is beneath the camera. The permanent change of the background is recognized by the segmentation system. Afterwards, the chair does not affect the system's performance. An exemplary image is shown in figure 4.12. The user's arm above the chair is segmented correctly. The segmentation works reliably, both for initially learned background content and adaptively learned further content. Some time later, the chair is moved slightly to the left. Immediately, the former background content is recognized at the old position of the chair, avoiding faulty segmentation results. The chair is not recognized as permanent background change as long as the user is moving close to the chair because both are recognized as one object (see figure 4.13). The permanent change at the new position of the chair is recognized when the distance between the user and the chair is large enough for a short while. Then, the segmentation provides good results both at the old and at the new position of the chair as shown in figure 4.14.

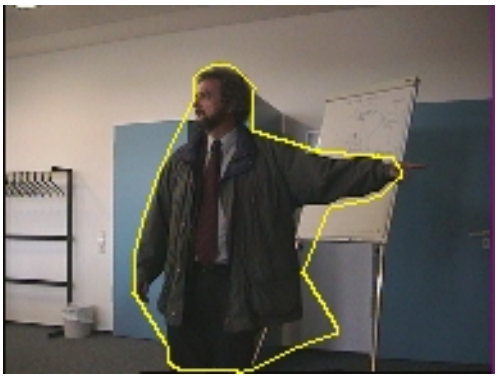


(a) Result of segmentation

(b) Result of contour approximation (yellow)

Figure 4.8: In rare cases, casted shadows are recognized wrongly as object.

(a) Result of segmentation



(b) Result of contour approximation (yellow)

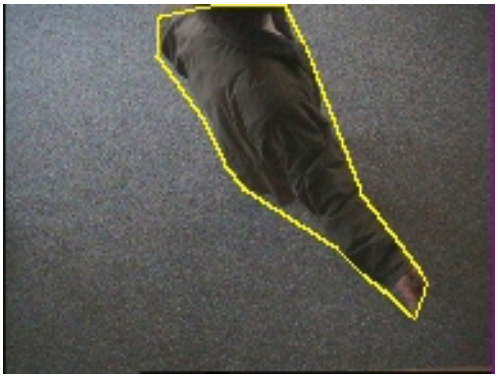
Figure 4.9: A couple of pixels classified wrongly lead to an outlier on the right of the user.

(a) Result of segmentation

(b) Result of contour approximation (yellow)

Figure 4.10: For 98.7% of the series the segmentation provides good results.

(a) Result of segmentation



(b) Result of contour approximation (yellow)

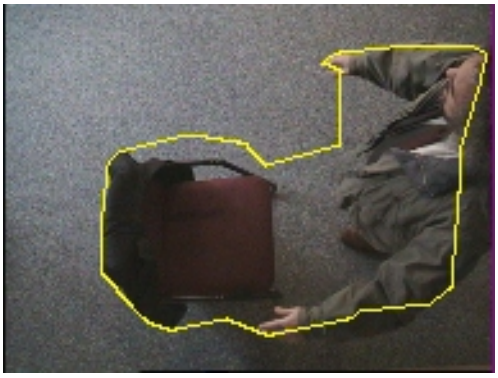
Figure 4.11: The segmentation works well, even though the contrast is weak because of a dimmed illumination.

(a) Result of segmentation

(b) Result of contour approximation (yellow)

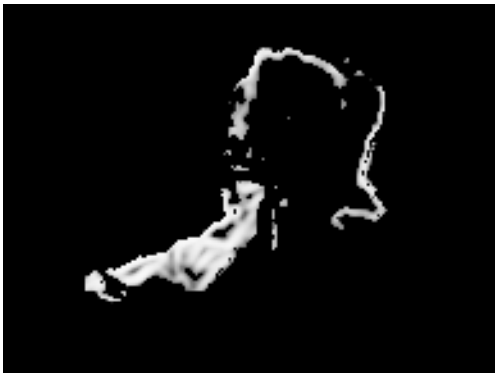
Figure 4.12: The chair which is moved into the scene is recognized as permanent background change. The segmentation performance is not affected.

(a) Result of segmentation

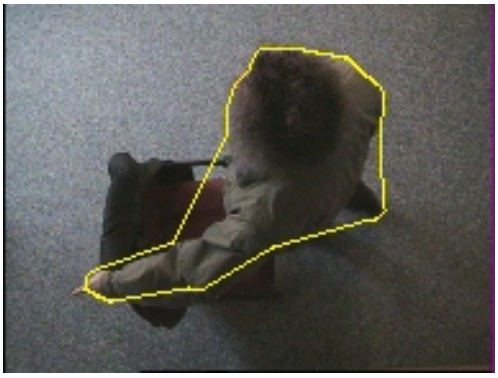


(b) Result of contour approximation (yellow)

Figure 4.13: The chair is not recognized as permanent background change as long as the user is moving near the chair and both are recognized as one object.



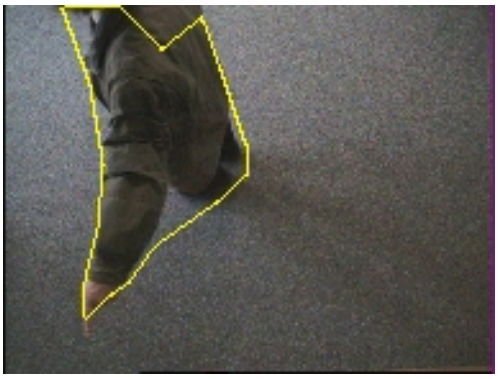
(a) Result of segmentation



(b) Result of contour approximation (yellow)

Figure 4.14: After the recognition of the permanent change at the new position, the segmentation provides good results, both at the old and at the new position of the chair.

(a) Result of segmentation



(b) Result of contour approximation (yellow)

Figure 4.15: The chair is moved out of the scene. The former background content is recognized leading to continuously good results of the segmentation.

Finally, the chair is moved out of the scene. Again, the former situation is recognized immediately, leading to continuously good results of the segmentation as shown in figure 4.15. Although shadows are casted visibly on the floor, there are no sharp edges. Thus, outliers rarely occur. Nevertheless, there were 26 images of the ceiling camera with an imperfect contour approximation, averaging out an error rate of 0.7% of all images or a correctness rate of 99.3%.

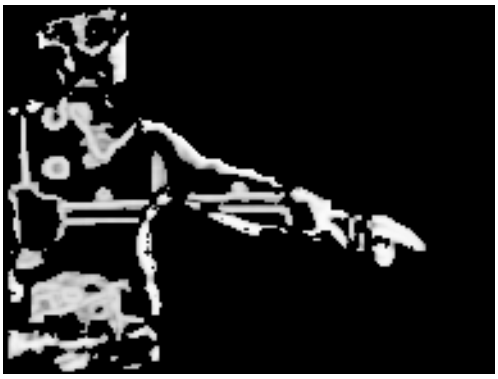
Figure 4.16: Back-projection wall in a mere indoor environment without any windows.

4.2.5 Third Scenario

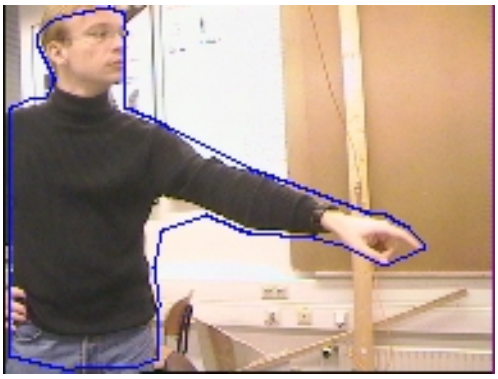
The third series was recorded in front of another back-projection wall and consists of 1288 images. The back-projection wall is located in a mere indoor environment without any windows. The scenario is shown in figure 4.16.

The segmentation results for the sidewards camera do not contain any fault. As expected, the indoor environment provides better basic conditions, 100% of the series of the sidewards camera were segmented correctly. A typical example of segmentation is shown in figure 4.17. The results for the ceiling camera are also very good, a typical example is shown in figure 4.18. The markers on the floor do not have anything to do with our application. They are utilized for calibration purposes of a magnetic motion tracking device. We did not remove the markers for our system in order to show that the segmentation is able to cope with arbitrary background situations.

During the series, not seen by the sidewards camera, a chair is moved into the scene, illustrated in figure 4.19. The chair is segmented as foreground object for a short while. Because of its static character, the permanent background change is recognized by the pattern recognition system. New low-level color and edge knowledge bases are added for the box areas containing the chair. Afterwards, the segmentation is not affected by the chair and provides good results for the user's arm above the chair as shown in figure 4.20. When the chair is moved out of the scene again, the boxes which show the former background content completely are immediately classified as "background". This effect is illustrated in figure 4.21. During the series, we counted 13 images with minor outliers of the kind illustrated in figure 4.22, leading to an error rate of 1.1% or consequently to a correctness rate of 98.9%.



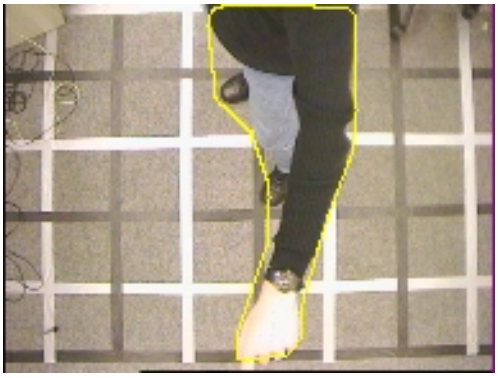
(a) Result of segmentation



(b) Result of contour approximation (blue)

Figure 4.17: The results of the sidewards camera do not contain any fault.

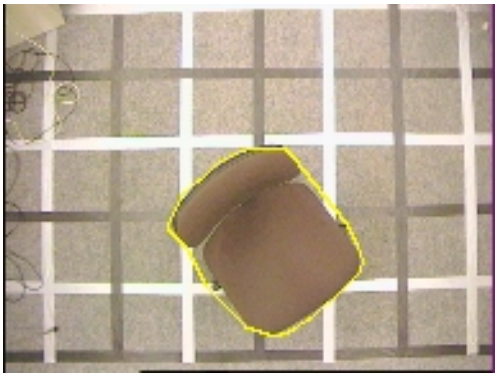
(a) Result of segmentation



(b) Result of contour approximation (yellow)

Figure 4.18: The results of the ceiling camera are also very good.

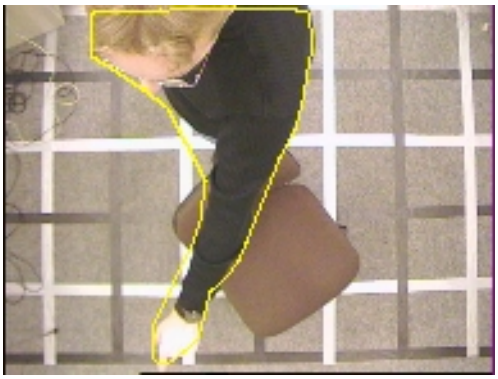
(a) Result of segmentation



(b) Result of contour approximation (yellow)

Figure 4.19: The chair is recognized correctly as foreground object for a short while.

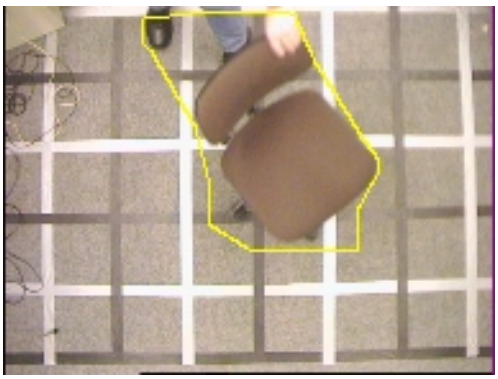
(a) Result of segmentation



(b) Result of contour approximation (yellow)

Figure 4.20: After a short while, the chair is recognized as permanent background change, the segmentation results are not affected.

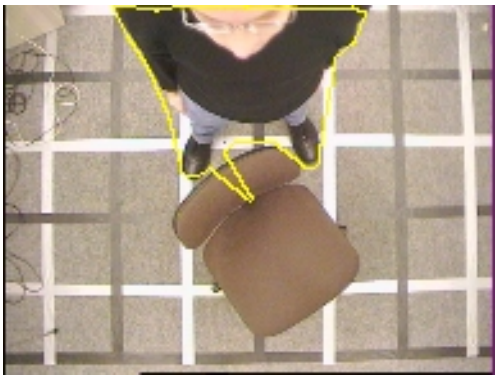
(a) Result of segmentation



(b) Result of contour approximation (yellow)

Figure 4.21: When the chair is moved out of the scene again, the former background is recognized immediately for these boxes which show the former background completely.

(a) Result of segmentation



(b) Result of contour approximation (yellow)

Figure 4.22: In rare cases, the contour approximation contains minor outliers.

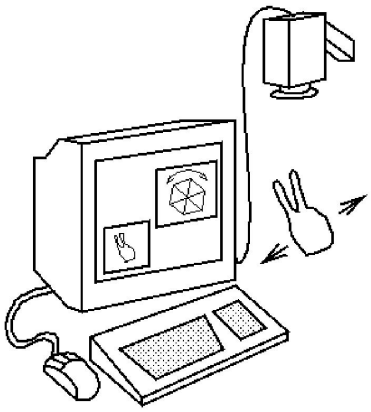


Figure 4.23: Typical interaction environment for the hand gesture recognition system *Zyklop*.

4.3 Zyklop - Hand Posture and Gesture Recognition

As a further example of application, we integrated the introduced multi-level segmentation approach into the existing hand posture and gesture recognition system *Zyklop*, which has been presented and extensively discussed in [KOH01].

4.3.1 Introduction

The hand posture recognition in *Zyklop* has been realized as a computer-vision-based approach. Typically, a video camera is mounted above a table and observes a locally restricted area on the table in which hand postures can be performed. The system is able to recognize previously trained hand postures reliably, while the training itself is a rather convenient process. The hand postures which are intended to be recognized by *Zyklop* have to be shown to the system only a few times. A typical scenario for the application of *Zyklop* is illustrated in figure 4.23.

On the application side, *Zyklop* has been utilized successfully to control, for example, the visualization software *GeomView* [GEO01] or the internet browser software *Netscape Navigator* [NET01]. Moreover, in the *Home AOM* project [HOM00], which was funded by the European community, *Zyklop* has been integrated into a framework of multi-modal interaction software that enables easy communication and interaction with typical household devices such as television and video cassette recorders, washing machines, or shutters for elderly or disabled people. Figure 4.24 shows a typical set of hand postures which can be used for the interaction with *Zyklop*.

4.3.2 Segmentation in *Zyklop*

In a first release of *Zyklop* [STA95], the segmentation was realized histogram-based with a global threshold. The video images are acquired from the camera in the YUV color model, which is customary in Germany. The image consists of three planes of color information: The Y-channel contains the lightness, color is represented in a two-dimensional coordinate system with the axes U (blue-yellow) and V (red-green). Due to the fact that skin color contains

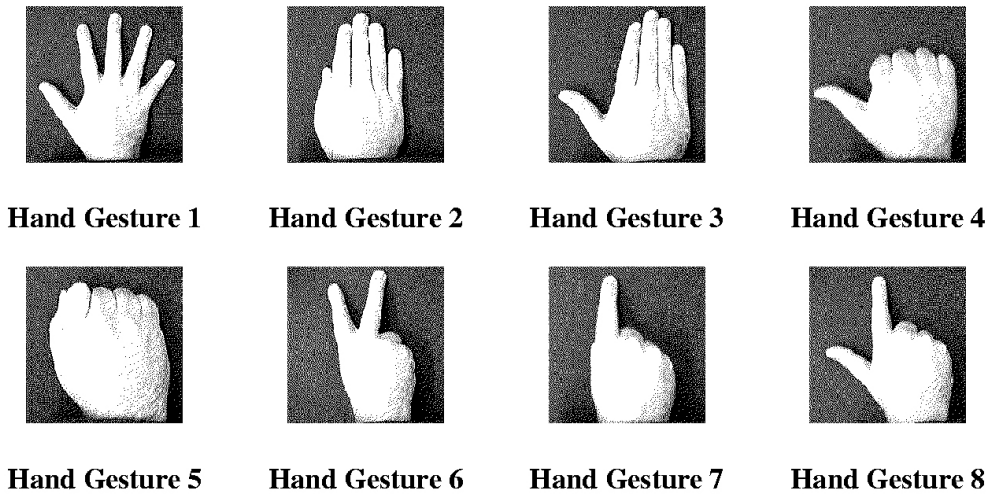


Figure 4.24: Typical set of hand postures which may be used for the interaction with *Zyklop*.

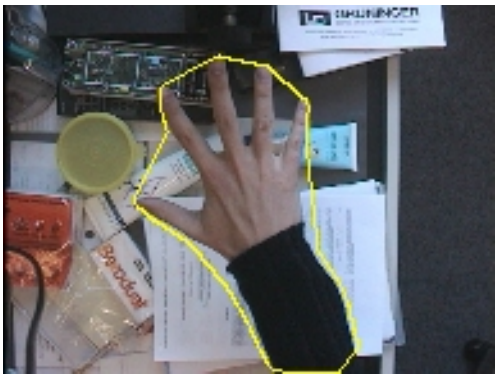
a significant amount of red and that within the YUV color model the U- and V-channel are theoretically independent of lightness, the V-channel was chosen as a basis for segmentation. Typically, a uniformly colored background whose colors are significantly different in comparison to skin color is chosen. Then, the histogram of the V-channel shows two significant peaks, one for the background regions and one for skin color. The local minimum between both peaks is chosen as threshold for the separation between background and human skin color. Based on the determined skin regions, further processing steps are performed in order to obtain, for example, contour information of the hand.

Basically, this segmentation approach works sufficiently well for a lot of situations. However, the segmentation fails considerably under certain circumstances, especially when for example a window is somewhere near the cameras or the illumination is not evenly spread across the interaction area. The failing of the segmentation stage is especially annoying because the recognition of hand postures itself works very reliably. In order to test and to evaluate our multi-level segmentation system, we supplemented the histogram- and threshold-based segmentation method in *Zyklop* by our approach. For this purpose, we perform a prelocalization of the user's hand in the interaction environment. In the prelocated image areas, as provided by our multi-level segmentation approach, the histogram-based search for skin color is performed. Moreover, integrating our approach allows for the interaction in arbitrarily composed surroundings. The existing *Zyklop* is normally applied with a uniformly colored background in order to avoid disturbances which may result from background colors that are similar to skin color. Applying our segmentation method for prelocalization purposes eliminates this problem.

4.3.3 First Scenario

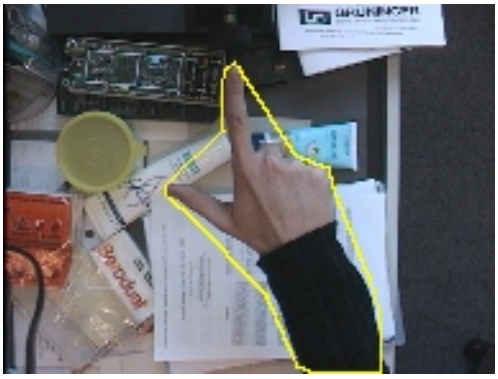
In the first scenario, the camera watches a table desk that is covered with a lot of different things. The interaction area is located directly near a window. Nevertheless, the segmentation worked very well for the first series, which consists of 547 images, and did not produce any outlier. Typical examples of application are shown in figure 4.25.

(a) Result of segmentation



(b) Result of contour approximation (yellow)

(a) Result of segmentation



(b) Result of contour approximation (yellow)

Figure 4.25: The prelocalization of the user's hand works very well for the entire series.

However, in some images a finger is cut off by the contour approximation. This may lead to classification failures in the *Zyklop* system. Because of the fact that in a subsequent processing stage skin color is identified in the images, it seems to be reasonable to change the parameterization of our system. As a consequence, the prelocalization could be performed more sensitively. This may negatively influence the occurrence of outliers as far as the contour approximation is concerned. However, outliers would not harm the overall result in this context because the final aim of segmentation, i. e., the user's hand, is detected independently within the approximated contour. Moreover, the finger tips are rather small in the video image, which means that they do not differ significantly from some parts of the background. Thus, the limitations of a background subtraction approach may be reached in this scenario.

A typical example of an inaccurate contour approximation is shown in figure 4.26. Two fingers are cut off by the contour approximation. With respect to our primary goal of avoiding outliers of the approximated contour, explained in section 4.1, we do not count these effects as errors. The visual inspection of the results has shown that outliers of the contour approximation do not occur during the test series.

4.3.4 Second Scenario

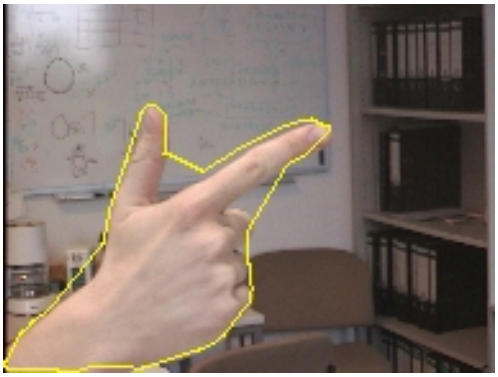
For the second scenario, the camera is oriented into the corner of the office. The user can perform hand gestures above his office desk towards the camera. The recognition performance is very good, insufficient contour approximations do not occur during the 1261 images of the

(a) Result of segmentation

(b) Result of contour approximation (yellow)

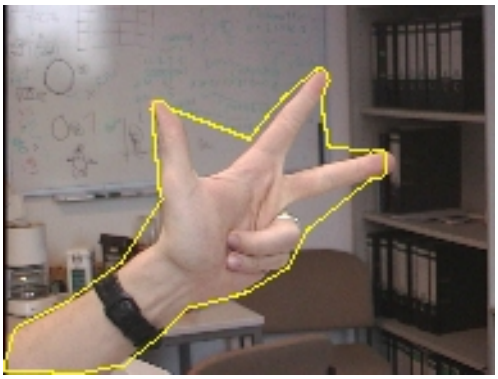
Figure 4.26: In rare cases, finger tips may be cut off by the contour approximation.

(a) Result of segmentation



(b) Result of contour approximation (yellow)

(c) Result of segmentation



(d) Result of contour approximation (yellow)

Figure 4.27: The segmentation worked perfectly for the second scenario.

series. Thus, we have a correctness rate of 100%. Some typical examples of application are shown in figure 4.27.

4.4 Parking Lot Surveillance

While the first and the second example are examples of indoor applications, we have successfully integrated our multi-level segmentation approach into an outdoor application that observes a parking lot as well. Parking lot surveillance is a common application in the field of computer vision, e. g, [IVA98, RIG00]. Outdoor applications are normally considered to be more difficult than indoor applications because changing and varying illumination conditions are more severe. For instance, moving clouds and suddenly appearing or disappearing sun light may considerably affect the segmentation process.

The goal of the parking lot surveillance is to observe the parking lot and to save images to the hard disk that contain one or more foreground objects. The determined foreground objects are moreover depicted by an approximated contour. For this example of application, the system has run for approximately 67 hours, including two nights, without being influenced manually. When the system was started, during the initial learning phase, the parking lot was quite full. Consequently, these cars have been learned as background and their disappearance leads to the recognition of foreground objects at their former position for a short while. Afterwards, the permanent change is detected and the actual background is learned.

After running the system for approximately 67 hours, we evaluated the results of the recorded video images. First of all, it can be summarized that only images showing significant changes have been saved. None of the images has been recorded because of changing illumination conditions. The system was able to smoothly cope with every illumination condition, which may have happened during this time. Due to the adaptive learning mechanisms, even the night time was completed successfully. During the nights, some lanterns have been switched on. As a consequence, the previously totally dark images suddenly contain ”white disks”, which are, of course, recognized as foreground objects for a short while until they are classified as permanent changes (see figure 4.29(b)). We do not judge this behavior as erroneous because the sudden appearance of these lights in an absolutely dark environment is a severe change that is detected correctly by the algorithm.

Because of the lack of comparable surveillance video tapes, we cannot ensure that the system has really detected every object that may have passed the scene. However, by comparing the recorded images from frame to frame, we can prove that the system did not miss a disappearing or newly arriving car. Such a failure would have become obvious, when, for instance, in succeeding recorded frames suddenly a car is missing. As a consequence, the system would not have recorded the car while leaving. From this point of view, the system worked perfectly and detected every appearing and leaving car. Moreover, the examples indicate that even smaller objects were recognized sufficiently well. For instance, in figure 4.29(d) even a small cat or rabbit is recognized and tracked correctly.

In order to illustrate and to demonstrate the capabilities of our system, some of the recorded images are shown in figure 4.28 and 4.29. During the 67 hours of operation, the system recorded 17433 images. Summarizing the results, the system was still running very well after 67 hours (see figure 4.29(f)). Changing and varying illumination conditions did not cause any wrong classification result, underlining the importance of the self-learning, adaptive approach of our system. Moreover, the system obviously did not miss any significant change on the parking lot. Again, a manual adjustment of parameters has not been performed. The parking lot surveillance was realized using the same parameter settings that have been used in the

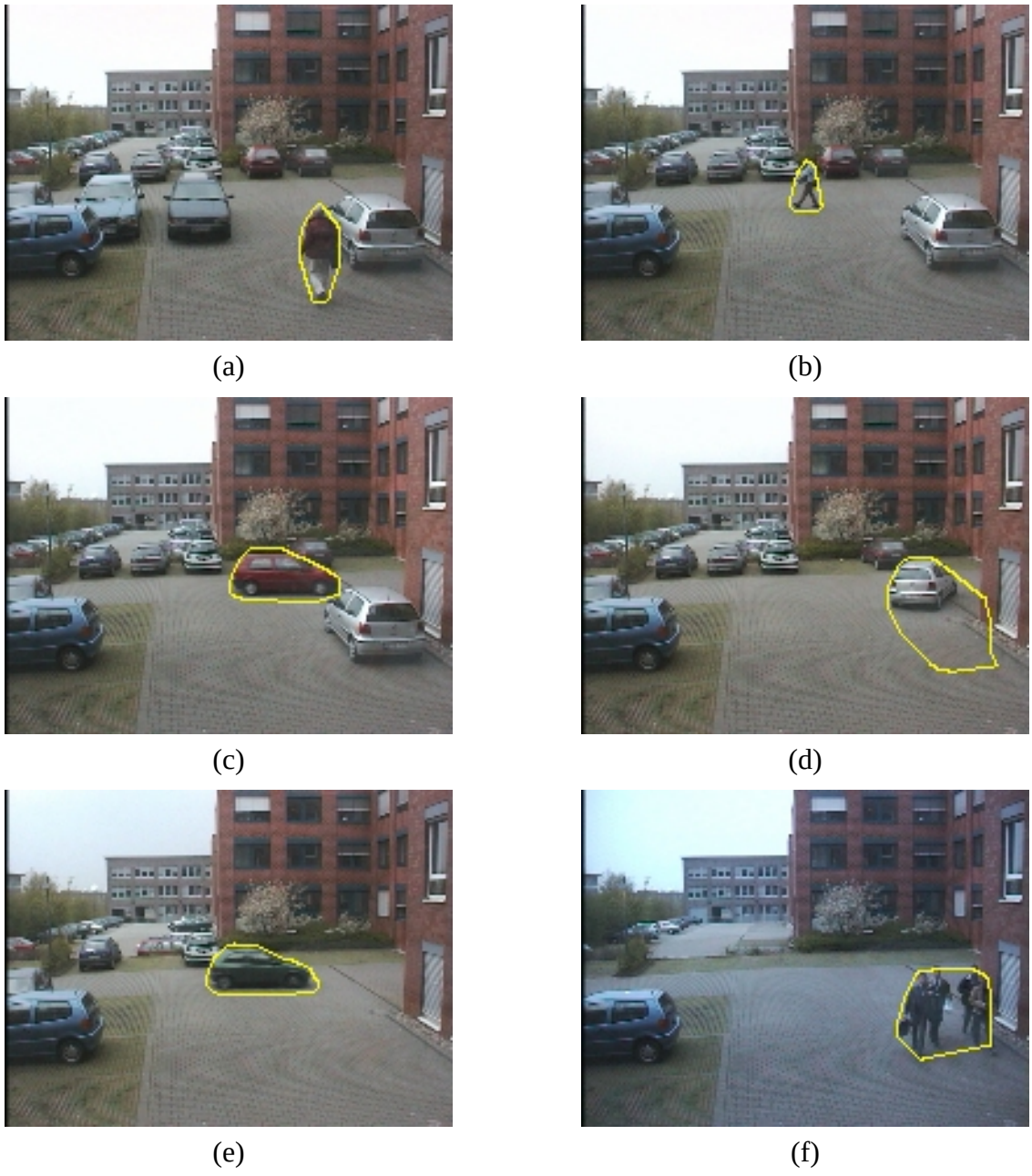


Figure 4.28: Examples of parking lot surveillance. In (a) and (b), two persons are tracked on the parking lot, in (c) a leaving car is shown. In (d), the disappearance of the car is detected as object for a short while because the car was learned as background during the initial learning phase. In (e), another car is leaving. Figure (f) shows a group of people on the parking lot.

other examples of application, indicating a general applicability of the empirically determined parameter settings.

Evaluating the recorded images, we have found 210 images with outliers in the contour approximation, leading to a correctness rate of 98.8% of the recorded images. Assuming an average frame rate of 10 frames per second, the system was altogether applied to approximately 2.4 million video images during the 67 hours, leading to an overall correctness rate of about 99.99%.

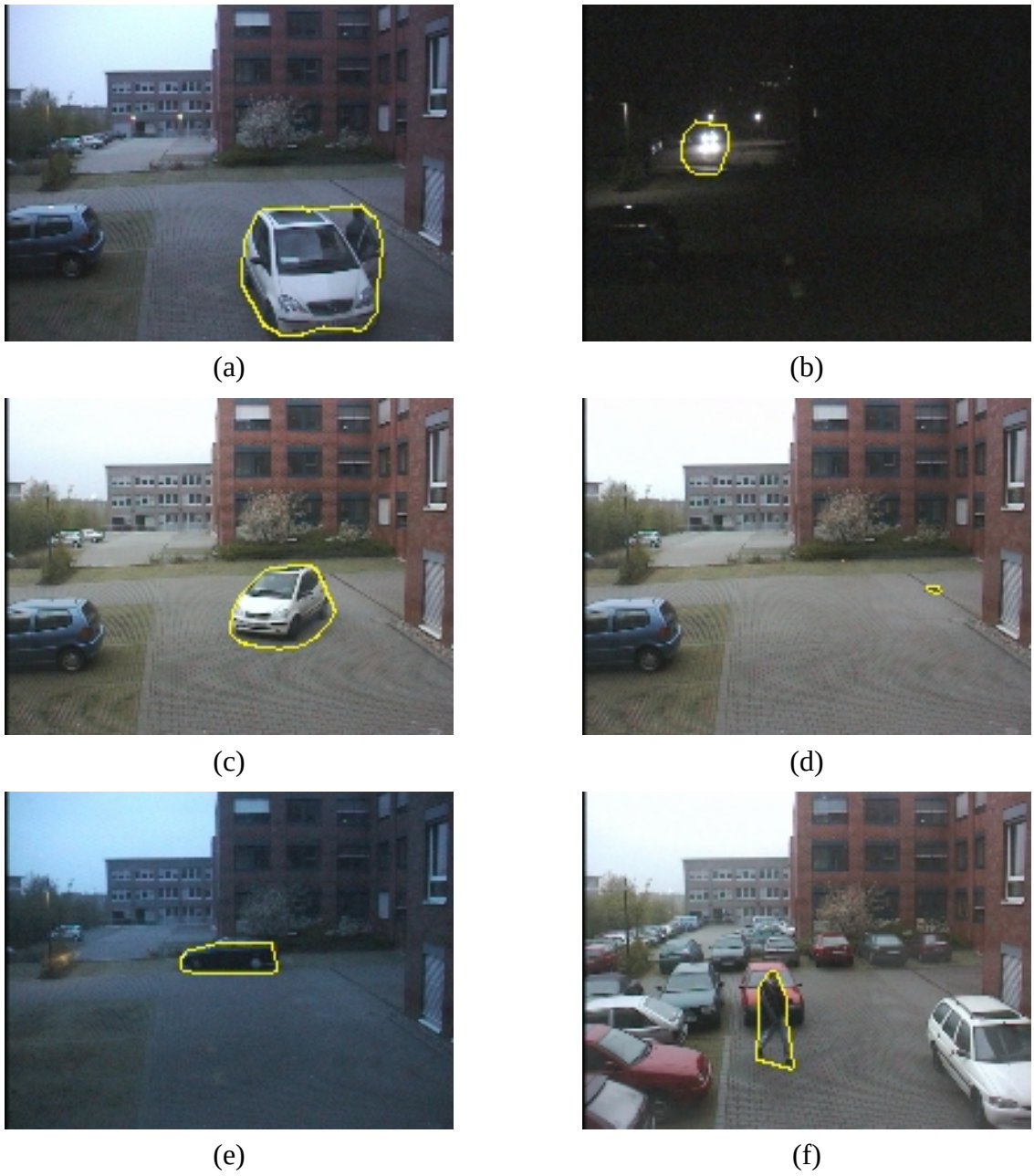


Figure 4.29: Examples of parking lot surveillance. Figure (a) shows the parking lot at 20:30 in the evening and it is already quite dark. However, the car of the security service is segmented correctly. In (b), the sudden appearance of the light source is recognized as an object. Figure (c) shows the second day at 11 o'clock in the morning, the system performs well after running during the night time. Figure (d) shows the sensitivity of the segmentation system that correctly tracks a little cat or rabbit on the parking lot. Figure (e) shows the third day, early in the morning when the first car arrives. Finally, the parking lot is full again in (f), each car has been observed successfully during its appearance on the parking lot.

More important than the correctness rate is in our opinion the stability the system has shown as far as this special application is concerned. The system was not affected by any change in the illumination conditions for about 67 hours including three nights. Typical changes of the illumination conditions are sunrise, dawn, moving clouds, or total darkness. Moreover, the segmentation performance after this long time of operation was just as good as immediately after the initial learning phase, proving the effectiveness of the adaptive learning and feedback mechanisms.

4.5 Summary of Application Results

We have presented three different types of application in which we successfully integrated both the multi-level segmentation approach and the polygonal contour approximation algorithm. The first and the second application can be characterized as indoor applications for interaction purposes, while the third application is a typical outdoor application. As we have shown for the different scenarios, our segmentation system works sufficiently well and reliably in every environment. Due to the automatic adaption to permanent background changes and the continuous adaption to slight changes in the background, the system is able to cope with any possible situation. Even in the worst case, i. e., when the background changes completely, the system is able to recover on its own and to work reliably again. The only condition is a more or less static background for a short while. This is the basic assumption for the application of our system, as explained in chapter 1.4.

The correctness rate of the presented applications and their different scenarios with respect to the interpretation of correctness as explained in section 4.1 is very good. For the first application, i. e., interaction with arm gestures, we examined three different scenarios with two different camera views. Summarizing the results of the first application, the correctness rates are listed in table 4.1.

For the second application, the *Zyklop* hand posture recognition system, we found a correctness rate of 100% in two scenarios. The parking lot surveillance system is obviously the most difficult scenario as far as the amount and frequency of changing illumination in such an outdoor application is concerned. However, our segmentation method performed very well. Based on the total amount of processed video images, we achieved a correctness rate of 99.99%. Based on the recorded images, the correctness rate is still 98.8%. Especially the parking lot surveillance has impressively proven that even the long operating time in an outdoor environment with all the illumination changes involved during nearly three succeeding days and nights did not disturb our system at any time. On the contrary, the performance of the system after all this time was at least as good as immediately after the initial learning phase, underlining the effectiveness and stability of the introduced segmentation approach.

Interaction with Arm Gestures

Correctness Rates	sidewards camera	ceiling camera
First Scenario	99.2 %	100 %
Second Scenario	98.7 %	99.3 %
Third Scenario	100 %	98.9 %

Table 4.1: Correctness rates of the first application

Chapter 5

Summary and Conclusion

We have presented a multi-level segmentation system that is based on background subtraction. For the low-level processing stage, which works pixelwise, we have introduced an improved color- and edge-based segmentation method that performs considerably better than the well-known standard background subtraction methods as shown in chapter 2.5.2. Due to the incorporation of fuzzy logic methodologies and the possibility to store and maintain a large amount of color knowledge for each pixel separately, the low-level processing stage is able to cope smoothly with varying illumination conditions. The mid-level processing stage brings in an advanced pattern recognition system the features of which are intended to be insensitive to changing illumination conditions. Moreover, the mid-level processing stage allows for the administration of different background image content, and recognizes permanent changes in the background automatically. It is also able to recall background knowledge, when former situations reoccur. The high-level processing stage incorporates application-dependent knowledge and identifies those objects which fulfill these requirements. An important aspect of the system are internal feedback mechanisms which allow for a continuous adaptation of parameters and a learning of current background content. In chapter 2.5, we have presented evaluation results that show significantly good results. Because of the lack of objective comparison possibilities to other state of the art methods, we have tried to show the universal applicability, the reliability of the segmentation performance, and the overall quality of our system in different examples of application. Although the scenarios in which our system was employed were very different – both indoor and outdoor applications – the parameter setting was not changed. Due to the self-learning and adaptive approach, the segmentation system performed very well in all of these applications.

Moreover, we have introduced a new approach to polygonal contour approximation in chapter 3, which extends the capabilities of the convex hull contour approximation. By this means, concave parts of objects can be approximated sufficiently well. Due to this flexible contour approximation approach, we are able to derive a closed contour from the determined object pixels, which are provided by the multi-level segmentation approach. Examples of application and a performance analysis of the contour approximation method have been presented in the chapters 3.7, 3.9, and 4. Compared to other state of the art methods, our approach yields more or less comparable results, which are geometrically more exact. Moreover, our algorithm works in realtime and its parameterization comes across vividly (see chapter 3.10).

Summarizing the presented results, the introduced segmentation approach is an almost universally applicable and powerful method for segmentation tasks, which principally allow background subtraction. As we have shown in chapter 1.2, most systems have a couple of

restrictions as far as the scenario of application is concerned. We think that our system eliminates or at least reduces a lot of these restrictions considerably. The required scenario for our system has been described in detail in chapter 1.4.

Future work may be dedicated to the development of further mid-level similarity measures, which may improve the classification results of this stage. As far as the high-level processing stage is concerned, a more refined tracking component could be integrated that incorporates additional knowledge about the behaviour of the expected objects, such as model-based assumptions about their movement or appearance. Due to the multi-level structure of the system, extensions can easily be incorporated.

Appendix A

A Brief Course In Fuzzy Logic

Fuzzy logic or fuzzy techniques are often employed in this contribution in order to cope with uncertainty and for decision-making. Thus, some basic notions are introduced in this section. Comprehensive introductions can be found for example in [KRU93, ZIM87]. For more specific introductions to decision processes see for example [ZAD75, NEG85].

Fuzzy logic - initiated by Lotfi A. Zadeh in 1965 [ZAD65] - is basically a multi-valued logic allowing intermediate values between conventional evaluations like *true / false*. The concept of linguistic terms and variables enables the processing of human notions such as ”pretty warm” or ”slightly cold” on computers. Observations and measurements often cannot be utilized to strictly distinguish between two classes. For example, the notion ”warm” is difficult to define. Some people may consider a temperature of above 25 Centigrade as ”warm”. The application of a threshold will lead to strange results, if for example the temperature is 24.9 Centigrades and is considered as ”cold”, while a temperature of 25.0 degrees Celsius is considered as ”warm”. Obviously, the same holds for the application of thresholds in the context of image processing. Where is the borderline between slight changes that we do not want to recognize (for instance because of varying illumination) and between severe changes we do want to determine (because an object of interest has entered the region under consideration)? The original statement of Bellman and Zadeh in 1970 about the role of fuzzy sets in decision analysis was the following and is especially interesting as far as the role of probability theory in contrast to fuzzy logic (or in this context possibility theory) is concerned:

> ”Much of decision-making in the real world takes place in an environment in which the goals, the constraints and the consequences of possible actions are not known precisely. To deal quantitatively with imprecision, we usually employ the concepts and techniques of probability theory and, more particularly, the tools provided by decision theory, control theory and information theory. In so doing, we are tacitly accepting the premise that imprecision – whatever its nature – can be equated with randomness. This, in our view, is a questionable assumption. Specifically, our contention is that there is a need for differentiation between *randomness and fuzziness*, with the latter being a major source of imprecision in many decision processes. By fuzziness, we mean a type of imprecision which is associated with *fuzzy sets*, that is, classes in which there is no sharp transition from membership to nonmembership. For example, the class of *green objects* is a fuzzy set. So are the classes of objects characterized by such commonly used adjectives as large, small, significant, important, serious, simple, accurate, approximate, etc.

> Actually, in sharp contrast to the notion of a class or a set in mathematics, most of the classes in the real world do not have crisp boundaries which separate those objects which belong to a class from those which do not. In this connection, it is important to note that, in the discourse between humans, fuzzy statements such as "John is *several* inches taller than Jim," "x is *much larger* than y," "Corporation X has a *bright future*," "the stock market has suffered a *sharp decline*," convey information despite the imprecision of the italicized words..." (Bellman and Zadeh in [BEL70], p. 141)

A.1 Basic Notions and Definitions

The very basic notion of a fuzzy system is the *fuzzy set*:

Definition A.1.1 (Fuzzy Set) *:*

Let U be a universe of discourse. Then a fuzzy set *A of U is a function*

$$\mu_A : U \longrightarrow [0, 1]. \tag{A.1}$$

μ_A is the membership function *that maps the universe U to the unit interval $[0, 1]$ and $\mu_A(x)$ is the degree of membership of $x \in U$.*

In classical set theory the membership of elements $x \in U$ can be defined as a function

$$U \longrightarrow \{0, 1\} \tag{A.2}$$

so elements $x \in U$ either belong or do not belong to a set. Fuzzy set theory extends this definition to degrees of membership. In order to cope with human notions, fuzzy sets often are designed to reflect linguistic terms such as "warm", "fast" or "young". Due to the close relationship to set theory in general, basic operations for sets such as *union* or *intersection* can be adapted to fuzzy set theory. Zadeh in 1965 [ZAD65] suggested the following concepts, which are, however, not the only possible way for the extension of classical set theory. For more detailed introductions into this topic see e. g. [KRU93, ZIM85].

Definition A.1.2 (Intersection Operation) *:*

Let μ_A and μ_B be membership functions of the fuzzy sets A and B. Then the membership function μ_C of the intersection *of A and B is determined as*

$$\mu_C(x) = \min(\mu_A(x),\ \mu_B(x)), \quad \forall x \in U. \tag{A.3}$$

The set theoretic intersection can be interpreted as a logical "and" operator.

Definition A.1.3 (Union Operation) *:*

Let μ_A and μ_B be membership functions of the fuzzy sets A and B. Then the membership function μ_C of the union *of A and B is determined as*

$$\mu_C(x) = \max(\mu_A(x),\ \mu_B(x)), \quad \forall x \in U. \tag{A.4}$$

The set theoretic union can be interpreted as a logical "or" operator.

Definition A.1.4 (Complement Operation) *:*

Let μ_A be a membership function of the fuzzy set A. Then the membership function μ_B of the complement *of A is determined as*

$$\mu_B(x) = 1 \; - \; \mu_A(x), \quad \forall x \in U. \tag{A.5}$$

The set theoretic complement can be interpreted as a logical "not" operator.

These basic operators are used within the multi-level segmentation system.

A.2 Approximate Reasoning

An important field of application for fuzzy logic is approximate reasoning, which can be employed for instance in control systems [MAM75]. The basic idea is to use the concept of linguistic variables and the methodology of fuzzy set theory to allow the evaluation of linguistic statements, which are often combined with a *rule base*. The main tools of reasoning in classical logic are tautologies such as the *modus ponens*:

$$(A \; \Rightarrow \; (A \; \Rightarrow \; B)) \; \Rightarrow \; B \tag{A.6}$$

This rule can be interpreted in the way that there is the premise "A is true", the implication "If A then B", and the conclusion "B is true". Zadeh in 1973 [ZAD73] proposed two generalizations of the modus ponens:

1. to allow statements which are characterized by fuzzy sets.
2. to relax the identity of "B" in the implication and the conclusion.

These modifications are usually referred to as the "generalized modus ponens" [ZAD73, ZIM85]. An example of application may be:

Premise	:	The meal is rather salty.
Implication	:	If the meal is salty then the eater becomes thirsty.
Conclusion	:	The eater is rather thirsty.

Zadeh proposed the *compositional rule of inference* for this type of fuzzy conditional inference, whereas the most popular approach is the min-max compositional rule of inference. An important aspect of Zadeh's approach is the realization of the fuzzy implication function using the $\min$-function. Decisions are made considering specified rule bases, which consist of IF-THEN statements and linguistic variables. The evaluation of fuzzy rule bases usually consists of four steps:

1. **Fuzzification**: Measured values are mapped onto fuzzy sets.
2. **Inference**: Implication rules are interpreted.
3. **Aggregation**: The results of the different rules are combined.
4. **Defuzzification**: The fuzzy result is mapped onto a crisp result value.

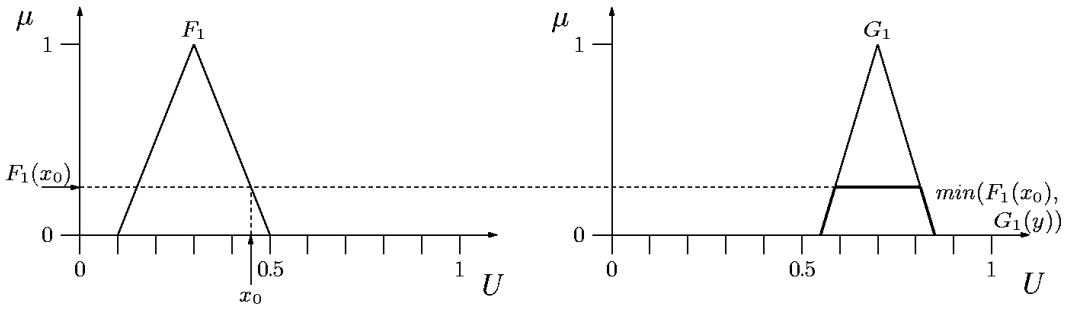


Figure A.1: Fuzzification of input value x_0 and implication with $\min$-function.

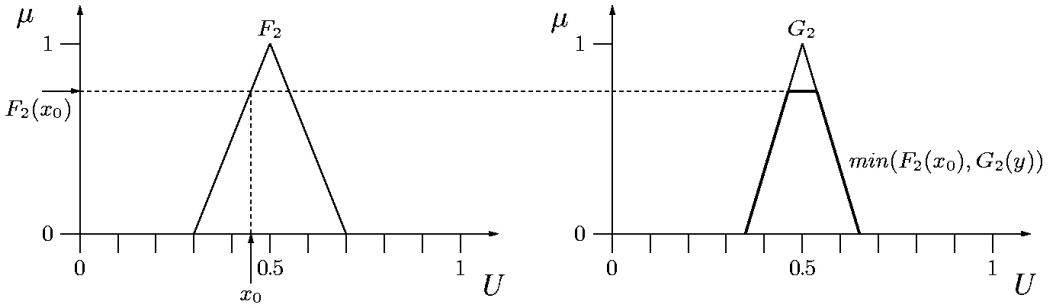


Figure A.2: Evaluation of second rule.

A very simple example of a rule-base consisting of two rules could be:

IF F_1 THEN G_1
IF F_2 THEN G_2

In figure A.1 and A.2, a measured input value x_0 is fuzzified and the implication is realized using the $\min$-function. The resulting fuzzy sets are shown in figure A.3, the result of the aggregation using the $\max$-function in figure A.4. For the defuzzification, the *center of area* (COA) method is often used, in which the centroid of the aggregated fuzzy set (figure A.4) is determined and mapped onto the universe (on the x-axis) of the G_i.

As already mentioned, there are a lot of ways, how fuzzy logic operators as an extension of two-valued logic can be realized. Another possibility to interpret fuzzy rule bases would

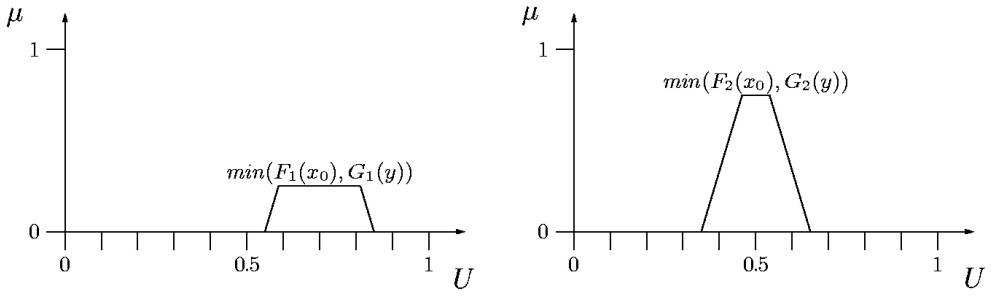


Figure A.3: Resulting fuzzy sets.

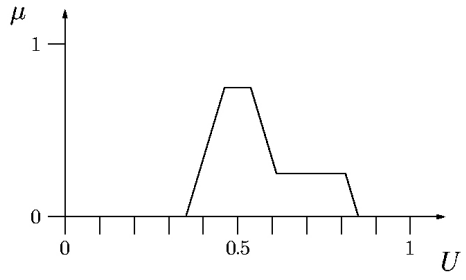


Figure A.4: Result of aggregation using the max-function.

be for example the realization of the implication as multiplication or the aggregation as sum. For comprehensive and extensive introductions and explanations see for example [ZAD65, ZAD73, ZAD75, ZIM85, ZIM87, KRU93].

Bibliography

[AHO82] A. Aho, J. Hopcroft and J. Ullman, *Data Structures and Algorithms*, Addison-Wesley, 1982.

[AZA96] A. Azarbayejani, C. R. Wren, and A. P. Pentland, *Real-Time 3-D Tracking of the Human Body*, In: IMAGE'COM 96, Bordeaux, France, 1996.

[BAL82] D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice-Hall, 1982.

[BEL70] R. E. Bellman and L. A. Zadeh, *Decison-making in a Fuzzy Environment*, Management Science 17(4), pp. 141-164, 1970.

[BEZ99] J. C. Bezdek (ed.), J. Keller, R. Krishnapuram and M. Pal, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, Kluwer Academic Publishing, 1999.

[BIC94] M. Bichsel, *Segmenting Simply Connected Moving Objects in a Static Scene*, IEEE Pattern Analysis and Machine Intelligence Vol. 16, 1994.

[CAN86] J. F. Canny, *A Computational Approach to Edge Detection*, IEEE Pattern Analysis and Machine Intelligence 8(6), pp. 679-698, 1986.

[COH91] L. D. Cohen, *On Active Contour Models and Balloons*, In: Computer Vision, Graphics, and Image Processing: Image Understanding 53(2), pp. 211-218, 1991.

[COX95] G. S. Cox, *Review - Template Matching and Measures of Match in Image Processing*, Technical Report, Dept. of Electrical Engineering, University of Cape Town, 1995.

[DIG99] V. Di Gesu and V. Starovoitov, *Distance-based Functions for Image Comparison*, Pattern Recognition Letters 20, pp. 207-214, 1999.

[DUB93] M.-P. Dubuisson and A. K. Jain, *Contour Extraction of Moving Objects in Complex Outdoor Scenes*, International Journal of Computer Vision 14, pp. 83-105, 1995.

[DUD00] R. O. Duda, D. G. Stork, and P. E. Hart, *Pattern Classification*, Second Edition, John Wiley and Sons, 2000.

[DUD76] S. A. Dudani, *Region Extraction Using Boundary Following*, In: C. H. Chen (Ed.), "Pattern Recognition and Artificial Intelligence", pp. 216-232, Academic Press, 1976.

[DUY01] P. Duygulu and F. Y. Vural, *Multi-Level Image Segmentation and Object Representation for Content Based Image Retrieval*, SPIE Electronic Imaging 2001, Storage and Retrieval for Media Databases, 2001.

[FOL94] J. Foley et al., *Introduction to Computer Graphics*, Addison-Wesley Publishing, 1994.

[GEO01] Geomview, Interactive 3D Viewing Program for Unix, http://www.geomview.org, 2001.

[GON92] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley Publishing, 1992.

[HAN90] E. R. Hancock and J. Kittler, *Edge-labeling Using Dictionary-based Relaxation*, IEEE Pattern Analysis and Machine Intelligence 12(2), pp. 165-181, 1990.

[HAN78] A. R. Hanson and E. M. Riseman, *Segmentation of Natural Scenes*, In: A. R. Hanson, and E. M. Riseman (Eds.), "Computer Vision Systems", pp. 129-164, Academic Press, 1978.

[HAR85] R. M. Haralick and L. G. Shapiro, *Survey: Image Segmentation Techniques*, Computer Vision, Graphics, and Image Processing 29, pp. 100-132, 1985.

[HAR98] I. Haritaoglu, D. Harwood, and L. S. Davis, W^4*: Who? When? Where? What? - A Real Time System for Detecting and Tracking People*, In: International Conference on Automatic Face and Gesture Recognition, 1998.

[HOC99] M. Hoch, *Intuitive Interface - A New Computer Environment for Design in Visual Media (In German)*, Verlag infix, 1999.

[HOM00] Home AOM Project, funded by European Community, successfully finished in 2000.

[HOU62] P. V. Hough, *Method and Means for Recognizing Complex Patterns*, U.S. Patent 3069654, Dec. 1962.

[HUA97] T. Huang and Y. Rui, *Image Retrieval: Past, Present, and Future*, In: International Symposium on Multimedia Information Processing, 1997.

[IBMDS] IBM. *Dream Space*, http://www.research.ibm.com/natural/dreamspace/

[IBMVS] IBM. *Visualization Space*, http://www.research.ibm.com/imaging/vizspace.html

[IVA98] Y. Ivanov, C. Stauffer, A. Bobick, and W. E. L. Grimson, *Video Surveillance of Interactions*, Proc. of. Computer Vision and Pattern Recognition (CVPR) Workshop on Visual Surveillance, 1998.

[JAI96] A. K. Jain and A. Vailaya, *Image Retrieval using Color and Shape*, Pattern Recognition, Vol. 29, No. 8, pp. 1233-1244, 1996.

[KAN94] A. Kandel and G. Langholz (Eds.), *Fuzzy Control Systems*, CRC Press, 1994.

[KAS88] M. Kass, A. Witkin, and D. Terzopoulos, *Snakes: Active Contour Models*, International Journal of Computer Vision, pp. 321-331, 1988.

[KIL94] M. Kilger, *Architekturkonzept eines adaptiven Systems am Beispiel eines intelligenten videobasierten Verkehrssensors*, Ph.D. Thesis in Computer Sciences, University of Dortmund, 1994.

[KIR98] C. Kirstein and H. Müller, *Interaction with a Projection Screen Using a Camera-Tracked Laser Pointer*, In: Proceedings of The International Conference on Multimedia Modeling (MMM'98), IEEE Computer Society Press, 1998.

[KOH01] Kohler, M., *New Contributions to Vision-based Human-Computer Interaction in Local and Global Environments*, Infix-Verlag, 1999.

[KUB98] T. Kubota and T. Huntsberger, *Adaptive Pattern Recognition System for Scene Segmentation*, Optical Engineering 37(3), pp. 829-835, 1998.

[KUN87] A. Kundu and S. K. Mitra, *A New Algorithm for Image Edge Extraction Using a Statistical Classifier Approach*, IEEE Pattern Analysis and Machine Intelligence 9(4), pp. 569-577, 1987.

[KRU93] R. Kruse, J. Gebhardt, and F. Klawonn, *Fuzzy-Systeme*, Teubner Verlag, 1993.

[LAI94] K. F. Lai, *Deformable Contours: Modeling, Extraction, Detection and Classification*, Doctor Thesis, University of Wisconsin-Madison, 1994.

[LEE98] T. S. Lee, D. Mumford, R. Romero, and V. A. F. Lamme, *The Role of the Primary Visual Cortex in Higher Level Vision*, Vision Research 38, pp. 2429-2454, 1998.

[LEU01] C. Leubner, C. Brockmann, and H. Müller, *Computer-Vision-based Human-Computer Interaction with a Back-Projection Wall Using Arm Gestures*, Proceedings of 27th Euromicro Conference, Warsaw, IEEE Press, 2001.

[MAM75] E. H. Mamdani and S. Assilian, *An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller*, In: International Journal of Man-Machine Studies 7, pp. 1-13, 1975.

[MAR72] A. Martelli, *Edge Detection Using Heuristic Search Methods*, Computer Graphics and Image Processing 1, pp. 169-182, 1972.

[MAR82] D. Marr, *Vision*, W. H. Freeman and Company, 1982.

[MAT02] MatLab, Mathematical Computing and Visualiuation Software, http://www.mathworks.com/products/matlab, 2002.

[MCC86] J. L. McClelland and D. E. Rumelhart, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 2, Cambridge MIT Press, 1986.

[MOE01] T. B. Moeslund and E. Granum, *A Survey of Computer Vision-Based Human Motion Capture*, In: Computer Vision and Image Understanding 81(3), pp. 231-268, 2001.

[MPEG4] MPEG-4 Proposal Package Description (PPD) - Revision 3 (Tokyo Revision), ISO/IEC JTC1/SC29/WG11 Document N0998, July 1995.

[NAK98] A. Nakazawa, H. Kato, and S. Inokuchi, *Human Tracking Using Distributed Video Systems*, In: International Conference on Pattern Recognition, 1998.

[NEG85] C. V. Negoita, *Expert Systems and Fuzzy Systems*, The Benjamin/Cummings Publishing Company, 1985.

[NER98] A. Neri, S. Colonnese, G. Russo, and P. Talone, *Automatic Moving Object and Background Separation*, Signal Processing 66, pp. 219-232, 1998.

[NET01] Netscape Navigator, Web Browser Software, http://www.netscape.com, 2001.

[NIL82] N. J. Nilsson, *Principles of Artificial Intelligence*, Springer Verlag, 1982.

[PAL93] N. R. Pal and S. K. Pal, *A Review on Image Segmentation Techniques*, Pattern Recognition Vol. 26, No. 9, pp. 1277-1294, 1993.

[PAV77] T. Pavlidis, *Structural Pattern Recognition*, Springer Verlag, 1977.

[OCV02] Open Computer Vision Library, http://www.intel.com/research/mrl/research/opencv, 2002.

[PRE85] F. P. Preparata, *Computational Geometry*, Springer-Verlag, 1985.

[PUZ99] J. Puzicha, T. Hofmann, and J. M. Buhmann, *Histogram Clustering for Unsupervised Segmentation and Image Retrieval*, Pattern Recognition Letters 20, pp. 899-909, 1999.

[RES02] *The Research Index*, http://www.researchindex.com, Internet Search Engine for Academical Researchers, 2002.

[RIG00] G. Rigoll, S. Eickeler, and I. K. Yalcin, *Performance of the Duisburg Statistical Object Tracker on Test Data for PETS2000*, IEEE Workshop on Performance Evaluation of Tracking and Surveillance (PETS), pp. 1-7, 2000.

[ROS76] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, *Scene Labelling by Relaxation Opeartions*, IEEE Transactions on Systems, Man and Cybernetics 6, pp. 420-433, 1976.

[RUI99] Y. Rui, T. Huang, and S. Chang, *Image Retrieval: Current Techniques, Promising Directions and Open Issues*, Journal of Visual Communication and Image Representation, Vol. 10, No. 4, pp. 39-62, April 1999.

[SAB97] E. Saber, A. M. Tekalp, and G. Bozdagi, *Fusion of Color and Edge Information for Improved Segmentation and Edge Linking*, Image and Vision Computing 15, pp. 769-780, 1997.

[SAX96] D. Saxe and R. Foulds, *Toward Robust Skin Identification in Video Images*, In: Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, pp. 379-384, IEEE Computer Society Press, 1996.

[SKA94] W. Skarbek and A. Koschan, *Colour Image Segmentation - A Survey*, Technischer Bericht 94-32, TU Berlin, 1994.

[SON98] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis and Machine Vision*, 2nd Edition, PWS Publishing, 1998.

[STA95] M. Stark and M. Kohler, *Video Based Gesture Recognition for Human Computer Interaction*, In: W. D. Fellner (ed.), "Modeling - Virtual Worlds - Distributed Graphics", Infix-Verlag, 1995.

[STR94] N. A. Streitz, J. Geißler, J. M. Haake, and J. Hol, *DOLPHIN: Integrated Meeting Support across LiveBoards, Local and Remote Desktop Environments*, In: Proceedings of the 1994 ACM Conference on Computer-Supported Cooperative Work (CSCW'94), pp. 345-358, 1994.

[TER98] J. Terrillon, M. David, and S. Akamatsu, *Automatic Detection of Human Faces in Natural Scene Images by Use of a Skin Color Model and of Invariant Moments*, In: Proc. of the Third International Conference on Automatic Face and Gesture Recognition, pp. 112-117, Nara, Japan, 1998.

[TIA98] L. F. Tian, and D. C. Slaughter, *Environmentally Adaptive Segmentation Algorithm for Outdoor Image Segmentation*, Computers and Electronics in Agriculture 21, pp. 153-168, 1998.

[TIZ98] H. R. Tizhoosh, *Fuzzy-Bildverarbeitung: Einführung in Theorie und Praxis*, Springer Verlag, 1998.

[WEB71] W. Weber, *Methoden der Regelungstechnik*, Band 1 und 2, Oldenbourg Verlag, 1971.

[WIL92] D. J. Williams and M. Shah, *A Fast Algorithm for Active Contours and Curvature Estimation*, CVGIP: Image Understanding Vol. 55, No. 1, pp. 14-26, 1992.

[WIS01] M. Wissen, M. A. Wischy, and J. Ziegler, *Realisierung einer laserbasierten Interaktionstechnik für Projektionswände*, In: H. Oberquelle, R. Oppermann and J. Krause (Eds.), "Mensch & Computer 2001", B.G. Teubner Verlag, 2001.

[WRE95] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, *Pfinder: Real-time Tracking of the Human Body*. In: Photonics East. SPIE Proceedings Vol. 2615, 1995.

[XU97] C. Xu and J. L. Prince, *Gradient Vector Flow: A New External Force for Snakes*, IEEE Computer Vision and Pattern Recognition, pp. 66-71, 1997.

[XU02] C. Xu and J. L. Prince, *Gradient Vector Flow, MatLab Source Code*, http://iacl.ece.jhu.edu/projects/gvf, 2002.

[XU00] Y. Xu, P. Duygulu, E. Saber, M. Tekalp, and F. Y. Vural, *Object Based Image Retrieval Based On MultiLevel Segmentation*, Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP2000), 2000.

[YAG92] R. R. Yager and L. A. Zadeh (Eds.), *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, Kluwer Academic Publishers, 1992.

[YAN98] J. Yang, W. Lu, and A. Waibel, *Skin Color Modeling and Adaptation*, In: Proceedings of the 3rd Asian Conference on Computer Vision, Volume 2, pp. 687-694, 1998.

[ZAD65] L. A. Zadeh, *Fuzzy Sets*, Information and Control 8, pp. 338-353, 1965.

[ZAD73] L. A. Zadeh, *The Concept of a Linguistic Variable and Its Application to Approximate Reasoning*, Memorandum ERL-M 411 Berkeley, 1973.

[ZAD75] L.A. Zadeh, K.-S. Fu, K. Tanaka, and M. Shimura (Eds.), *Fuzzy Sets and their Application to Cognitive and Decision Processes*, Academic Press, 1975.

[ZIM85] H. J. Zimmermann, *Fuzzy Set Theory and its Applications*, Kluwer Academic Publishers, 1985.

[ZIM87] H. J. Zimmermann, *Fuzzy Sets, Decision Making and Expert Systems*, Kluwer Academic Publishers, 1987.