Vision-based Recognition of Gestures with Context

Jan Nikolaus Fritsch

Dipl.-Ing. Jan Nikolaus Fritsch
AG Angewandte Informatik
Technische Fakultät
Universität Bielefeld

email: jannik@techfak.uni-bielefeld.de

Genehmigte Dissertation zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.).
Von Jan Nikolaus Fritsch am 27. März 2003
der Technischen Fakultät an der Universität Bielefeld vorgelegt.
Am 7. Juli 2003 verteidigt und genehmigt.

Prüfungsausschuß:

- Prof. Dr.-Ing. Gerhard Sagerer, Universität Bielefeld
- Prof. Terry Caelli, University of Alberta, Edmonton, Canada
- Prof. Dr. Helge Ritter, Universität Bielefeld
- Dr.-Ing. Stefan Kopp, Universität Bielefeld

Vision-based Recognition of Gestures with Context

Dissertation zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

der Technischen Fakultät der Universität Bielefeld
vorgelegt von

Jan Nikolaus Fritsch

27. März 2003

Acknowledgments

This thesis would not have been possible without the support of many people who have contributed to my professional and personal development in very different ways throughout these years. Finishing this thesis gives me an opportunity to thank all of them. However, if I try to name them all, I will definitely fail to give everyone the appropriate credit. Therefore, I will name only very few people here, but I hope everybody to whom I am thankful and who is not mentioned knows from personal interaction with me, how much I owe him for getting here.

On the professional side, I have cooperated with many people while carrying out my research. First of all, I therefore want to thank everyone in the Applied Computer Science Group and in the collaborative research center SFB 360 for their cooperation. Particularly, I wish to thank my officemate Elke Braun with whom I have enjoyed many fruitful discussions and who provided many suggestions for improving this thesis. The person that I am most deeply grateful to is my advisor Gerhard Sagerer who believed in me even at times when it was difficult to fully concentrate on my research work. He has given me the opportunity to find my own way during the development of this thesis and created an environment providing the different kinds of support necessary for carrying out this work. I would also like to thank Terry Caelli who agreed to be the second reviewer.

On the personal side, I would like to thank my parents who have both contributed to my development in their unique ways. I am especially thankful for the constant and enthusiastic support of my mother Lisabeth van Iersel during all my life. Last but not least, a thousand kisses for my girlfriend Jomuna Choudhuri who managed to live with my late night writing sessions and makes my life brighter every time she smiles...

Abstract

Out of the large range of human actions, gestures performed with the hands play a very important role during everyday life. Therefore, their automatic recognition is highly relevant for constructing userfriendly human-machine interfaces. This dissertation presents a new approach to the recognition of manipulative gestures that interact with objects in the environment. The proposed gesture recognition can serve to realize pro-active human-machine interfaces enabling technical systems to observe humans acting in their environment and to react appropriately.

For observing the motions of natural hands, a non-intrusive vision-based technique is required. For this purpose, an adaptive skin color segmentation approach that is capable of detecting skin-colored hands in a wide range of lighting conditions is developed. The adaptation step is controlled by using additional scene information to restrict updating of the color model to image areas that actually contain skin areas. With the trajectory data that results from detecting hands in a sequence of images, gesture recognition can be performed.

To recognize gestures with context, a new method for incorporating additional scene information in the recognition process is described. The context of a gesture model consists of the current state of the hand and the object that is manipulated. The current hand state is needed to capture the applicability of a gesture model while the manipulated object needs to be present in the vicinity of the hand to enable recognizing the gesture model. Through the proposed context integration, the developed recognition system allows to observe gestures that are mainly characterized by their interaction with the environment and do not have a characteristic trajectory. The performance of the gesture recognition approach is demonstrated with gestures performed in a assembly construction scenario and in a typical office environment.

The use of the recognition results for improving human-machine interfaces is shown by applying the gesture recognition in a 'situated artifical communicator' system that is situated in an assembly construction domain. Here the information about the executed gesture can be used for improving dialog interaction by providing information about the hand contents. Besides this direct improvement of the human-machine interface, the recognized gestures can also be used as context knowledge for other system components. This is demonstrated with the observation of construction gestures that provide relevant context information for the vision algorithms aiming at recognizing the objects and assemblies in the construction scenario. In this way the gesture recognition results improve the human-machine interface also indirectly.

Contents

List of Figures

List of Tables

1 Introduction

How do humans interact with their environment?

Clearly, there is a huge number of different ways in which humans interact with their environment. Concentrating on body motions, one answer is:

Humans interact with their environment by acting with their hands.

In everyday life, humans make intensive use of their hands to communicate with other humans, or to manipulate their environment. We denote such hand motions as *gestures*. An example for a *communicative* hand gesture is the fist with the thumb up meaning 'everything is alright'. While the usage of gestures for communicative purposes is a very important aspect, many of the hand gestures performed by humans are intended to physically change the environment. For example, we take a cup of coffee and lift the cup to the mouth to drink (see Fig. 1.1). Although the primary goal of such a *manipulative* gesture is the actual manipulation of the environment, other humans can observe the gesture to reason about the acting person ('What is he/she doing'). This reasoning is used by humans for a variety of purposes. For instance, to manage communication: if a human enters a room to talk to somebody, he will usually first check what the person is currently doing. In this way, he can make sure that the person will be able to pay attention to him.

(a) start of gesture at t_0 (b) taking the cup at t_1 (c) drinking coffee at t_2

Figure 1.1: Three pictures taken from an image sequence showing the manipulative gesture 'drinking'.

This smoothness of human-human interaction is one of the goals pursued in research on *human-machine interfaces*. If a technical system with gesture recognition capabilities

can infer what the human is doing by observing his gestures, it will provide a more natural human-machine interface. Although this statement sounds obvious, the interaction of current technical systems with the user is often limited to *reactive* behavior [102]. These systems are constantly waiting for a communicative command from the user to start an operation or to change their current status. A truly *proactive system* would try to gather information about its environment, and/or the user, without being given information explicitly [102]. Such a system would recognize gestures even if the human acts without the intention to communicate. For example, if a human silently drinks a cup of coffee, a proactive system could recognize the gesture and offer the human to play some relaxing music or read out loud new e-mails while he/she is drinking.

Clearly, the recognition of human gestures is just one technique that can be used for improving the interface of a technical system. Other important cues are, for example, the analysis of natural language, prosody, or facial expression. Only the integration of several of these modalities into a multi-modal human-machine interface will enable the construction of systems with human-like communication capabilities that are easy to instruct and use. The development of such artificial systems is believed to be a prerequisite in order to enable users without technical background to utilize technical systems. For example, *Personal Robots* that exhibit entertainment and interaction skills, like the NEC PaPeRo® [72], are intended to become a standard consumer product similar to the personal computer. To enjoy owning such a robot and, therefore, ensure its commercial success, the interaction with the personal robot needs to be very easy for any non-trained person.

In addition to reasoning about the acting human based on observations of his manipulative gestures, a robotic system could learn how to perform the manipulative gestures by itself. This so-called *imitation learning* [4] is one method applied frequently by humans to learn new ways of interacting with their environment and opens up a very interesting approach to instruct robots. Ultimately, it allows us to teach a robot a new task by letting it observe a human performing the task. Many questions relating to, e.g., scalability, still need to be addressed, but first implementations already demonstrate the principal feasibility of this approach [23, 45, 59]. In future robotic systems that are capable of imitation learning in less artificial domains, a method for the recognition of manipulative gestures will play an important role. If an observed action can be recognized as an already learned action, the associated action procedures learned previously can be directly started. This avoids performing a detailed analysis of the motion for imitation learning and, more importantly, allows the use of stored procedures that have been iteratively improved during a learning phase. In this way, gesture recognition enables the transfer from the initial learning of an action to its later imitation based on the learned parameters.

The application of a recognition system for manipulative gestures is not limited to improving the human-machine interface. Rather, the recognized gestures can be used as context knowledge for improving other components of a technical system. One domain considered in this thesis is the construction of assemblies from a toy construction kit for

children called baufix®. If a human builds a complex assembly by connecting elementary baufix® parts, the construction actions leading to this assembly can be recognized with a gesture recognition system. Observing the construction actions allows a technical system to learn the symbolic construction plan for building the assembly, i.e., the sequence of elementary construction actions. Moreover, the assembly structure resulting from the construction plan can serve as *a priori* information for vision algorithms aiming at recognizing the assembly.

Summarizing, the applications of a recognition system for manipulative gestures range from improving proactive human-machine interfaces to the use as context knowledge in other system components. The recognition of manipulative gestures could also form the basis for the advanced instruction of robots through imitation learning.

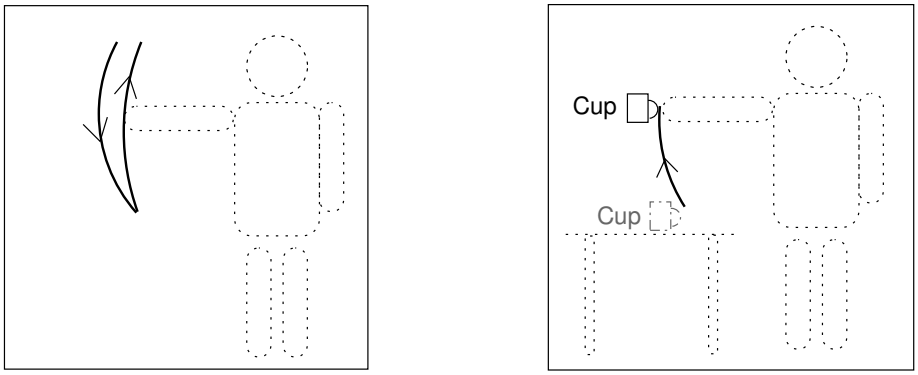
1.1 Categorization of Gestures

As the recognition of gestures is an important technique for advanced human-machine interfaces, there is a large variety of research activities approaching the recognition task from different perspectives. We will focus here on the recognition of gestures carried out with the hands although humans are capable of performing a much broader range of gestures using the complete body. As already introduced above, human gestures can be subdivided into two broad categories based on the aim of the gesture [76]: communicative and manipulative gestures. Let us now take a closer look at the definition of these two gesture types and their associated properties:

Communicative gestures are used by humans during everyday life for a wide range of communication purposes. They are intended to be understood by other humans only through observing the hand motions performed by the gesturing person and/or the hand posture. To assure a successful communication, the gesturing person typically orients the motions and/or the posture towards the recipient of the gesture [21]. This implies that communicative gestures are viewpoint dependent and can only be successfully recognized by an observer if positioned at the viewpoint expected by the gesturing person. Figure 1.2(a) depicts a sketch of the communicative gesture 'waving' consisting of a characteristic motion trajectory but no posture.

Manipulative gestures are different in nature as their primary goal is the manipulation of objects. As handling different objects does usually not result in a large difference in the hand motion, the trajectories of manipulative gestures are often very similar. This makes the recognition of the gesture based only on the *sensory* trajectory information very unreliable or even impossible. However, a manipulative gesture is always accompanied by one or several objects that are being manipulated. For example, Fig. 1.2(b) depicts a symbolic sketch of the manipulative gesture 'picking' that moves a 'cup' object.

The manipulated objects represent *symbolic* information as their type is the relevant feature that needs to be known. Therefore, two types of information are relevant for the recognition of a manipulative gesture: sensory data describing the hand motions, and symbols representing the manipulated objects.



(a) Communicative Gesture (b) Manipulative Gesture

Figure 1.2: The two gesture types resulting from categorizing gestures based on their aim.

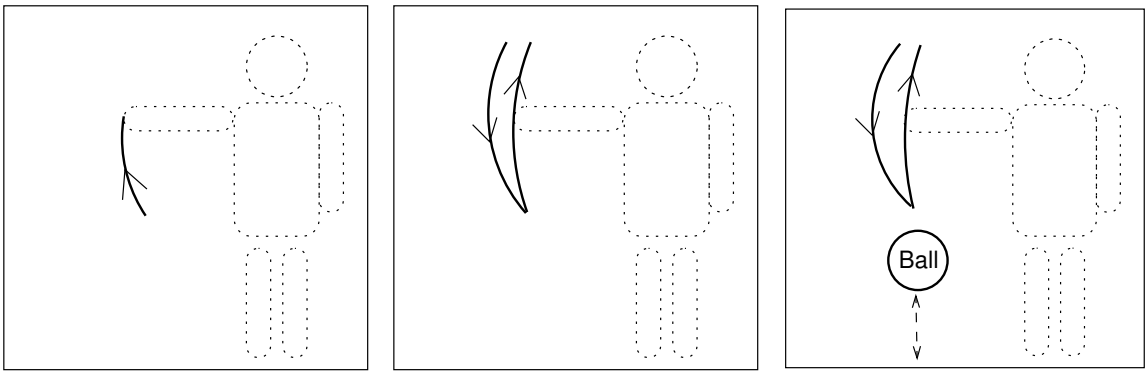
The preceding classification of gestures into two types is motivated by the purpose of the gesture. Another categorization focusing on the differences in gestures relevant for their recognition is more appropriate from the perspective of a system designer aiming at recognizing human gestures. For this purpose, we will now introduce a different classification scheme proposed by A. Bobick [13] that divides human motions into three different categories: movement, activity, and action. Bobick's categorization aims at allowing a better understanding of the underlying assumptions used in the approaches published on recognizing human motions based on visual observations (for an overview see e.g., [68, 1, 36]). The proposed categorization allows a better understanding *what kind of* knowledge is used in a human motion understanding approach and *where* it is used in the recognition process. The three different motion categories are equivalent to different levels of knowledge:

Movement: A basic motion that can be reliably detected solely on the basis of low-level images features is defined as a *movement*. This implies that the variation within different instances of the movement is small, and the internal representation of a movement for recognition depends only on the viewing condition. In particular, the execution speed of different motions is assumed to vary only linearly, i.e., a movement can be executed at different speeds but there are no speed variations within a single movement. Note that the recognition of a movement requires no context knowledge like previous movements or objects in the vicinity of the moving human.

Activity: An *activity* describes a motion that is made up of a sequence of movements that can be recognized with pattern matching approaches. In particular, the sequence of movements of an activity can contain movements that are optional to this activity. Moreover, an activity exhibits more complex temporal variations than the linear scaling of movements. Like movements, activities do not refer to elements external to the actor performing them.

Action: At the highest level of abstraction are *actions*. Bobick describes actions as being "at the boundary of where perception meets cognition". This view on actions is due to the fact that different instances of the same action can have different perceptual features. Only by linking the observed motion to its context, e.g., to a manipulated object, it is possible to recognize actions. Therefore, the recognition of actions necessarily needs some kind of symbolic information.

Figure 1.3 shows a symbolic sketch of a human performing three different hand motions relating to the three categories defined by Bobick. Note that only the information represented by solid lines is available to the algorithms for performing motion understanding, indicating the human by a dotted line is done to facilitate an intuitive example. The hand going up in Fig. 1.3(a) is a movement, as it can be represented by the motion of the hand only. The gesture in Fig. 1.3(b) is an activity, as it consists of a sequence of up and down hand movements, with arbitrary delays inbetween. Depending on the context in which this gesture is executed, it can be associated with different activities like, e.g., 'waving', 'hitchhiking', or 'exercising'. Finally, Fig. 1.3(c) depicts the same hand motions, but now there is an additional symbolic information 'ball' present. To recognize this motion as action 'dribbling', the trajectory data has to be linked to the object 'ball' in its vicinity.



(a) Movement (b) Activity (c) Action

Figure 1.3: Examples for the three categories of human motion defined by Bobick [13]. The different motion categories are equivalent to different levels of knowledge necessary for recognizing such motions.

Note that it is possible to hypothesize the action 'dribbling' from recognizing the activity depicted in Fig. 1.3(b) without analyzing the symbolic information. However, inferring actions based on activities is limited to domains containing no other gesture with a similar trajectory. Given two gestures with similar hand motions like, e.g., 'exercising' and 'dribbling', these gestures can only be differentiated based on the presence/absence of the symbolic information 'ball'.

Relating these three categories to the two gesture types introduced above can be done easily for the manipulative gesture. As this gesture is always related to a symbol, i.e., some object that is manipulated, a manipulative gesture is in Bobick's categorization an action. In contrast, communicative gestures cannot be associated with a single category, as humans use a large number of different communicative gestures covering the whole range of categories. Fig. 1.3(a) and Fig. 1.3(b) have already shown two examples of communicative gestures being a movement and an activity. A communicative gesture of the action type is, for example, a deictic gesture. In such a gesture pointing at an object there is a strong relation between the trajectory and the symbolic object information emphasizing the action type of the communicative gesture.

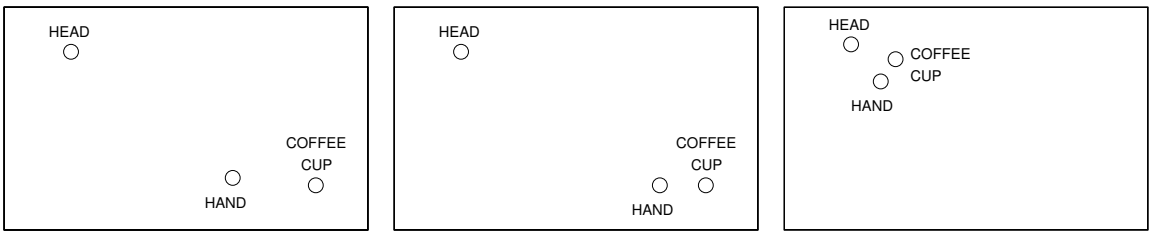
A different categorization for describing arbitrary scenes containing moving objects has been proposed by Nagel [71] at the beginning of image sequence analysis in 1988. He introduced the terms 'change, event, verb, episode, history' for denoting the intermediate levels of description between the difference in image appearance of two consecutive images and the associated high-level description of 'what is happening'. Nagel's categorization was part of his pioneering work on motion understanding and allowed to focus more closely on the relation between image sequences and their natural language descriptions. Therefore his categorization reflects different dimensions than those discussed in this thesis. Here we concentrate on a specific type of motions, the gestures performed with the hand, and we are interested in bridging the gap between sensory and symbolic information for recognizing these gestures. For this purpose, we use Bobick's categorization as it is more appropriate.

1.2 Recognition of Manipulative Gestures

In the previous section we have seen that the recognition of manipulative gestures requires the analysis of symbols and sensory data. To highlight the different properties of both data types, we will first take a closer look at them using the 'drinking' example from Fig. 1.1 before going into the details of actually recognizing gestures.

Consider a scenario where all gestures manipulate objects by translations or rotations, and the information about *how* objects were moved is not relevant. In such a domain, it is possible to base a recognition approach solely on the position changes of the objects in the scene to deduce the actions that must have been executed. As such an approach relies only on rules and symbols, i.e., on the symbolic object labels extracted from the image data by an object recognition algorithm, we will call this approach *symbol-based*

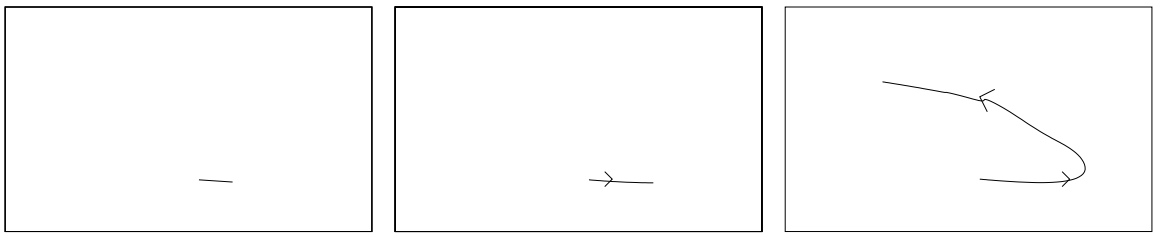
action recognition. However, such a symbol-based approach relies on perfect object recognition results. To give an example for the symbolic information that is available to such a recognition approach, Fig. 1.4 depicts the object recognition results for the three time instances of the drinking action shown in Fig. 1.1. Note that the time instances for analyzing the symbols need to be chosen carefully and that the successful recognition of an object during manipulation is not always possible due to, e.g., the hand occluding the object.



(a) symbols at t_0 (b) symbols at t_1 (c) symbols at t_2

Figure 1.4: Symbolic information for the manipulative gesture 'drinking'.

Alternatively, if the information about *what* object was moved is not relevant, the sensory data of the hand trajectory is sufficient, and the use of symbolic information can be avoided. This is visible in Fig. 1.5 showing the hand trajectory for the 'drinking' gesture. The trajectory does not allow for discriminating between 'drinking' and 'answering a phone call' as the information about what object is being moved is crucial to differentiate these two gestures. Nevertheless, the trajectory is sufficient to separate between 'drinking' and 'writing' without having explicit information about the objects that are manipulated. In a restricted domain with only these two gestures, a *data-driven activity recognition* using the trajectories of the hand motions allows us to successfully differentiate these two actions.

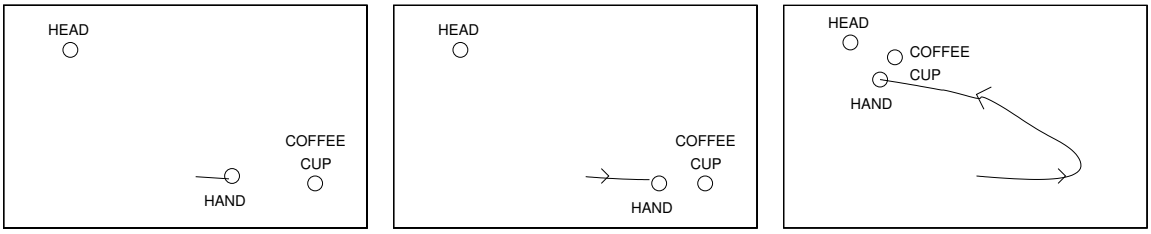


(a) sensory data at t_0 (b) sensory data at t_1 (c) sensory data at t_2

Figure 1.5: Hand trajectory representing the sensory data for the manipulative gesture 'drinking'.

Obviously, the combined analysis of the symbolic object information and the sensory data of the hand motions is often needed to recognize a wider range of manipulative

gestures. As an illustrative example, the two types of information are depicted in Fig. 1.6 for the 'drinking' gesture.



(a) both data types at t_0 (b) both data types at t_1 (c) both data types at t_2

Figure 1.6: Symbols and sensory data for the manipulative gesture 'drinking'.

Performing Gesture Recognition

The majority of the published research work on gesture recognition (overviews can be found in [54, 76, 104]) deals with recognizing communicative gestures. All the gestures considered fall into the movement or activity category defined by Bobick. In human-machine interfaces, these communicative gestures are often used for the *direct* instruction of systems [95, 12, 98].

In contrast, the recognition of manipulative gestures is a relatively unexplored topic. As pointed out above, all manipulative gestures are of the action type as they require symbolic information for their recognition. A few approaches aim at inferring human actions based on evaluating only symbols extracted from the scene [89, 63, 67]. Approaches to action recognition that work directly with image data use the sensory hand trajectory information only for focusing on the image areas relevant for action recognition [77, 3]. The actions that are recognized are defined in terms of low-level image features like, e.g., an intensity change if a cabinet is opened. In this way, the symbolic information indicating a change in the state of an object ('cabinet opened') is the only feature used for recognition. Besides analyzing low-level image features, the closeness of the hand to a predefined image area is also used for hypothesizing actions. Obviously, there is no sophisticated analysis of the sensory data as the only purpose of the trajectory information is the selection of the interesting image area for extracting the symbolic information.

One noticeable approach to action recognition in an office environment that combines the analysis of sensory data **and** symbolic context information provided by a scene model is the work by Moore et al. [69]. In their work, the symbolic information about the objects in the vicinity of a human hand comes into play *after* the analysis of the sensory data from the hand motions. Their integration scheme performs a *sequential* fusion by post-processing recognized activities to realize action recognition.

We argue that the *parallel* fusion of both cues in an *integrated action recognition* approach improves the recognition quality, and allows for an easy implementation of the developed techniques in different application domains. An important advantage of an integrated action recognition system is the possibility to concentrate computational resources on gestures possible with the objects available in the current scenario. For example, a 'drinking' gesture can only occur if some sort of jar containing a beverage is present in the scene. Without the consideration of context knowledge, a recognition algorithm would need to analyze the complete set of gestures. Especially in domains with a large set of gestures, this can result in spending much computational time on gestures currently impossible due to missing objects.

Extracting Symbols: Some Remarks on Object Recognition

Considering that the symbolic information needed for the recognition of manipulative gestures is defined by the objects in the scene, we will point out some aspects relevant to their extraction from vision data. In general, a manipulative gesture changes the position, form, and appearance of the object that it manipulates. In the 'drinking' example in Fig. 1.1, the person first changes the position of the cup by lifting it to the mouth, and then changes the contents of the cup itself by removing the contained coffee. As these changes of the object are directly related to the manipulative gesture, their recognition is an important source of information for a gesture recognition system.

Research activities dealing with the problem of recognizing objects and their current states show that this is a very demanding challenge [52]. Pure translational and rotational motions do not change the object itself, but its appearance may change due to a new viewing angle or a different lighting condition. The recognition of a rigid object under different viewing angles and variations in lighting is already a challenging task for state-of-the-art object recognition algorithms. It becomes an open problem if flexible objects are considered whose form varies unpredictably. Even if the variations in the object's form can be modeled in the recognition algorithm, i.e., they are known beforehand, the recognition task is still challenging. To our knowledge, there are no feasible object recognition algorithms that actually tackle the problem of recognizing a number of deformable objects based on 2D images from a color camera.

Consequently, we consider here only object manipulations that do not change the objects or their appearance in a substantial way. In this thesis, we will use the term *manipulation* to denote only translational and rotational motions with objects or relative to objects. This includes manipulative gestures like the example in Fig. 1.1 that change the objects contents but do not change substantially the visual appearance of the object as seen by the camera. In general, only small changes in the object's appearance that do not prohibit the successful recognition of the object are allowed.

Extracting Sensory Data: Hand Detection & Tracking

Due to its importance for recognizing communicative gestures, the detection and tracking of hands in image data is an active research field [91, 108, 78]. To observe the motions of the hands, different types of sensor devices can be used. Basically, the hand motions can be measured either through sensors attached directly to the hands, or via some kind of remote sensing. While the use of attached sensors is easier to realize from a technical point of view, it requires the human to wear special equipment. This requirement conflicts with the goal outlined above to have user-friendly systems that are easy to use. From the perspective of a system designer, remote sensing techniques are much more appealing as they allow any user to immediately interact with the system without any preparation. However, besides this pragmatic advantage, the non-intrusive measurement of hand motions is especially important for recognizing manipulative gestures that involve physical contact with the object. Only remote sensing techniques allow the human to handle objects naturally without any constraints introduced by, for example, data gloves that would make object manipulations more difficult or even impossible. Therefore, we will concentrate here on non-intrusive sensing techniques applied to gesture recognition.

Instead of resorting to infrared cameras or laser range finders, we will focus on the processing of color images provided by a standard video camera for obtaining the hand motions. The choice of a video camera as input device is motivated by the goal of recognizing the hand gestures within their context. As ordinary objects do not exhibit any infrared radiation, the context cannot be detected by infrared cameras. Laser range finders are only appropriate if the objects are separable by depth information. Additionally, laser range finders are expensive sensors that would prohibit the use of the proposed gesture recognition approach in a commercial system where sensors must be cheap. In contrast, video cameras are still becoming cheaper as there is a mass market for cameras used in video-conferencing applications.

The images provided by a video camera have to be processed with vision algorithms to obtain the hand motions. A prominent visual feature of a human hand is the color of the skin. Given a color model for human skin, an image can be segmented into skin and non-skin areas and the region descriptions of the skin-colored areas in the input image can be extracted. Segmenting an image based on the color cue provides the positions of all skin-colored objects in two-dimensional image coordinates. Due to the fact that the color cue is scale-invariant, no depth information is available for the segmented areas. The positions of the segmentation results of individual images can be associated over time by using constraints enforcing the temporal coherence of the motions of the individual regions found. This leads to trajectories describing the two-dimensional motions of the skin-colored regions in the image sequence. If several trajectories have been found, additional constraints depending on the domain may have to be applied to select the trajectory of the hand performing the gestures.

The sensory trajectory data can be used together with the symbolic object information

to recognize manipulative gestures. Many classification algorithms are available for recognizing trajectory data by comparing the sensory data with known motion models of human hands [54, 76, 104]. However, the incorporation of symbolic information into the trajectory analysis is not straightforward. In the published action recognition approaches applying symbolic information, the integration schemes are specifically designed for the domain of interest, and fusion is performed independently of the trajectory analysis [77, 3, 69].

1.3 Contribution

The contribution of this thesis is the development of a novel approach for the recognition of manipulative gestures that incorporates symbolic and sensory information *during* the recognition process. The proposed approach is based on vision data obtained from a camera observing the gesturing person. Due to the missing depth information in color images, the described framework concentrates on the analysis of data describing the sensory and symbolic information in the two-dimensional image coordinates.

One important source of information for gesture recognition is the trajectory of the gesturing hand. To obtain the hand trajectory, we introduce an adaptive algorithm for skin color segmentation that localizes human skin regions under varying lighting conditions. Based on domain-dependent constraints, the trajectory belonging to the gesturing hand is selected. This sensory data forms the input to a particle filtering algorithm that is used for recognizing the gestures. To integrate symbolic context knowledge necessary for classifying manipulative hand gestures, we extend the recognition algorithm for sensory data (a version of the CONDENSATION-algorithm [12]). Our approach adds a processing step that compares the expected symbolic context with the symbols currently in the scene. To keep the computational complexity tractable, we define 'context areas' for manipulative gestures that are searched for the expected symbolic object context. The proposed integration framework can be easily applied to different application domains. We have tested our approach in two domains: assembly construction and usual office behaviors.

The assembly construction domain is part of the Situated Artificial Communicator research project described in more detail in Section 2.2. The objects are wooden parts from the baufix® construction kit that are used to form assemblies with 'pick', 'place', and 'connect' actions. This domain serves to prove the benefit of recognizing manipulative gestures without a direct communicative purpose. Monitoring the assembly construction process with the integrated gesture recognition reveals the construction actions leading to the actual scene. The integration of this context knowledge has been evaluated for object and assembly recognition algorithms [18, 8]. Furthermore, the current state of the scene and the construction process is used to improve speech understanding [7], and to control the dialog with the user for resolving ambiguities [29]. The overall performance of the integrated Situated Artificial Communicator system has been evaluated

in several experiments [9] proving the relevance of the non-communicative information from manipulative gesture recognition for the overall interaction quality.

Different from the assembly construction domain, the office domain contains more manipulative gestures that interact with a wider range of objects. In this setup, the camera observes a human sitting at an office desk and performing the actions 'pick up phone', 'hang up phone', 'take cup', 'stop drinking', 'pick book', 'stop reading', and 'type on keyboard'. As the manipulated objects are spread out over the office desk, the gestures performed in this domain exhibit more variations in the trajectory data due to the different ways in which humans handle the objects. Therefore, these gestures can be recognized correctly only by incorporating the symbolic context information describing the objects on the table.

The evaluation of our integrated recognition approach in these two domains demonstrates that the proposed framework is capable of recognizing manipulative gestures from two-dimensional motion information extracted from vision data. With the applications realized in the Situated Artificial Communicator, we show that our framework provides the basis for improving the overall performance of systems interacting with humans who gesture with their hands.

1.4 Outline

The contents of this thesis are organized as follows: We start in Chapter 2 with a description of the assembly construction domain and the realized *symbolic action detection* approach to recognize assembly construction actions based on symbolic information. After presenting the symbol-based recognition approach, we turn to the analysis of sensory trajectory data for recognizing gestures. Chapter 3 introduces a model for human activities, and gives an overview of relevant probabilistic recognition algorithms for recognizing human activities. Using such a data-driven activity recognition approach to recognize manipulative gestures requires the availability of descriptive features for the human hand performing the gestures. For detecting human hands and faces in color images, we developed an adaptive skin color segmentation method that is described in Chapter 4. Based on the detection of human hands, trajectories can be constructed and, subsequently, a *visual activity recognition* can be performed. Chapter 5 covers the realization of such an approach for the recognition of manipulative gestures in the assembly construction domain.

With the symbolic action detection presented in Chapter 2, and the visual activity recognition described in Chapter 5, we have introduced two different recognition approaches that use only a single type of information for recognizing manipulative gestures. Based on these two approaches, the developed framework for fusing symbolic and sensory information in an integrated action recognition system is described in Chapter 6. Using the proposed framework for recognizing manipulative gestures is demonstrated in the assembly construction domain and in an office domain. The developed action recog-

nition system allows us to reason about the human by visually observing his actions. Chapter 7 proves the benefit of utilizing this context knowledge with various applications increasing the overall performance of the human-machine interface in the Situated Artificial Communicator. Chapter 8 summarizes the contents of this thesis and the conclusions we draw from this work.

2 Symbolic Action Detection

In the first computational approaches to image understanding, the image contents were represented by symbols and logical operators used these symbols to reason about the image. For example, Sussman [94] developed a system that learns actions in a blocks world domain. In this system, the symbols are the positions of the different blocks and the changes between two configurations of blocks are used to construct logical operators. These logical operators are applied to subsequent configurations and on finding a contradiction the operators are changed appropriately.

Attempts to test these early approaches in real world environments revealed the basic problem of symbolic approaches: how to relate symbols to the dynamic features of a continuous world extracted, e.g., from vision data. This is also known as the *symbol grounding* problem [40]. A system intended to perform image understanding using logical operators can achieve symbol grounding in two different ways:

Separate Algorithms: Symbol grounding and image understanding are accomplished separately. The extraction of symbols from sensory data is extensively studied in the research work on pattern classification to develop, for example, object recognition algorithms. Such a pre-programmed or learned mapping between sensory data and symbols forms the basis of a subsequent symbolic image understanding algorithm. However, dynamic environments as well as ambiguous sensory data often cause errors in a hard-wired symbol grounding approach. Another more important aspect is the fact that many environments contain a potentially huge amount of different symbols. These symbols can represent physical objects like persons or specific spatio-temporal properties of the observed scene. For example, the symbol 'waving' could be assigned to some characteristic hand motion. In general, mapping spatio-temporal data to a symbol is more difficult than mapping the visual observations of a static, physical object to a symbol. Consequently, the selection of the appropriate set of basic symbols denoting basic motions is crucial to perform correct symbol grounding and detect actions successfully.

Integrated Algorithm: Using an integrated approach for symbol grounding and image understanding allows us to perform task-dependent symbol grounding. Through applying vision algorithms tailored to the symbols relevant in the current context, 'early' errors during symbol grounding that prevent a successful image understanding can be avoided. Therefore, such an integrated algorithm that exploits the inherent dependencies between symbol grounding and image understanding is

much more powerful. However, up to now no generic approach capable of performing in a wider range of application domains has been published. Instead, a few applications with a high implementational effort have been developed that were tailored to specific situations (e.g. [60]). Problems like, e.g., the scalability of an architecture containing task-specific vision algorithms have been identified, but to our knowledge no solution has been offered up to now.

In order to evaluate the potential of symbolic information for action recognition, we chose to implement a rule-based method operating on appearing and disappearing objects in a scene to infer the construction actions executed with parts from the wooden construction kit baufix® for children. This assembly construction scenario is used as experimental setting in a Collaborative Research Center[1] for the development of Situated Artificial Communicators. Embedding our approach in this research project allows the use of the action sequence recognized during assembly construction as context knowledge for other modules in the system. Applying this information to improve the human-machine interface will be described in Chapter 7.

The symbolic action detection for the Situated Artificial Communicator scenario makes use of the symbolic information from object recognition algorithms. Besides recognizing elementary baufix® objects, more complex objects constructed by connecting elementary objects have to be recognized. In the following, these complex objects are denoted 'assemblies' and we will use the term 'part' to denote either an elementary object or an assembly made from elementary objects. The changes in the scene over time, i.e., the appearing and disappearing parts, are combined with a rule-based approach to infer the actions executed in the scene. Implementing the action recognition separately from the symbol grounding offers two advantages: First, in the overall system symbol grounding is already achieved since symbolic object labels are needed by subsequent processing modules anyway, e.g., for realizing a dialog with the user about the scene. Consequently, as the object recognition algorithm relates sensory data to object labels, the symbol grounding problem can be viewed as 'solved' for this domain. Secondly, using already available symbolic data avoids the need for additional sensors and additional computational cost for processing sensory data.

We will start in Section 2.1 with a review of some recent approaches to symbolic image understanding for action recognition. The details of the assembly construction domain and the algorithms for extracting the symbolic data are outlined in Section 2.2. Our symbolic approach to action detection is presented in Section 2.3. The chapter concludes with a summary in Section 2.4.

[1] funded by the German Research Foundation (DFG) as 'Sonderforschungsbereich 360: Situierte Künstliche Kommunikatoren'

2.1 Related Work

Recognizing human actions by means of symbolic reasoning has a long history in the field of artificial intelligence. However, due to the inherent symbol grounding problems described above the interest in purely symbolic approaches declined.

In the last years, important results on the computational perception of actions and events have been achieved by R. Mann, A. Jepson, and J. Siskind. Siskind developed a rich framework incorporating qualitative physics to link visually observed events with spatial-motion verbs like *pick*, *place*, *drop*, and *throw* [89]. The visual observations consist of contour and line segments which are automatically grouped to objects during the analysis. For these objects, simple events like changing support, contact, and attachment relations are extracted by kinematic constraints. The verbs are essentially aggregate event descriptions that are recognized based on the truth values of the simple events detected. However, this framework has been tested only with simulated contour and line data to avoid the symbol grounding problem. To give an example for the type of input data that can be processed by Siskind's event perception system, several key frames from a synthetic movie are depicted in Fig. 2.1.

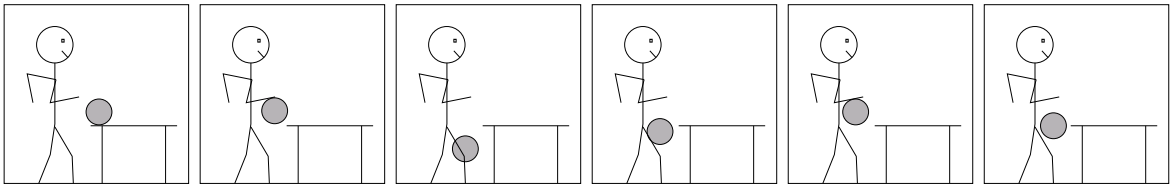


Figure 2.1: Several example key frames from a 'dribbling ball' sequence representing the type of input used by Siskind's event perception system [89].

In a joint work, Mann, Jepson, and Siskind developed an algorithm for interpretation of real images to recognize events like 'picking a coke can' [64]. Here the instantaneous motion of object polygons observed at particular frames of an image sequence is used as input data. The basic symbols, i.e., the object polygons, are extracted by a separate algorithm that uses templates for finding the objects in an image. While Siskind [89] used qualitative physics, Mann et al. [64] use Newtonian mechanics for extracting the 'force-dynamic' properties of all objects detected in the image. Due to the analysis of single images, each object has several plausible properties. Reasoning about the objects with all their different possible force-dynamic descriptions leads to several interpretations for a single scene. Using a preference hierarchy, all plausible interpretations are selected, i.e., in their joint work there is no single recognition result but rather a set of results.

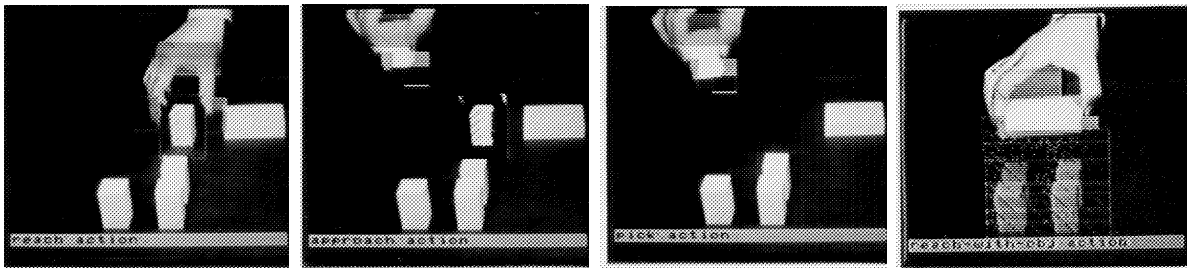
In later work, Mann and Jepson extended the instantaneous scene analysis system to handle temporal information [63]. This extension reduces the number of plausible interpretations, as only those interpretations that are consistent with the temporal context are considered further. While their approach is theoretically well grounded, their system depends on several manual steps to be applied to real world problems. For example, on

initialization the templates describing objects of interest have to be manually selected to enable the system to track the objects in the image sequence. Using an automatic object recognition system, a large number of objects would be detected. With a growing number of objects, the scalability problem would become more important, as the selection of the recognition result from the set of plausible scene interpretations becomes much more difficult.

Another approach applying physics-based representations is the work by Ikeuchi et al. [44, 45] extended later in cooperation with Miura [67]. In their approach, kinematic relations between objects in the scene are extracted from data obtained by a laser-range sensor. These kinematic relations are used in the so-called *Assembly Plan from Observation* framework to generate robot control programs for assembly construction automatically by observing a human constructor. However, the details of the human hand motion itself are not analyzed. Observing the human hand is only performed to detect *whether* an action has been executed. On detecting a large motion of the hand in the scene, it is assumed that an action has been performed and the differences in the kinematic relations of the objects are computed. Based on changed contact relations between objects, the assembly actions that must have been executed are hypothesized. To build an identical assembly with a robot, the robot must be controlled so that it achieves the same differences in the contact relations as have been observed during the human assembly construction.

In the work of Kuniyoshi et al. [60, 59], the spatio-temporal features to realize a qualitative action recognizer are extracted by means of image processing. The recognizer consists of an environment model representing the different elements in the scene and an action model to capture the current state of action recognition. Both, environment model and action model, have a hierarchical structure to capture the different levels of detail that make up an action. For example, an action may be a sequence *local motion - transfer - local motion* where the local motion itself is a sequence *approach - fine motion - depart*. Each element in the different hierarchical levels of the environment model has simple visual routines [97] attached to it that extract specific visual features relevant for action recognition. For example, an element representing a *hand* has visual routines that detect fast or slow motion of the hand. A hierarchical parallel automata is used to relate action and environment model. The state transitions at the different hierarchical levels are initiated based on the visually detected events associated with the elements in the hierarchical environment model. Maintaining the correct motion context, i.e., to what level of the hierarchy the current motion belongs, is achieved by taking advantage of the hierarchical structure. If a visual event is detected, all automata at lower hierarchical levels are forced to terminate because they process more detailed information that does not need to be analyzed any more if a 'larger' motion is detected. In this way, the focus of attention is controlled and motion segmentation points are automatically generated. Recognizing an action consists of detecting the associated sequence of states in the automata at the highest hierarchical level where each state is detected if all states of its corresponding automata at the next lower hierarchical level are successfully detected.

The approach has been demonstrated to successfully recognize the construction actions of a human building towers and arches in a simple blocks world (see Fig. 2.2). However, to our knowledge this system has not been applied to other problems and has not been extended in any substantial way.



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

Figure 2.2: Example images from the blocks world domain used by Kuniyoshi's system for recognizing construction actions (from [60]).

2.2 The Situated Artificial Communicator Domain

The symbolic action detection described in this chapter has been developed within the Situated Artificial Communicator scenario. The system is intended to behave like a human constructor with regard to its communication abilities and its actuators. Since this general formulation of an *Artificial Communicator* has proven to be a very challenging goal, the research project focuses on developing an artificial communicator for a special domain with restricted complexity [27]: the system is *situated* in an environment consisting of wooden, colored parts from the construction-kit baufix® for children. Consequently, it only has to understand instructions and execute actions related to the construction of assemblies. The set of baufix® parts available for construction and a simple assembly are depicted in Fig. 2.3.

The scene in which all construction actions are carried out is limited to the area of a table. Figure 2.4 gives an impression of our setup that is used to develop the *perceptual front-end* of the system. Note that in this setup used for developing the perceptual front-end of a Situated Artificial Communicator, no robotic manipulator is present for actually executing actions. To obtain speech input with good signal quality, the human instructor wears a headset. A static camera is mounted in an angle of 45 degrees above the table to obtain images of the scene. Note that the static camera can only acquire images of the table scene. The human instructor is observed by an active pan-tilt camera that is positioned next to the static camera.

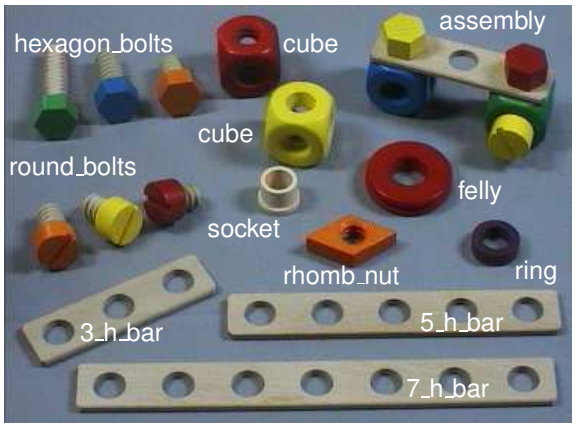


Figure 2.3: Several baufix® parts and a simple assembly made from individual parts.

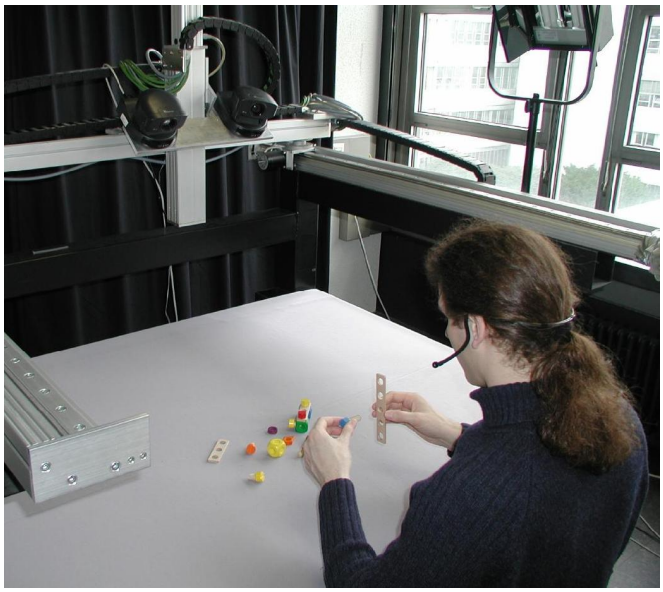


Figure 2.4: The Situated Artificial Communicator scenario with a human instructing the system. The scene on the table is observed by the camera on the left and the user actions are observed by the camera on the right.

2.2.1 The Assembly Model for baufix® Parts

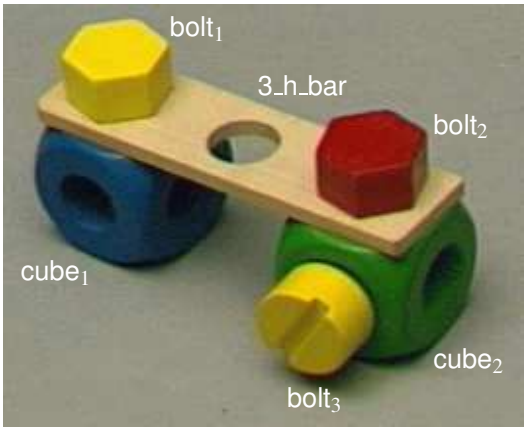
In the scenario described above, an assembly is constructed by connecting elementary baufix® parts. All assemblies considered here are of the bolt-nut-type where miscellaneous parts like rings and bars can be put on bolts and are fastened using nut-type parts like cubes or rhomb-nuts. Due to the multi-functional nature of the parts, the number of different assemblies that can be built is enormous. This large number makes it impractical to model every assembly in advance.

To avoid modeling all possible assemblies, a functional assembly model was developed by C. Bauckhage ([6], Ch. 3) that is based on the functional properties of the parts. Each assembly is modeled to be composed of a bolt part, a nut part, and a number of optional miscellaneous parts. The length of the thread of a bolt part and the size of the miscellaneous parts determines how many miscellaneous parts can be put on a bolt part. Each of the parts in an assembly can be an elementary baufix® part or an assembly itself, therefore the assembly model has a recursive nature. Written as a grammar with terminals (baufix® parts in lowercase letters) and variables (uppercase letters), the recursive assembly model is ([6], page 31):

ASSEMBLY:	BOLT_PART MISC_PART* NUT_PART
BOLT_PART:	ASSEMBLY | round_bolt | hexagon_bolt
MISC_PART:	ASSEMBLY | 3_h_bar | 5_h_bar | 7_h_bar | felly | socket | ring
NUT_PART:	ASSEMBLY | cube | rhomb_nut

Due to the multifunctional nature of the elementary baufix® parts that make up an assembly, each assembly can be used as a component in a larger assembly. To function as a BOLT_PART, MISC_PART, or NUT_PART component in a larger assembly, an assembly needs to contain free 'ports' of the appropriate type. For example, an assembly can only function as NUT_PART if there is at least one free thread in the assembly. Likewise, it can only function as MISC_PART if there is at least one free hole in the assembly.

Based on the compact model of bolted assemblies, it is possible to derive hierarchical structural descriptions for every assembly that can be constructed. One disadvantage of this structural description is the fact that it is not unique. Depending on the number of bolts in the assembly, there exist several structural descriptions for one assembly. Fig. 2.5 shows an example assembly together with a few structural descriptions.



(a) ASSEMBLY: $bolt_2$ MISC_1 NUT_1
MISC_1: $bolt_1$ 3_h_bar $cube_1$
NUT_1: $bolt_3$ $cube_2$

(b) ASSEMBLY: $bolt_3$ NUT_1
NUT_1: $bolt_2$ MISC_1 $cube_2$
MISC_1: $bolt_1$ 3_h_bar $cube_1$

(c) ASSEMBLY: $bolt_1$ MISC_1 $cube_1$
MISC_1: $bolt_2$ 3_h_bar NUT_1
NUT_1: $bolt_3$ $cube_2$

Figure 2.5: Example assembly together with three structural descriptions.

The graphical representation of the structural descriptions listed in Fig. 2.5 is shown in Fig. 2.6. The algorithm developed by C. Bauckhage [6] for automatic extraction of these structures from image data will be presented after first describing the method for recognizing elementary parts.

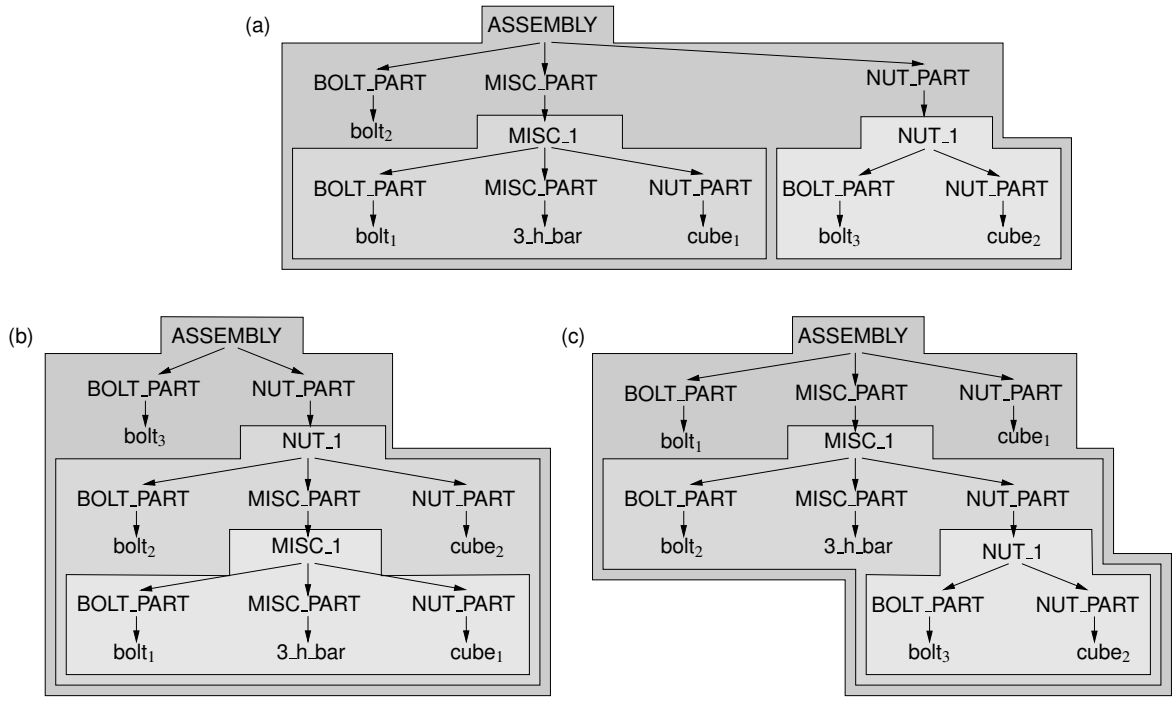


Figure 2.6: Graphical representation of the three structural descriptions.

2.2.2 Object Recognition

To perform the recognition of elementary baufix® parts from images depicting the table scene, E. Braun combined multiple cues from segmentation and classification algorithms in a recognition framework [18, 85]. In the following we will give a short overview of the object recognition approach, the interested reader is referred to [85] for more details.

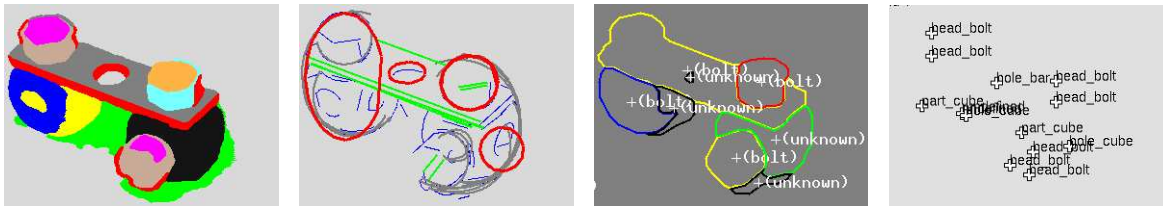
Currently the following modules provide segmentation or classification results that are incorporated into the recognition approach:

Color-based region segmentation: Following the mean-shift algorithm introduced by Comaniciu and Meer [19], the number of colors occurring in the image is reduced problem-independently by clustering within the color space. Afterwards each pixel is associated to a color class and regions of homogeneous color are extracted based on neighboring pixels belonging to the same color class (Fig. 2.7(a)).

Contour-based perceptual grouping: In a first step contours are extracted from the results of a standard Sobel operator ([39], Ch. 3.7) by approximating edge elements with straight line segments and elliptical arcs [62]. Then, these elements are grouped with an approach developed by Massmann et al. [65] by applying various Gestalt laws independent of the actual scenario. Beside others, hypotheses for closed contours are generated based on collinearity, curvilinearity and proximity and are used as hypotheses for object surfaces (Fig. 2.7(b)).

Hybrid object recognition: The first classification module developed by F. Kummert et al. [58] provides hypotheses for elementary baufix® parts based on regions with baufix® color. The center of gravity of each region serves as a *point of interest*, where feature extraction and classification by an artificial neural network takes place [42]. This result initializes the instantiation process of a semantic network [86] exploiting knowledge about the fixed set of baufix® parts and the features of the color regions. As this module is optimized for the detection of isolated baufix® parts, its results are reliable only if elementary parts are totally visible or just slightly occluded (Fig. 2.7(c)).

Holistic detection of object parts: Due to perspective occlusions often only subparts of elements are visible. The goal of a second classification module developed by G. Heidemann [43, 41] is to recognize them by first selecting interesting points based on symmetry and color homogeneity. The image areas around these so-called *focus points* are classified by a neural classifier to obtain hypotheses for object parts (Fig. 2.7(d)).



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

Figure 2.7: Results for the example assembly (a) Color-based region segmentation; (b) Contour-based perceptual grouping; (c) Hybrid object recognition; (d) Holistic detection of object parts;

Generally, the segmentation algorithms are not able to yield a single segment for each object that has to be identified. Additionally, the object hypotheses generated from the different classification modules may contradict each other because of classification errors. Given this, deriving hypotheses for the baufix® elements means to coherently group and label the image segments that probably correspond to object regions. To

accomplish this task, the object recognition algorithm first constructs a unified representation for the segmentation and classification results that is called *segmentation hierarchy* (see Fig. 2.8).

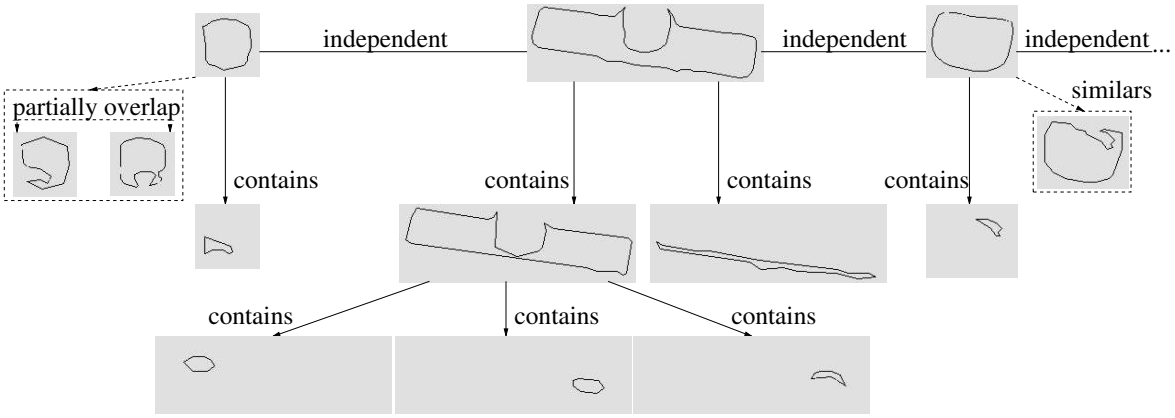


Figure 2.8: Hierarchy of segmentation results built from areas and their relations. Relations that are dashed within the figure are not considered for extracting object hypotheses.

In a second step, the segmentation hierarchy is analyzed. The procedure interpreting the hierarchy examines each independent area sequentially starting with the smallest segmentation results contained in an area. For each of these 'elementary' areas at the lowest hierarchical level of an independent area, a hypothesis is generated according to the attached object label. In case of more than one label for an elementary area, the most probable label is chosen by voting [75]. Subsequently, the next higher level of the hierarchy is evaluated. For each area, the object labels already attached to it and the results from the contained lower level areas are taken into account.

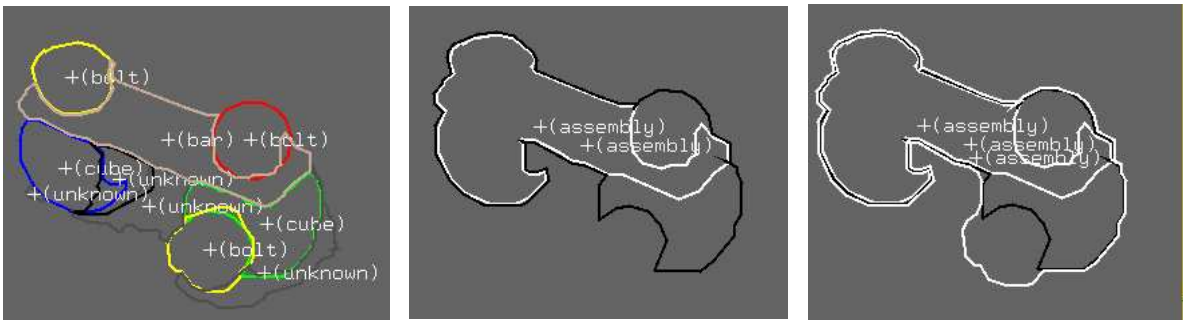
In processing higher levels the question arises whether an area and all its included areas belong to one element or if this area represents several elements. As this question cannot be answered based only on the local information, competing results are generated at all levels of the hierarchy and serve as input for higher hierarchical levels. If results contradict a set of geometry rules, they are pruned. For example, a bolt cannot contain another element and a hole of a bar cannot contain more than one bolt. After evaluating the hierarchy, several competing interpretations of the scene are available.

The selection of the best interpretation is performed by calculating a judgment for each interpretation. This judgment is derived based on the segmentation and classification data contained in the interpretation as well as on context knowledge. For example, context knowledge can be used if individual baufix® elements are likely to be part of an assembly due to their proximity. In this case assemblage knowledge provides a measure how well a specific element would fit into an assembly made out of the surrounding elements.

2.2.3 Assembly Recognition

An important functionality needed in the Situated Artificial Communicator is the ability to recognize assemblies made of elementary baufix® objects. Based on the functional model for assemblies introduced in Section 2.2.1, a knowledge-based approach to recognize baufix®-assemblies has been implemented by C. Bauckhage [6]. To realize the assembly recognition, the recursive functional model was implemented as a semantic network using the semantic network language ERNEST [86]. The input data are elementary baufix® objects detected in the image data by the object recognition approach described in the previous section.

To recognize the assembly structure of the assembly depicted in Fig. 2.5, the labeled cluster of regions shown in Fig. 2.9(a) is sequentially analyzed as follows: since a bolt is an obligatory component of every assembly in our scenario, the analysis of the cluster starts with a region representing a bolt. After instantiating the bolt as the BOLT_PART of an assembly in the semantic network, the region is marked `considered` and the neighboring region is examined. If it depicts a miscellaneous object, the corresponding concept MISC_PART is instantiated and the region is marked `considered`, too. This process will be iterated until a region representing a nut-type object is found. If in such a way all obligatory parts of an assembly were found but there are still regions in the cluster which have not been marked yet, it is heuristically determined what kind of role the just detected assembly may play as a subassembly and the analysis is continued correspondingly.



(a) (b) (c)

Figure 2.9: (a) The results from object recognition depicted by colored polygons surrounding the regions belonging to the elementary objects for the example assembly in Fig. 2.5; (b) Intermediate result of assembly recognition after analyzing the second bolt in the assembly. The different hierarchical levels of the two subassemblies are depicted by a white and a black polygon; (c) Final results of assembly recognition.

An intermediate result as well as the final assembly structure are depicted visually in Fig. 2.9(b) and Fig. 2.9(c). The final recognition result corresponds to the hierarchical

description in Fig. 2.5(b), Fig. 2.6(b): there are three shapes, each including parts of a subassembly. The innermost white shape ($MISC_1$) includes the regions representing the bolt, the bar, and the blue cube and belongs to the lowest level of the hierarchy. The surrounding black shape (NUT_1) includes this subassembly as well as the red bolt and the green cube. The outermost white shape includes this subassembly and the yellow bolt and represents the highest level of the hierarchy, i.e., the whole assembly.

While the basic processing strategy for assembly recognition outlined above is quite simple, several implementational problems have to be solved for realizing a working algorithm. For example, if there is more than one neighboring region, topological considerations have to be taken into account to search for the correct region. Other problems include the perspective distortion of large objects like, e.g., bars with five or seven holes and the skipping of small unknown regions due to shades or not correctly recognized objects. For a complete description of the assembly recognition method and further details on the implementation, the reader is referred to [6].

2.3 Detecting Actions by Symbolic Inference

After introducing the Situated Artificial Communicator domain and the object and assembly recognition algorithms for this domain, we now turn to the description of the approach for recognizing the construction actions necessary for building assemblies. In the Situated Artificial Communicator domain, the construction actions consist of picking baufix® parts and mounting them together. These actions are therefore essentially manipulative gestures as defined in Section 1.1. As it is not necessary to know *how* the parts were moved, it is possible to base the recognition approach solely on symbolic information representing changes of the object positions to detect the actions that must have been executed.

In the Situated Artificial Communicator domain, parts are not simply moved but instead connected together. The connecting of parts is carried out by the two hands and is only possible if the two parts in the hands are 'compatible'. Therefore, not only the temporal appearance and disappearance of parts but also the actual contents of the hands have to be analyzed in a symbolic action detection approach for the Situated Artificial Communicator. For this purpose, a *two-hand model* is introduced. It is assumed that the parts in the scene are recognized by an object/assembly recognition algorithm taking the actual view of the scene as input. Monitoring the detected objects over time gives the appeared and disappeared parts in the scene. Together with the current state of the two-hand model, a rule-based approach is used to infer the actions that have led to the observed new/disappeared parts in the scene. In Fig. 2.10 the different components for performing action detection are depicted, the details will be described in the following.

Out of the recent symbolic approaches to action recognition described in Section 2.1, the approach of Kuniyoshi [60] is closest to our work. Similar to our work, it explicitly defines an environment model to hold the current scene state. However, his aim is the

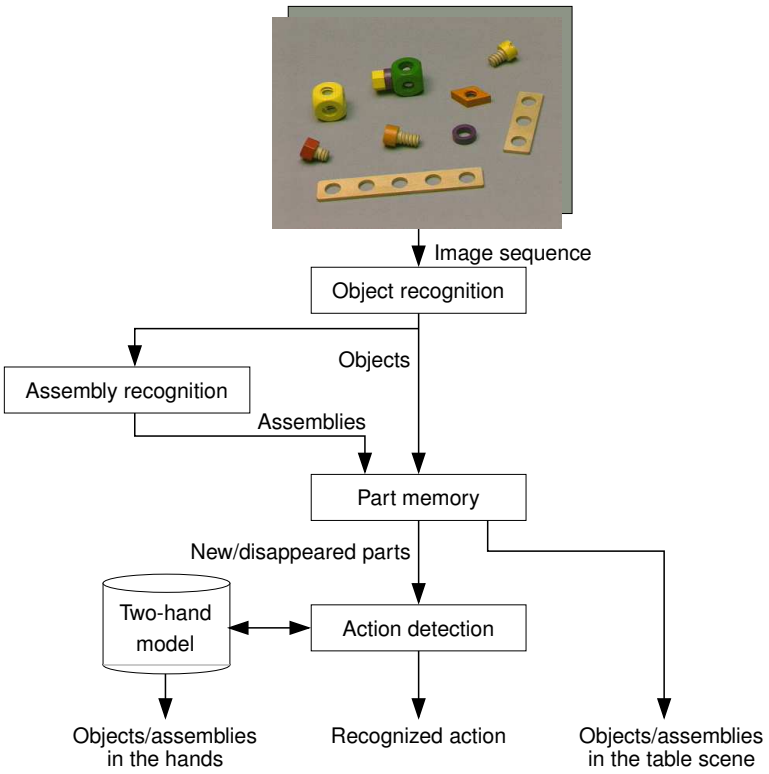


Figure 2.10: A schematic drawing of the relations between the different modules for action detection.

development of a generic action recognition theory. Therefore, an appropriate control mechanism and specific image processing routines are directly integrated into his framework. The detection of relevant events in the scene is implicitly accomplished by the specialized visual feature detectors of the currently active environment elements. While the basic idea is therefore very generic, the implementation becomes very specific due to the visual feature detectors. As our approach aims at the recognition of actions in a specific scenario, we can use symbolic events that are detectable based on the development of the overall scene without performing additional processing of the visual input.

2.3.1 Two-Hand Model to Capture Actual Hand Contents

For the detection of actions in the construction scenario we implemented the actions *New*, *Pick*, *Place*, *Put* and *Screw*. As our approach is based on symbolic data alone, these actions can be recognized independently of the constructor, i.e., differences between persons in the way how they handle the baufix® parts do not influence the recognition result. Therefore, this approach can also be used to recognize actions that are executed by a robot if the robot is compatible to the two-hand model, i.e., if it has two manipulators that can hold baufix® parts or assemblies and connect them together.

To have a suitable representation of the environment for the construction task, we introduce a two-hand model that is used to hold the current states of the two hands. Each of the two states in the two-hand model contains information what part, partial assembly or complete assembly is currently in this hand. Through representing the contents of the hands explicitly in the hand states, the symbolic action detection can use this information for inferring actions. The following gives the formal definition for the two identical sets of hand states in the two-hand model:

Contents of hand state 1: Empty | BOLT_PART | MISC_PART | NUT_PART | (BOLT_PART - MISC_PART^{+2}) | ASSEMBLY

Contents of hand state 2: Empty | BOLT_PART | MISC_PART | NUT_PART | (BOLT_PART - MISC_PART^{+2}) | ASSEMBLY

The hand states are updated each time an action is recognized that changes the contents of the hands. Besides supporting the action detection, the two-hand model allows other modules to incorporate the context knowledge available from the hand contents. This is used in the Situated Artificial Communicator domain by the vision-speech interaction module to resolve instructions that refer to parts in the hands of the constructor. For example, resolving the instruction 'Put the bar on the red bolt' requires a bar and a bolt to be available. The parts currently laying on the table are provided by the part memory (see Fig. 2.10). However, if the red bolt has already been taken by the constructor, it is no longer visible in the image of the static camera looking at the table. Consequently, it is not contained in the part memory. Only through observing the actions in the scene and maintaining the current contents of the two hands in the two-hand model, it is possible to resolve instructions referring to parts already taken by the constructor like the red bolt.

2.3.2 Detecting Scene Changes with the Part Memory

Detecting the appearance and disappearance of parts is done by monitoring over time the object and assembly recognition results. The object recognition results are generated based on the images from the static camera looking at the table. The results are provided in the form of a *stream* [26] that is updated after each execution of the object recognition algorithm described in Section 2.2.2. Based on the object recognition results, the assembly recognition is carried out if several new objects are detected at once. The results of object and assembly recognition are monitored by a *part memory* that stores all parts and provides the new and disappeared parts to the action detection (see Fig. 2.10).

[2] This represents a partial assembly which is not yet tightened with a NUT_PART to form a complete assembly. The plus operator indicates the possibility to put at least one MISC_PART on a BOLT_PART limited by its length.

Unfortunately, recognition results vary significantly between two consecutive images. Besides normal removal and insertion of parts in the scene by the user, variations in the detected parts occur due to

1. some parts being temporarily occluded due to a hand moving through the field of view of the camera

2. erroneous elementary parts being temporarily detected by the object recognition algorithm due to, for example, shades from a moving hand

3. varying object labels for an elementary part due to, for example, noisy image data

To prevent detecting a changed part in the first two cases, the recognition results are low-pass filtered in the part memory. Unfortunately, in the third case using a low-pass filter may only lead to a slower variation of the object labels, but it may not completely eliminate the instability. To circumvent this problem, the object labels of a new image feature, i.e., in our case an image region, are accumulated over the first N_{rec} recognition phases. The dominant object label in this sequence is taken to be the 'true' object label and the part memory notifies the action detection module about a newly appeared part X with the message `X` → `new`. If subsequently a different object label is received for a region of the same color and at approximately the same position, the part memory replaces it with the stored 'true' object label.

If the part X is missing in the stream of recognition results for more than T_{skip} seconds, the part memory notifies the action detection module about the disappeared part with the message `X` → `disappeared`. In Fig. 2.11 an UML sequence diagram [14] depicts the module interactions over time for detecting a *Pick (X)* action based on a part that has disappeared for longer than T_{skip} in the object recognition results.

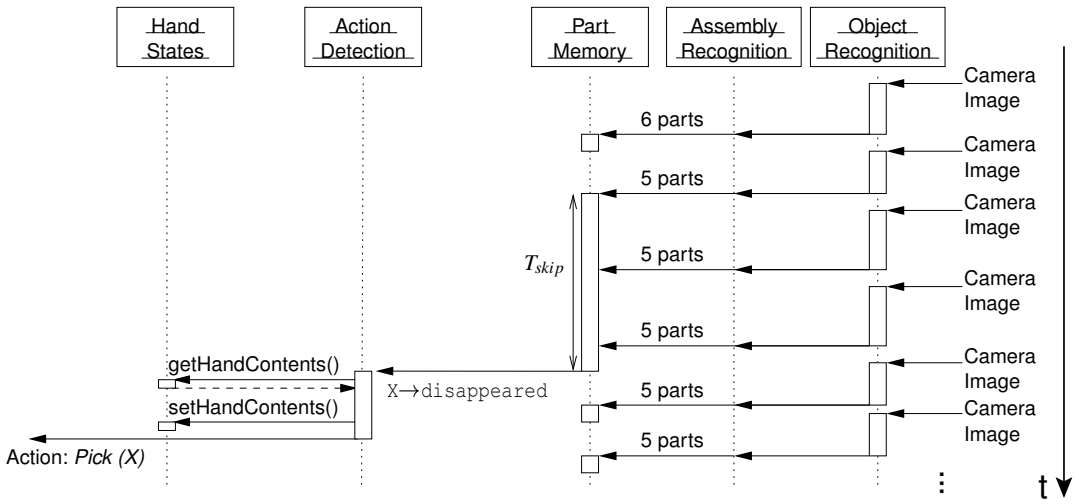


Figure 2.11: An UML sequence diagram showing the module interactions over time.

Notice that the time necessary for object recognition is not constant as the processing times of the different segmentation and classification algorithms depend on the image contents, e.g., how many homogeneously colored image regions are present. The rule-based detection of actions based on the described messages provided by the part memory will be covered in the following section.

2.3.3 Inferring Actions from Scene Changes

The two-hand model is used together with the information about new and disappeared parts extracted by the part memory to infer the actions occurring in the scene. The rules for action detection operate on the states of the two-hand model and the changes in the scene, i.e., on the table. If a change in the scene is detected resulting in a part appearing or disappearing, a rule with preconditions matching the observed scene change and the current hand states is searched. If a suitable rule is found, that action is hypothesized and the hand states are changed accordingly. Figure 2.12 shows the rules for the actions currently implemented in our system. The presentation is similar to the notation for planning operators. The functional properties (e.g., BOLT_PART) of all complex parts are extracted following the definition given in Section 2.2.1.

For example, two preconditions must hold to infer the *Pick (X)* action: a part X must be on the table and one hand must be empty. If the part memory detects a part X to be missing in the stream, it notifies the action detection module about the disappeared part with `X` $\rightarrow$ `disappeared`. Now the *Pick (X)* action is inferred and the state of the hand model is changed to capture the new state of the hand holding part X.

Note that the *Put* and *Screw* actions are not directly linked to changes in the scene because mounting parts together does not lead to new or disappeared parts on the table. Therefore, the *Put* and *Screw* actions are not directly observable and can only be inferred if the next *Pick* action has happened:
If both hands contain parts X, Y and another part Z disappears, a *Put* or *Screw* action is inferred if the two parts in the hands confirm to the preconditions of one of these actions. In this case, it is assumed that the parts have been connected together to form an assembly (X, Y). Inferring the *Put* or *Screw* action changes the two-hand model: now one hand contains the (partial) assembly (X, Y) and the other hand is empty. Consequently, the preconditions of the *Pick* action are satisfied and now the empty hand can take the disappeared part Z.

The strategy described above is also applicable if a new assembly is detected on the table but none of the hands contains a complete assembly that can be placed on the table[3]. If the contents of the two hands can be connected together to form a complete assembly, i.e., one hand holds a BOLT_PART and one hand holds a NUT_PART, the appropriate *Put* or *Screw* action is inferred. Now the one hand holding the just completed assembly can execute a *Place* action.

[3]Here a complete assembly means an assembly where all MISC_PARTs are fastened with a NUT_PART.

New (X)
Preconditions: ¬ (hand: X) ∧ `X` → `new`
Effects: X on_table

Pick (X)
Preconditions: hand: Empty ∧ `X` → `disappeared`
Effects: hand: X ∧ ¬ (X on_table)

Put (X,Y)
Preconditions: hand_1[a]: X (BOLT_PART) ∧
hand_2[a]: Y (MISC_PART)
Effects: hand_1[a]: (X,Y) ∧ hand_2[a]: Empty

Screw (X,Y)
Preconditions: hand_1[a]: X (BOLT_PART) ∧
hand_2[a]: Y (NUT_PART)
Effects: hand_1[a]: (X,Y) ∧ hand_2[a]: Empty

Place (X)
Preconditions: hand: X ∧ `X` → `new`
Effects: hand: Empty ∧ X on_table

[a]The notation hand_1 and hand_2 is only used to indicate that two different hand states are used, each of the two states of the two-hand model can be hand_1 or hand_2.

Figure 2.12: The rules for inferring actions based on the actual state of the two-hand model and the changes in the scene.

To give an impression of the symbolic action recognition in the Situated Artificial Communicator setting, Fig. 2.13 shows an example for the temporal development of a table scene. In the left column the scene images obtained with the table camera together with the recognized object labels are shown. In the middle column the detected object changes for the left image compared to the previous image are listed. Based on these object changes, the recognized actions as well as the resulting hand states are shown. The right column depicts a virtual representation [99] of the actual scene. All parts below the horizontal line in this virtual representation represent parts currently on the table, i.e., contained in the part memory. For representing parts contained in the two hand states of the two-hand model, two positions above the line at the top of the figure are used.

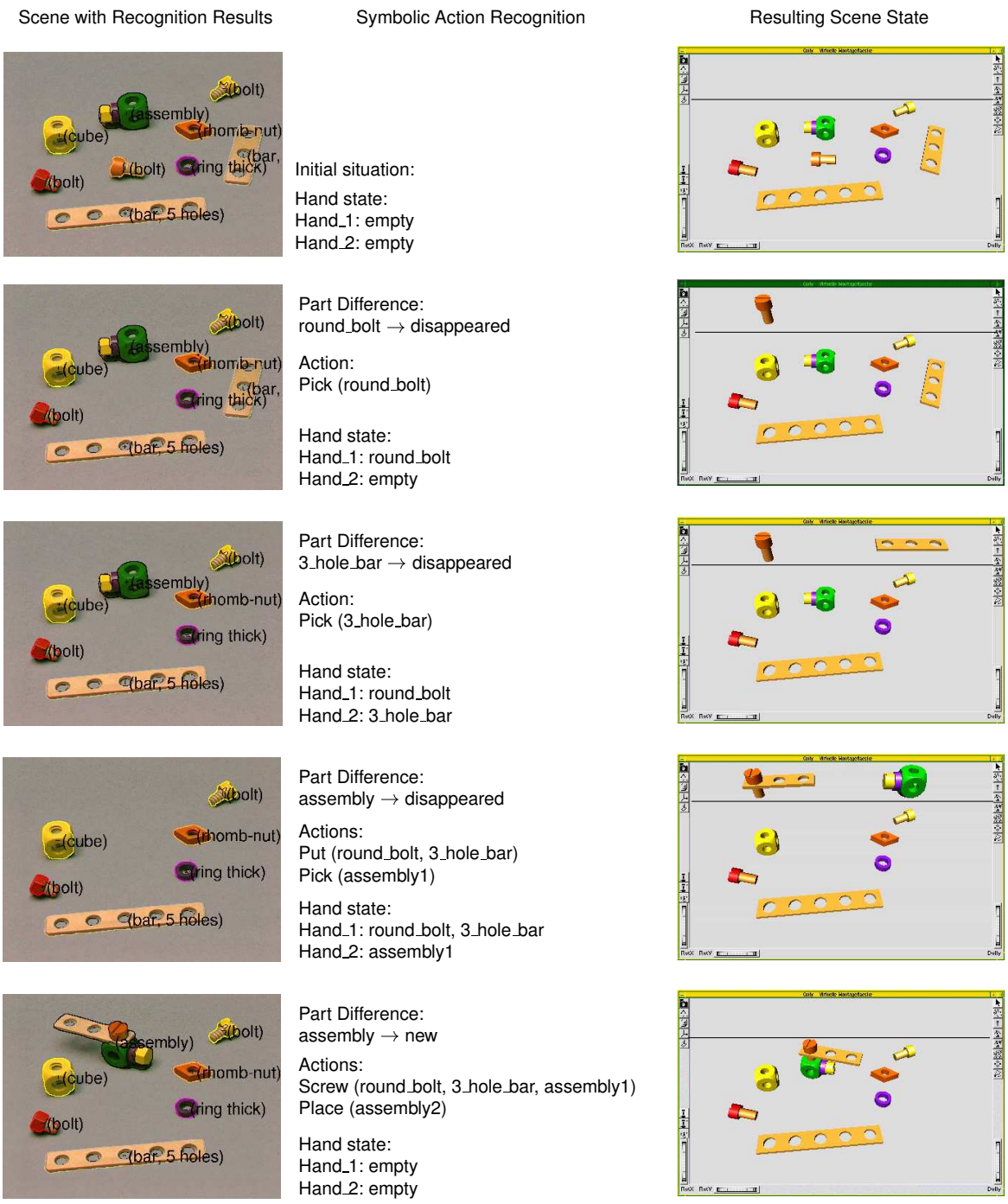


Figure 2.13: An example for an action sequence resulting from part changes during the temporal development of a scene. A virtual representation [99] is used to visualize the actual scene as hypothesized by the symbolic action recognition. The parts below the horizontal line are currently laying on the table, while the parts above the line indicate the contents of the two-hand model.

2.3.4 Limitations of the Symbolic Approach

Since the described symbolic action detection is based solely on the changes of the parts in the scene, the user must obey some restrictions to avoid erroneously detected actions:

- Each hand may hold only one part or (partial) assembly. Although the flexibility of the human hand allows to hold several parts at the same time, the human must follow this restriction to enable the system to detect his actions correctly.

- No parts may be put down outside the visible table scene. This *closed-world assumption* is necessary as the system cannot detect if a part is put down outside the field of view of the table camera and will therefore assume that the part is still in the hands of the human. Although removing parts from the scene is therefore not possible, the *New (X)* action (see Fig. 2.12) allows the adding of parts[4] resulting in a partial closed-world restriction.

If the human obeys these restrictions and the new/disappearing parts in the scene are reliably detected, the rule-based action detection works correctly. However, as noted in Section 2.3.2 the results from object recognition may vary significantly over time. The developed part memory can reduce the influence of varying object recognition results, but it cannot assure a correct detection of new/disappeared parts. If a part is not recognized and therefore missing in the object recognition results longer than T_{skip}, it will be detected as 'disappeared' and a *Pick* action will be recognized. The second problem besides the stability of the object recognition results is the quality of the results. If a part is labeled incorrectly as a different part with another function, a *Put* or *Screw* action cannot be recognized. If the user connects this part to another part, the action detection will fail to recognize the action as the two part functions are incompatible. In this case the hand states are not changed and the system generates an error message.

2.4 Summary

In this chapter, the Situated Artificial Communicator domain for assembly construction and the object and assembly recognition algorithms for this domain were introduced. We presented our method for symbolic action detection that is based on the results from the vision-based object and assembly recognition algorithms. Through extracting the changes of the scene over time and applying a rule-based approach, actions are hypothesized. To enable a correct action detection, the human actor must obey the restrictions imposed by the rule-based approach and the scene changes have to be detected reliably. The latter is a crucial point as our action detection method is implemented

[4]This rule allows to add a new part as long as no similar parts (same type and color) are in the hands during adding of a part. Otherwise the system will hypothesize that the hand holding a similar part placed this part on the table with a *Place* action and is now empty.

separately from symbol grounding. Using the symbolic information available from object and assembly recognition avoids to perform symbol grounding and the associated computational cost but makes our approach depending on the quality of these recognition algorithms. It would be desirable to have an action detection system that is not as sensitive to object/assembly recognition errors and uses additional cues, e.g., visual observations of the hands, to support the action recognition task.

3 Recognizing Human Gestures

An alternative to the symbolic action detection presented in the previous chapter is the recognition of gestures based on sensory trajectory data. Following Bobick's categorization of human motions (see Section 1.1), gestures that can be completely described by trajectory information are activities. To recognize activities based on their trajectories, we will first introduce a probabilistic model for describing human activities. This model is the underlying assumption on which all of today's dominant probabilistic recognition techniques are based. Subsequently, three important approaches to activity recognition will be presented in more detail. In Chapter 6 we will extend a probabilistic approach for activity recognition by integrating symbolic information to obtain a structured framework for action recognition. We will show there that incorporating the symbolic context in the proposed framework allows for the successful recognition of manipulative gestures in a wide range of domains.

3.1 Modeling Human Activities

Let us assume that we want to model a human activity with statistical methods. The human activity is given as a sequence Z_T of T observation vectors[1] at discrete time steps where each observation vector $\mathbf{z}_t$ consists of m measurement values:

$$Z_T = (\mathbf{z}_1, \ldots, \mathbf{z}_t, \ldots, \mathbf{z}_T) \quad with \quad \mathbf{z}_t = (z_{t1}, \ldots, z_{tm}), \quad z_{ti} \in \mathbb{R} \tag{3.1}$$

One well-known method for statistical modeling of sequences of observations originating from a parametric random process are *Markov processes*. The underlying idea is that the parameters of the stochastic process leading to the observed temporal sequence Z_T can be estimated in a well-defined manner. For each time step t, the state of the stochastic process giving rise to the observation $\mathbf{z}_t$ is denoted by the random variable q_t:

$$q_t \in \mathbb{R} \tag{3.2}$$

For clarity of the presentation, we assume that q_t is a scalar variable, but in general this could be a vector. As an example, consider the activity 'waving' depicted schematically

[1] In this thesis scalar values are represented by normal lower case letters while bold lower case letters denote vectors. Normal upper case letters denote matrices and bold upper case letters represent a sequence of matrices. An exception to this notation are the upper case letters T for the length of a time sequence as well as M and N for numbers.

in Fig. 3.1. If the human activity, i.e., his hand gesture, is observed with a camera, the position of the hand could be extracted from every image of the image sequence showing the activity. The measurement vector $\mathbf{z}_t$ therefore consists of two measurement values, the x- and y-position of the hand in image coordinates. The measured hand position can be related to a variety of different types of states causing this measurement like, e.g., the current arm muscle activation or the angle between the body and the arm. In Fig. 3.1 the angle is assumed to be the state q_t of the stochastic process leading to the observation of the hand position.

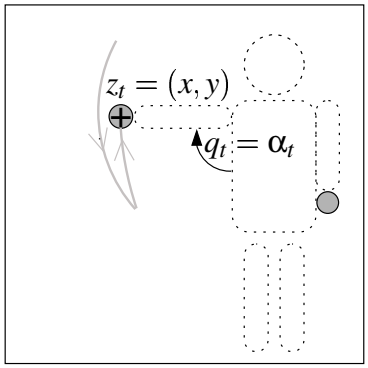


Figure 3.1: Measurement vector $\mathbf{z}_t$ (hand position) and process state q_t (arm angle) of the human activity 'waving'.

The development of the state q_t over time is described by the *system dynamics*. If the *probability density function* (pdf) of the current state depends only on the previous state q_{t-1} and not on the complete state history $\Omega_{t-1} = \{q_1, q_2, \ldots, q_{t-1}\}$, the system dynamics are:

$$p(q_t|q_{t-1}) = p(q_t|\Omega_{t-1}) \tag{3.3}$$

This is the special case of a first-order Markov process. In a k-order Markov process, the current state depends on the k most recent states. To give a practical example, the system dynamics for the 'waving' activity could model the fact that the arm angle varies at a constant rate, i.e., the current value can be calculated by extrapolating the previous value using its derivative.

At each time instance t, the state pdf can now be calculated from the previous state pdf and the system dynamics:

$$p(q_t) = \int p(q_t|q_{t-1})p(q_{t-1})\,\mathrm{d}q_{t-1} \tag{3.4}$$

This equation does not incorporate any feedback from actual measurements, i.e., it is an open-loop calculation without any 'grounding' in actual observations. For successful tracking of human activities, we have to incorporate the sequence of measurements Z_T that are extracted from observing the human activity. Instead of the density $p(q_t)$ from

Eq. 3.4, we can calculate the *a priori* density $p(q_t|Z_{t-1})$ and the *a posteriori* density $p(q_t|Z_t)$ before and after incorporating the measurement at time t, respectively. Let us assume that the conditional observation density $p(\mathbf{z_t}|q_t)$ for a measurement at time t is independent of the observation history Z_{t-1}, and depends only on the current state pdf:

$$p(\mathbf{z}_t|q_t) = p(\mathbf{z}_t|q_t, Z_{t-1}) \tag{3.5}$$

This allows us to determine the *a posteriori* density $p(q_t|Z_t)$ from the *a priori* density $p(q_t|Z_{t-1})$ using Bayes' rule:

$$\begin{aligned} p(q_t|Z_t) = p(q_t|\mathbf{z}_t, Z_{t-1}) &= \frac{p(\mathbf{z}_t|q_t, Z_{t-1})\; p(q_t|Z_{t-1})}{p(\mathbf{z}_t|Z_{t-1})} \\ &= c\; p(\mathbf{z}_t|q_t, Z_{t-1})\; p(q_t|Z_{t-1}) \\ &= c\; p(\mathbf{z}_t|q_t)\; p(q_t|Z_{t-1}) \end{aligned} \tag{3.6}$$

In this equation $c = 1/p(\mathbf{z}_t|Z_{t-1})$ is a normalization factor independent of q_t, and $p(q_t|Z_{t-1})$ is the prior from the accumulated observation history up to time $t-1$. This is equivalent to the posterior at the previous time step $p(q_{t-1}|Z_{t-1})$ predicted to the actual time step using the system dynamics (Eq. 3.3):

$$p(q_t|Z_{t-1}) = \int p(q_t|q_{t-1}) p(q_{t-1}|Z_{t-1})\, \mathrm{d}q_{t-1} \tag{3.7}$$

Based on this model, a human activity can be tracked over time by calculating at every time step first the *a priori* density $p(q_t|Z_{t-1})$ from Eq. 3.7 using the system dynamics, and then evaluating the *a posteriori* density $p(q_t|Z_t)$ with Eq. 3.6 based on the new measurement $\mathbf{z}_t$. This method to track the state probability density function over time with integration of measurements is known as *recursive Bayesian filter* (see Fig. 3.2).

Note that tracking and recognition are very similar, as recognizing a human activity amounts to successful tracking of a motion with a model of the system dynamics that describes the activity (Eq. 3.3). Stated differently, if the tracking of the motion failed, the model of the system dynamics must have been inadequate to describe the activity. Consequently, if several different models of the system dynamics are used, the model belonging to the Bayesian filter with the highest overall probability indicates the human activity that was executed, i.e., this activity has been recognized.

As a practical example, consider again the 'waving' activity sketched in Fig. 3.1. If we propagate two Bayesian filters with different system dynamics, one assuming a steady increase in the angle value and one assuming a steady decrease, the differentiation between the 'up' and 'down' motion can be done based on the overall probability of the two filters.

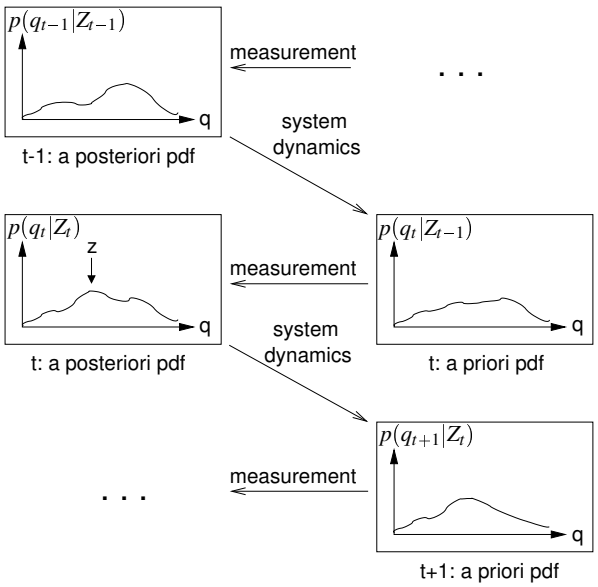


Figure 3.2: The individual steps of a recursive Bayesian filter.

3.2 Recognizing Activities

Depending on the properties of the state distribution, the system dynamics, and the measurement process, different algorithms have been proposed to perform recursive Bayesian filtering. The three dominant techniques used to track/recognize human activity that are based on the probabilistic formulation given above will be reviewed in the following, namely *Kalman filters* [5], *Hidden Markov Models* (HMM, [79]), and *particle filtering* [2]. A special form of particle filtering is better known in the computer vision community under the name CONDENSATION (conditional density propagation) algorithm [48].

3.2.1 Kalman Filters

In a Kalman filter [5], the state pdf $p(q_t)$ is modeled as a single Gaussian with mean μ_t and covariance σ_t. In general, the state pdf of a Kalman filter can be a vector, but for simplicity we will consider here only the one-dimensional case (see Fig. 3.3).

The state q_t is hidden, i.e., it is not directly observable, and it is related to the observation z_t by a transfer function h_t that is a scalar value for the one-dimensional case:

$$z_t = h_t q_t \tag{3.8}$$

Given a state q_{t-1} at time $t-1$, predictions of its Gaussian parameters $(\hat{\mu}_t, \hat{\sigma}_t)$ for time t can be calculated using the system dynamics a_t and the uncertainty of the state

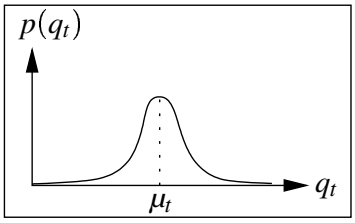


Figure 3.3: The unimodal Gaussian state probability density function modeled by a Kalman filter.

prediction modeled with the covariance σ_q:

$$\hat{\mu}_t = a_t \mu_{t-1} \tag{3.9}$$
$$\hat{\sigma}_t = \sigma_{t-1} + \sigma_q \tag{3.10}$$

Similarly, the observation can be predicted to time t using the transfer function h_t and the uncertainty of the measurement process modeled with σ_z:

$$\hat{z}_t = h_t q_{t-1} + \sigma_z \tag{3.11}$$

The parameters of the Gaussian density function can now be propagated over time by fusing the predictions of the parameters $(\hat{\mu}_t, \hat{\sigma}_t)$ and the prediction of the observation $\hat{z}_t$ with the actual observation z_t using the *Kalman gain* k_t:

$$\mu_t = \hat{\mu}_t + k_t(z_t - h_t \hat{\mu}_t) \tag{3.12}$$
$$\sigma_t = \hat{\sigma}_t - k_t h_t \hat{\sigma}_t$$

In this fusion process, the predictions and observations are weighted by their estimated uncertainties. To minimize the a posteriori error covariance σ_t in Eq. 3.12 (see [5] for details), the Kalman gain k_t is chosen to[2]:

$$k_t = \frac{\hat{\sigma}_t h_t^T}{h_t \hat{\sigma}_t h_t^T + \sigma_z} \tag{3.13}$$

The Kalman filter as introduced above is an optimal estimator if the state density and the observation density are Gaussian and, consequently, all densities stay Gaussian during the propagation. In this case, the Kalman filter results in an optimal estimator for the error variance, and the state mean is equal to the most probable state.

However, in real applications the state density often exhibits multiple modes due to, e.g., noise in the observations. The Kalman filter is not able to track state densities containing multiple modes. Therefore, several extensions to the standard Kalman filter

[2]For a state vector $\mathbf{q}$, the transfer function h_t^T would need to be the matrix transpose H_t^T of the transfer function H_t but in the one-dimensional case is $h_t^T \equiv h_t$.

have been proposed. These approaches for tracking of multiple hypothesis represent multimodal distributions with a series of Kalman filters where each filter is responsible for one hypothesis (see e.g. [5, 83]). In these 'multiple hypothesis tracking' approaches, specific functionalities are needed to add and remove Kalman filters if new hypotheses arise or old hypotheses can be discarded.

3.2.2 Hidden Markov Models

While Kalman filters are used primarily for tracking, Hidden Markov Models (HMM) are a well-known probabilistic technique for recognizing human activities. After the development of the underlying theory in the 1970's, the first important application area was the automatic recognition of speech [79]. Following the successful use of HMM's for speech recognition, researchers have started to use this technique for the analysis of vision data. The earliest approach was probably a system by Yamamoto [106] for recognizing six different strokes of a tennis player. Over the years, researchers have proposed numerous extensions to the basic theory in order to enable the application of HMM's for specific recognition problems (cf. e.g. [17, 37, 11]). To allow for an easy comparison with Kalman filters and Particle filtering, we will concentrate in the following on the basic algorithm.

An HMM can be described as a finite state machine with probabilistic state transitions. The current state of an HMM consisting of N states will be denoted here with the multinomial variable s:[3]

$$s = i \quad \text{with} \quad i \in \{1 \dots N\} \tag{3.14}$$

Depending on the current state, the HMM generates an observation symbol with an emission probability. Only the observation symbols can be observed while the actual state s of an HMM is unknown, hence the name *hidden* Markov model. The observations emitted by an HMM state can be either discrete or continuous. Following our model of a human activity as outlined in Section 3.1, we will assume continuous observation vectors $\mathbf{z}_t$. Figure 3.4 depicts the discrete modeling for the state probability of the process generating the observations $\mathbf{z}_t$.

The introduction of discrete state positions q^s is the only conceptual difference to the Kalman filter, where q has continuous values and represents the mean of the state distribution and the associated covariance matrix. The discrete nature of the state q in an HMM has large implications: different from the Kalman filter, the HMM can deal with nonlinear evolution of the state of the stochastic process. However, the underlying process generating the observations must have discrete states to perfectly model it with an HMM. Assuming a stochastic process taking on discrete states, the integration over the state density $p(q_t)$ becomes a summation over the individual state probabilities

[3] In the classical notation of HMM's, the multinomial label used to denote the individual states is typically the letter 'q'. As this conflicts with our use of q to denote the state of the underlying process generating the observations, we use s to denote the HMM states.

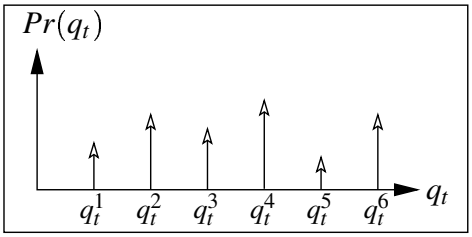


Figure 3.4: The discrete state probability for six states modeled by an HMM.

$Pr(q_t^s)$. Now the prediction (Eq. 3.7) and update (Eq. 3.6) equations of the recursive Bayesian filter can be written as:[4]

$$Pr(q_t|Z_t) = \sum_{s=1}^{N} Pr((q_t = q_t^s|Z_t) \tag{3.15}$$

$$Pr(q_t|Z_{t-1}) = \sum_{s=1}^{N} Pr(q_t = q_t^s|Z_{t-1}) \tag{3.16}$$

The state-specific probabilities can be obtained from

$$\begin{aligned} Pr(q_t = q_t^s|Z_t) &= \frac{Pr(\mathbf{z}_t|q_t = q_t^s, Z_{t-1})\ Pr(q_t = q_t^s|Z_{t-1})}{Pr(\mathbf{z}_t|Z_{t-1})} \\ &= \frac{Pr(\mathbf{z}_t|q_t = q_t^s)\ Pr(q_t = q_t^s|Z_{t-1})}{Pr(\mathbf{z}_t|Z_{t-1})} \\ &= \frac{Pr(\mathbf{z}_t|q_t = q_t^s)\ Pr(q_t = q_t^s|Z_{t-1})}{\sum_{j=1}^{N} Pr(\mathbf{z}_t|q_t = q_t^j)\ Pr(q_t = q_t^j|Z_{t-1})} \end{aligned} \tag{3.17}$$

with

$$Pr(q_t = q_t^s|Z_{t-1}) = \sum_{j=1}^{N} Pr(q_t = q_t^s|q_{t-1} = q_t^j)\ Pr(q_{t-1} = q_t^j|Z_{t-1}) \tag{3.18}$$

The preceding equations are only applicable to perform recursive Bayesian filtering based on observations from a discrete process. An HMM can also be used to approximate a continuous state pdf if the N positions in the state space are chosen carefully and the state pdf exhibits Gaussian modes. However, to obtain a good approximation of a continuous state space, a sufficient number of states is needed at the expense of a large computational cost.

Let us now turn to the practical use of this framework for recognition purposes. In the classical notation [79], an HMM consists of a tuple $\lambda = (A, \mathbf{b}, \pi)$ where A is an

[4] In the equations in this section, we will use the notation $Pr(\cdot)$ to emphasize the discrete nature of the probabilities $p(\cdot)$.

$N \times N$ matrix containing the individual state *transition* probabilities a_{ij}, and $\mathbf{b}$ is an N dimensional vector with the *emission* probability density functions for all states. The notation used in the preceding equations is related to these HMM parameters by:

$$Pr(q_t = q_t^i | q_{t-1} = q_t^j) = \quad a_{ij} \qquad \forall \quad i, j \in \{1 \dots N\} \tag{3.19}$$

$$Pr(\mathbf{z}_t | q_t = q_t^i) = \quad b_i(\mathbf{z}_t) \qquad \forall \quad i \in \{1 \dots N\} \tag{3.20}$$

For initialization of the stochastic process, the N dimensional vector π contains the initial state probabilities:

$$Pr(q_1 = q_1^i | Z_0) = \pi_i \quad \forall \quad i \in \{1 \dots N\} \tag{3.21}$$

To generate an HMM, the parameter values $(A, \mathbf{b}, \pi)$ have to be determined. Usually the parameter values are learned automatically from training examples via the Expectation-Maximization (EM) [22] optimization method. The application-dependent design decision that has to be made by the designer before the training is to define the number N of different states and the topology of the state transition network, i.e., which transitions a_{ij} are allowed.

While the 'transition' and 'emission' probabilities in an HMM recovered by the training algorithm are interpretable by a human system designer, the actual states $s = 1 \dots N$ to which the transition and emission probabilities belong do not always have a clear interpretation. Even with a priori knowledge about the characteristics of the underlying process generating the observations, it is difficult for a system designer to incorporate this knowledge into the framework.

The recognition of a human activity is performed by computing for all HMM's, where each HMM represents one model, the most likely state sequence using the Viterbi algorithm [79]. Computing the most likely state for each time step t individually does not necessarily give the same result, as not all transitions between different states may be allowed. Given an observation sequence, the most likely state sequence of a specific HMM recovered by the Viterbi algorithm allows us to compute the associated maximal emission probability of this HMM. The recognized activity is then given by the HMM that generates the largest emission probability. Note that for any input sequence this recognition scheme provides only the information which of the HMM's best explains the observed data, but there is no mechanism to indicate if an input is unknown. To avoid false positives, a separate rejection method has to be implemented, for example by using thresholds on the probability values for the most likely state sequence.

3.2.3 Particle Filtering

One alternative to HMM's that recently gains increasing interest are approaches applying particle filter algorithms [48, 2]. Particle filtering is a technique for implementing a recursive Bayesian filter by Monte Carlo simulations. The basic idea is to represent the posterior density $p(q_t|\mathbf{z}_t)$ from Eq. 3.6 by a set of N weighted samples, the *particles*, and to compute estimates of the posterior based on these weighted samples (see Fig. 3.5). Different from Kalman filters and HMM's, a Particle filter allows us to model a non-linear, non-Gaussian state pdf by approximating it with a set of weighted samples.

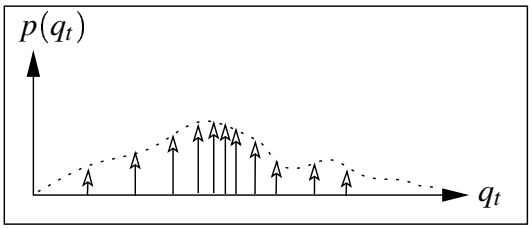


Figure 3.5: The state probability density function modeled by a particle filter.

Each sample contains a vector in the m-dimensional feature space that is suitable to describe the motion:

$$\mathbf{s}^{(i)} = (\mathbf{x}) \quad with \quad \mathbf{x} = (x_1, \ldots, x_m) \tag{3.22}$$

Together with its associated weight $\pi_t^{(i)}$, each sample $\mathbf{s}_t^{(i)}$ represents a support point in the posterior pdf. The complete set of samples with their weights gives the sample set:

$$\left\{(\mathbf{s}_t^{(1)}, \pi_t^{(1)}), \ldots, (\mathbf{s}_t^{(N)}, \pi_t^{(N)})\right\} \tag{3.23}$$

For $N \to \infty$, the overall *a posteriori* probability $p(\Omega_t|Z_t)$ at time t given the sequence of states $\Omega_t = \{q_1, q_2, \ldots, q_t\}$ and samples $S_t^{(i)} = \{\mathbf{s}_1^{(i)}, \mathbf{s}_2^{(i)}, \ldots, \mathbf{s}_t^{(i)}\}$ can then be represented by [2]:

$$p(\Omega_t|Z_t) \approx \sum_{i=1}^{N} \pi_t^{(i)} \delta(\Omega_t - S_t^{(i)}) \tag{3.24}$$

The sequence of observations $Z_t = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_t\}$ is implicitly contained in Eq. 3.24 as for every time step t the observation $\mathbf{z}_t$ is used for calculating the weights $\pi_t^{(i)}$. The temporal propagation of the N weighted samples is carried out through first propagating the samples to the new time step $t+1$ and then updating the weights. The state propagation usually contains a deterministic drift based on the system dynamics and some diffusion resulting from the uncertainty in the system dynamics (see Fig. 3.6).

The weights are chosen based on a technique called *Importance Sampling*: Let us assume that there is a probability density $p(y)$ that is difficult to sample but for which

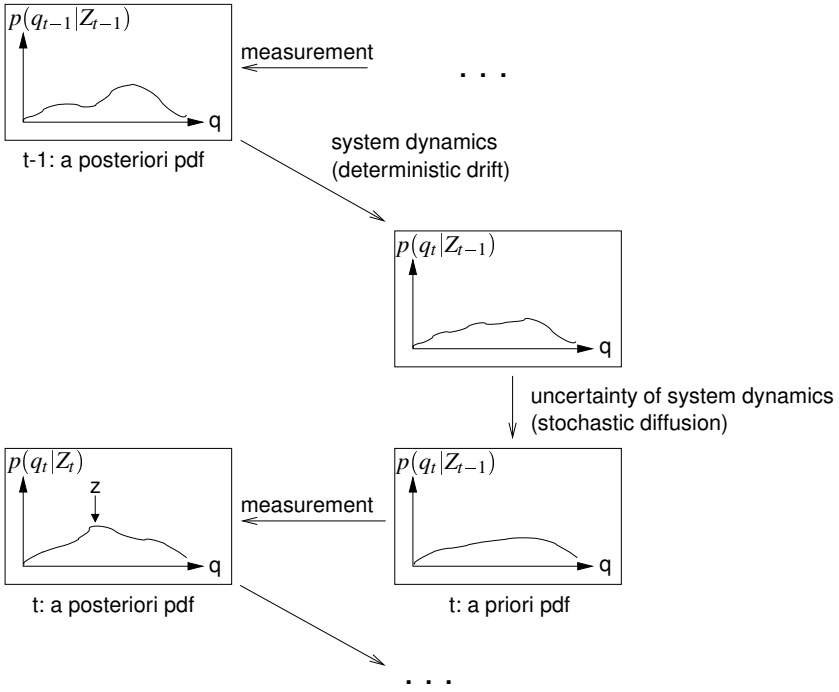


Figure 3.6: The propagation of the sample set in a particle filter.

a proportional pdf $r(y)$ is available that can be evaluated:

$$p(y) \propto r(y) \tag{3.25}$$

If we have an *importance density* $t(y)$ from which we can easily draw samples $x^{(i)}$, we can calculate a weighted approximation to $p(y)$ by:

$$p(y) \approx \sum_{i=1}^{N} \pi^{(i)} \delta(y - x^{(i)}) \quad with \quad \pi^{(i)} \propto \frac{r(x^{(i)})}{t(x^{(i)})} \tag{3.26}$$

In this approximation, the weights $\pi^{(i)}$ compensate for the difference between the importance density $t(y)$ and the probability density $p(y)$.

Using importance sampling, the weights in Eq. 3.24 can be calculated. This requires an importance density $t(\Omega_t|Z_t)$ for drawing samples $S_t^{(i)}$ giving:

$$\pi_t^{(i)} \propto \frac{p(S_t^{(i)}|Z_t)}{t(S_t^{(i)}|Z_t)} \tag{3.27}$$

If only a filtered estimate $p(q_t|Z_t)$ is needed at sequential time steps and the human activity can be modeled by a first-order Markov process as stated in Section 3.1, it can be shown (see [2]) that the weights can be calculated sequentially from:

$$\pi_t^{(i)} \propto \pi_{t-1}^{(i)} \frac{p(\mathbf{z}_t|\mathbf{s}_t^{(i)}) p(\mathbf{s}_t^{(i)}|\mathbf{s}_{t-1}^{(i)})}{t(\mathbf{s}_t^{(i)}|\mathbf{s}_{t-1}^{(i)}, \mathbf{z}_t)} \tag{3.28}$$

Using this sequential weight updating, the posterior filtered density $p(q_t|Z_t)$ can now be approximated for every time step t using:

$$p(q_t|Z_t) \approx \sum_{i=1}^{N} \pi_t^{(i)} \delta(q_t - \mathbf{s}_t^{(i)}) \tag{3.29}$$

The method presented above to sequentially propagate a sample distribution over time is known as *Sequential Importance Sampling* (SIS). After a few iterations of the weight calculation using this type of importance sampling, a few samples will have high weights while most of the samples will have negligible weights. This *degeneracy phenomenon* cannot be avoided in the standard SIS approach and results in a waste of computational power for updating a large amount of particles having small weights. To circumvent the degeneracy phenomenon, either the importance density must be chosen very carefully or a resampling of the sample distribution must be introduced. Today there exists a wide range of particle filtering approaches that follow the basic SIS algorithm outlined above but differ in the choice of importance density and/or resampling step [2].

Within the vision community, a specific particle filtering technique called *Sampling Importance Resampling* (SIR) is better known as *Conditional Density Propagation* (CONDENSATION) introduced by Isard and Blake [47] to track objects in noisy image sequences. In the SIR algorithm the importance density is chosen to be the prior density:

$$t(\mathbf{s}_t^{(i)}|\mathbf{s}_{t-1}^{(i)}, \mathbf{z}_t) = p(\mathbf{s}_t^{(i)}|\mathbf{s}_{t-1}^{(i)}) \tag{3.30}$$

This leads to a simplification of the weight update equation 3.28:

$$\pi_t^{(i)} \propto \pi_{t-1}^{(i)} p(\mathbf{z}_t|\mathbf{s}_t^{(i)}) \tag{3.31}$$

Resampling is performed in SIR at every time step, so that $\pi_{t-1}^{(i)} = 1/N$ and the weight update becomes:

$$\pi_t^{(i)} \propto p(\mathbf{z}_t|\mathbf{s}_t^{(i)}) \tag{3.32}$$

Notice that the importance density is independent of the observation $\mathbf{z}_t$ and, consequently, the state space is explored without any knowledge of the observation. Additionally, resampling in every time step could result in a loss of diversity among the samples, as the samples with high weights will be selected very often during resampling. If the process noise is small, the samples selected several times during resampling will not differ very much after the sample propagation. Therefore, the sample set will eventually contain multiple instances of the same sample. However, in computer vision applications the observations obtained from image data are usually very noisy so that sample set degeneracy is very unlikely.

Following the original publication, this tracking framework has been extended to automatically switch between several movement models to provide a mechanism for classification of the movements [49]. For this purpose, a multinomial label μ is added to each

sample $\mathbf{s}_t^{(i)}$ indicating the movement model the sample belongs to:

$$\mathbf{s}^{(i)} = (\mathbf{x}, \mu) \quad with \quad \mathbf{x} = (x_1, \ldots, x_m), \quad \mu \in \{1 \ldots l\} \tag{3.33}$$

A model specific sample propagation from time t to time $t+1$ is used to propagate the samples depending on the model they represent. The recognition of a specific model is realized by calculating at every time step t and for every model μ the *model probability* $P_t(\mu = j)$ based on all samples belonging to this model:

$$P_t(j) = \sum_{i \in \Upsilon_j} \pi_t^{(i)} \quad with\ \Upsilon_j = \{k \mid \mathbf{s}_t^{(k)} = (\mathbf{x}, \mu = j)\} \tag{3.34}$$

Now at every time step the highest model probability indicates which model is currently dominant in the state space and therefore best represents the observed data.

Following the first publication of particle filters in the vision community under the name CONDENSATION [47], many vision applications have used the particle filtering framework for analyzing image sequences. Applications range from the tracking of multiple objects based on laser range data acquired by a mobile robot [56] over the 3D-tracking of a walking human based on 2D image data [88] to the recognition of hand gestures drawing on a whiteboard with a specially colored marker [12].

3.3 Summary

In this chapter, we presented a model for human activities that assumes that the underlying process leading to the observations of an activity can be modeled by a first-order Markov process. Under this assumption the observations can be used in a recursive Bayesian filter to recognize the performed activity. Three algorithmic techniques that are realizations of a recursive Bayesian filter were presented, namely Kalman filtering, Hidden-Markov-Models, and Particle filtering. Each of these techniques can cope with a different complexity of the state probability distribution function that describes the current state of the process giving rise to the observations. With the introduction of these techniques, we now have the background to deal with the recognition of activities based on observations.

4 Finding Human Hands in Color Images

The recognition of activities as outlined in the previous chapter depends on the availability of a sequence of features $Z_T = \{\mathbf{z}_1, \ldots, \mathbf{z}_t, \ldots, \mathbf{z}_T\}$ that characterizes the motion of interest. For the recognition of a hand gesture the motion of the hand has to be extracted from the image sequence depicting the gesture. This amounts to finding the hand in each image and tracking the hand over the complete image sequence. In this chapter, we concentrate on the image processing necessary for extracting the hand from a single image to obtain the feature vector $\mathbf{z}_t$ describing the position of the hand in this image. Tracking the hand to obtain the feature sequence Z_T will be covered within the following chapter dealing with the actual recognition of gestures.

Today, there exists a variety of different methods for feature extraction from image data, each with specific advantages and limitations. The cues commonly used can be grouped into three categories: motion, shape, and color. Each of these cues has certain strengths and weaknesses that make it suitable for specific applications. The cue chosen for recognizing a gesture should enable the detection of the hand in the images of an image sequence but not get distracted by other influences.

For example, the *Motion* cue does not only detect moving hands but also the motions of manipulated objects as well as any non-static objects in the background. It is therefore only useful in domains with static background. The *Shape* is only a good cue if the hand that executes the motion exhibits a stable visual structure during the motion. However, human hands usually exhibit a wide range of shapes during interacting with the environment. Only if the hand shape variations are limited, e.g., if most of the fingers are visible during the motion, an *adaptive* hand-shape tracker [47] could be used.

The *Color* cue is suitable for feature extraction in arbitrary domains with respect to hand shape variations and motions in the background, as it is rotation and scale invariant as well as motion independent. However, the variations in lighting conditions pose major challenges to a color-based feature extraction. Even in a room without windows, different lighting conditions are encountered at different positions in the room. This is due to the individual light sources at the ceiling, and the shading introduced by objects or moving persons. In a typical office environment with the different intensity of the light passing through the window during the day, this becomes even worse.

Resulting from the changes in the color cue due to lighting changes, it is not possible to use a static color model without restricting the environment. However, one can use a dynamic color model and adapt it to the changing color of the object of interest. Such an *adaptive color segmentation* allows us to extract features in arbitrary environments with

varying lighting conditions. For the goal of extracting features for gesture recognition in a wide range of domains, the color cue is therefore the best choice if an adaptive method is applied.

Developing adaptive color segmentation methods gains interest during the last years as color processing in general becomes more widely used and increasing processing power starts to allow adaptive solutions. Especially the growth of mobile devices and their widespread use has recently attracted interest in adaptive skin-color segmentation because lighting conditions are uncontrollable in mobile devices. For example, to enable low-bandwidth video conferencing applications with camera-equipped cellular phones, it is necessary to track the human face in arbitrary indoor and outdoor environments [90].

The purpose of this chapter is to present the adaptive skin-color segmentation algorithm developed for feature extraction. The presented technique allows us to obtain region descriptions for human hands and faces. These regions and their features are the measurement values needed by the classification techniques introduced in the previous chapter to perform gesture recognition. To start with, the properties of skin color relevant for its detection in camera images are outlined in Section 4.1. Subsequently, Section 4.2 gives an overview of previous adaptive skin color segmentation methods and their limitations. A general description of the individual image processing steps that have to be performed is the topic of Section 4.3. Our method for adaptive skin color segmentation is explained in Section 4.4. The performance is demonstrated in Section 4.5 qualitatively with snapshots from image sequences containing varying lighting conditions. The presented approach for image segmentation based on skin-color is summarized in Section 4.6.

4.1 Properties of Skin-color

Realizing an adaptive color segmentation approach that is able to deal with arbitrary and varying backgrounds requires the incorporation of additional constraints. Segmenting an image into areas with the color of interest requires to first classify each individual pixel. After performing pixel classification on the complete image, the areas of homogeneous color representing the color of interest can be constructed. Incorporating additional constraints for an adaptive segmentation approach can therefore be done in the pixel classification step as well as in the segmentation step.

In this section, we will concentrate on the pixel classification step. Obviously, the properties of the color cue and its variations under different lighting conditions can be exploited for adaptation. Several researchers have carried out basic studies on the properties of skin color and methods to model skin color to perform pixel classification. We will review in the following some important contributions to this field as these findings form the basis of our adaptive skin-color segmentation method.

Representing Skin-color

A basic paper by Yang et al. [107] analyzes the color properties of face images based on a database containing about 1000 faces of people of different races. In their work three properties important for modeling skin color are identified:

1. Skin color is clustered in a small region in a color space, i.e., the individual color values are not randomly distributed. The clustering property is independent of a specific color space, only the compactness of the cluster varies for different color spaces (see also [50, 109]).

2. The variance of the skin color distribution can be reduced by intensity normalization. Choosing a color space with intensity normalization is therefore advantageous for modeling skin color. Yang et al. propose the *normalized color space* that is obtained by removing the luminance from the color representation through normalization of the individual RGB values:

$$r = \frac{R}{R+G+B} \quad g = \frac{G}{R+G+B} \quad b = \frac{B}{R+G+B} \tag{4.1}$$

   As the value for b can be calculated based on the values of r and g with $b = 1 - r - g$, it does not contain additional information and the normalized color space is therefore in the following referred to as *r-g color space*. In the literature this color space is also referred to as *chromatic color space*.

3. The skin color distribution for a specific lighting condition can be characterized by a multivariate normal distribution in the normalized color space.

Another study focusing on skin color contained in isolated pictures collected over the internet is the work by Jones and Rehg [51]. For their study, a total of 4675 images from the internet containing skin were segmented manually. They compared two methods for modeling normalized skin color, namely mixture models and color histograms. A performance evaluation revealed that histogram models are superior to Gaussian mixture models if a large training set is available. For modeling the skin with a histogram model, 32 bins/channel were found to give the best result for their training set. They point out that the choice of the bin number is a crucial point in histogram models. A small number of bins leads to over-generalization while a large number of bins results in a model too specific for a slightly changed situation. Their work allows us to draw the conclusion that Gaussian mixture models are a feasible approach to recognize human skin in environments with limited training data but are outperformed by histogram models otherwise.

Despite of this conclusion it must be noted that even modeling skin distributions with a histogram does not result in a perfect pixel classification. To analyze the classification quality of histogram modeling, Brand and Mason [16] used the data set collected by

Jones and Rehg [51] to construct a histogram. Subsequently, they set the classification threshold so that 95 % of the skin pixels were correctly classified. This threshold resulted in an incorrect classification of around 20 % of non-skin pixels that were classified as skin [16].

This high rate of false positive classification may be due to overlapping skin and non-skin distributions in the training data. This assumption is supported by recent work of Gomez and Morales [38] who constructed a rule-based classification system for their training set. The resulting decision rules correctly classified 96 % of skin with a false positive classification of only around 6%. Although these numbers are a major improvement compared to the results of Brand and Mason [16], they show that a completely correct classification of individual pixels may be impossible due to overlapping skin and non-skin distributions.

Summarizing, most publications proposing to model skin color distributions with histogram models indicate a substantial amount of false positives. Only a recent publication by Gomez et. al [38] working with another data set presents a reasonable low rate of false classified pixels. However, the achieved classification quality does not allow to draw inferences how the error rates at the pixel level influence the overall segmentation result for an arbitrary image. What if under a certain lighting condition the complete face is within the false negatives? Vice versa, what if the color of a wooden book shelf next to a face is contained in the false positives?

Another, probably more important, aspect is the superior image quality of pictures taken manually. The images of an image sequence taken automatically during observing a human may exhibit, for example, heavy shading due to insufficient lighting. Therefore the skin pixels contained in such images will cover a wider range of the color space that has a larger overlap with non-skin pixels making a discrimination more difficult. A solution to this problem is the use of a local skin color model that is adapted to the current lighting situation.

Representing the Global Skin Distribution: the Skin Locus

Störring et al. concentrate in their work on the properties of skin color in faces of different ethnical subjects under changing lighting conditions [92, 93]. Their study verifies a physics-based model of skin color that predicts the appearance of skin color under varying lighting conditions. For this purpose images of different people under different illumination conditions are captured. The test set consists of seven subjects from around the world (Latvia, Denmark, Greece, Spain, China, Iran, India and Cameroun) to capture all possible variations of skin type. Controlling the illumination conditions and knowing the spectral sensitivity of the camera allows to prove the validity of a theoretical framework modeling how skin color distributions are affected by changing illumination conditions. The relation between changes in the lighting condition and the resulting changes in the mean skin color chromaticity can be seen in Fig. 4.1. The means of the skin color values for all training subjects and the area predicted by the theoretical

model are depicted in Fig. 4.2(a) for four illumination conditions. The distribution of the individual skin color values under the four illumination conditions for the Caucasian skin type (from Latvia) can be seen in Fig. 4.2(b).

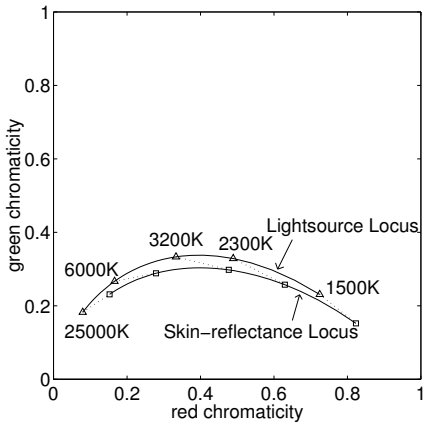
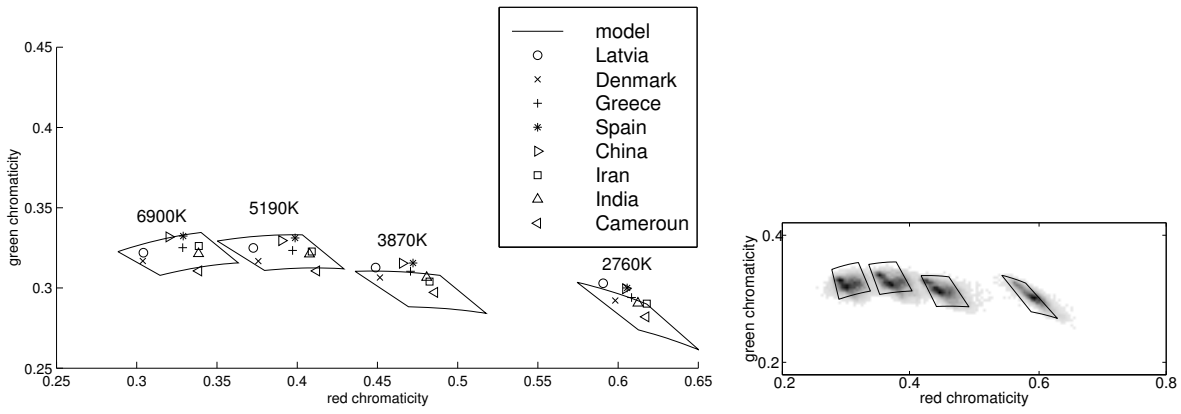


Figure 4.1: The location of light sources and corresponding mean values of skin color areas calculated with a theoretical model for different color temperatures if the camera is white balanced to a Blackbody radiator of T=3200K (from [92]).



(a) (b)

Figure 4.2: (a) The means of the skin color values for all seven training subjects together with the skin color range predicted by the theoretical model. (b) The distribution of the individual skin color values for the Caucasian skin type from Latvia (both images from [92]).

Following the theoretical model, it is shown that the area occupied by the skin color distribution of all different skin types under all possible lighting conditions occupies a shell-shaped area in the r-g normalized color space. This area can be modeled by two quadratic functions [90] and is referred to as the *skin locus* in the following.

For real applications the area of the actual skin locus can be measured from labeled training images for all relevant illumination conditions. This avoids knowing the spectral sensitivity of the camera sensor. Using a measured skin locus allows us to realize a preprocessing step in skin color segmentation approaches by discarding all pixel values that are not contained in the measured skin locus [90]. Through this preprocessing step an adaptive algorithm can be prevented from adapting to a color that is not skin-like. The size of the measured skin locus depends on the skin color of the training subjects and the illumination conditions of the training images.

4.2 Related Work

A large number of applications have been developed that segment images based on the color cue to track human motions. Probably the most famous adaptive image segmentation system is the Pfinder ("person finder") system developed at MIT by Wren et al. [103]. Pfinder can track a single human in real-time based on a multi-class statistical model of the color and shape of body parts. For this purpose, a simple representation of the human body with head, torso, arms, hands, legs and feet is used. The color of each body part is modeled as Gaussian in the YUV color space [105] and its position as Gaussian in image coordinates (x,y). To enable a foreground-background segmentation, the statistics of each background pixel are learned during an initialization phase without a human in the field of view. Based on the pixel statistics, each background pixel is modeled by a Gaussian distribution that is slowly adapted to lighting changes.

After initialization, a large change in the background statistics of an image area is interpreted as a human entering the scene. In this case, all pixels not belonging to the background are used to initialize the color models of the individual body parts following a simple model of the body configuration. During tracking each new image is processed by calculating for each pixel whether the closest body model in image coordinates or the corresponding background pixel model has the highest likelihood. All pixels assigned to a body model are subsequently used to smoothly update its Gaussian parameters allowing Pfinder to cope with smooth changes in lighting. Resulting from the modeling of the background and the body parts, only a single, completely visible human wearing homogeneously colored clothes is allowed to enter a scene with a nearly static background. Despite these limitations, Pfinder has been applied very successfully in many applications, for example in American sign language recognition [91].

The related LAFTER system by Oliver et al. [73] is intended to track and recognize the face of a single user sitting in front of a computer. Here the overall background is modeled by a mixture of Gaussians instead of modeling each pixel individually. This accomodates varying backgrounds resulting from the use of a pan-tilt camera for active face tracking. For modeling the face, the intensity-normalized r-g color space (see Section 4.1) is used as it is well suited for representing skin color over a wide range of lighting conditions [107]. The face color is represented by a mixture of Gaussians that

is learned offline. Classification is performed by assigning each pixel to the class with the highest probability. Performing a connected components analysis ([39], Ch. 2) on this labeled image yields the skin-colored regions in the image. Adaptation of the face color model consists of updating each class with the assigned pixels through an iterative version of the EM-algorithm. As the distance between user and camera does not change much, the region with shape and size similar to the expected face is selected to represent the face.

A similar approach by Raja et al. [82, 80] also employs a mixture of Gaussians for modeling skin color in r-g color space to track the face of a single person in real-time. However, no background model is used as their approach aims at finding the face position in cluttered, non-constant backgrounds. After manual initialization of the color model, the skin probability image is computed by calculating for each image pixel its probability. Assuming only a single skin-colored object in the image, the face position is obtained by calculating the *center of mass* (COM) for the overall skin probability image. Here the mixture model is updated based on all pixels within a bounding box around the COM. In the example in [82], the active camera performs changes of pan, tilt, and zoom while the tracked face of a human moving through an office undergoes large changes in illumination.

Recently, Soriano et al. [90] proposed an adaptive algorithm that tracks single faces on camera-equipped mobile phones. To reduce the computational load, their algorithm employs a histogram representation for modeling the current skin color distribution in r-g color space. During initialization, manually segmented skin regions are used to construct the skin histogram $S(r, g)$. A ratio histogram $R(r, g)$ is obtained by dividing every bin in $S(r, g)$ by the corresponding bin in the histogram of the complete image $I(r, g)$. This results in small ratio values for colors that occur more often in $I(r, g)$ than in $S(r, g)$, i.e., that are also present in the background. For color values that do only occur in the skin histogram $S(r, g)$, the ratio value is close to one. Consequently, the ratio histogram contains for every pixel color the probability that is skin color and allows us to directly compute the skin probability image for a new input image. Similar to Raja et al. [82], the face is located by computing the COM of the skin probability image. The pixels in a bounding box around the COM are used for updating the skin histogram and calculating the new ratio histogram. To avoid tracking background objects, all non-skin pixels contained in the bounding box are discarded using an empirically determined global skin locus (see Section 4.1). Modeling the skin color distribution with a ratio histogram requires a suitable choice of the histogram bin size. A large number of bins leads to a sparse ratio histogram that is not able to cope with small lighting changes. Vice versa, a small number of bins may result in segmenting background objects with a color close to skin. The published results indicate that the approach is able to track a single face positioned close to the camera, i.e., occupying a large area of the camera image, using a ratio histogram with 32 bins/channel.

4.3 Extracting Features with Image Processing

The extraction of relevant features from image data plays an important role in research work on image understanding. A variety of different methods has been developed to detect, e.g., lines, edges, or regions. For recognizing human hand motion in arbitrary environments, we have pointed out above that the color cue is well-suited. The color cue is shape and scaling invariant and, additionally, it is a simple feature as it is directly available from the image sensor.

In image processing systems the feature extraction usually consists of several distinct processing steps carried out sequentially:

1. Image Acquisition (quantization, digitalization)
2. Image Enhancement (filtering, color correction, ...)
3. Iconic Feature Extraction (color label, gradient, ...)
4. Segmentation (edges, regions, ...)
5. Image Feature Extraction (e.g., for regions: size, compactness, area, ...)

We will describe in the next paragraphs these general image processing steps more precisely, concentrating on those aspects that are relevant for the task of finding skin-colored objects. In this way, we set the stage for the description of our method for skin color segmentation that will be introduced in Section 4.4 by referring to the different processing steps explained in more detail in the following.

Image Acquisition

The image acquisition is done primarily with standard video cameras. In the camera the spectral light distribution reflected from the scene is measured for each image position with a spatial matrix of charge coupled devices (CCD). To obtain a color representation of the scene, the incoming light spectrum is separated with filters in the three primary colors red, green, and blue following the RGB color model [105]. Notice that the camera sensor does not measure exactly one wavelength but integrates the spectral intensity that passes the optic filter around the primary wavelength. Consequently, as this optic filter is not identical for different cameras from different manufacturers, the RGB values of a light spectrum observed by different cameras do not match perfectly.

Besides these differences between cameras, it is necessary to calibrate the gains of the three different RGB values of a single camera to a common base value. This calibration is called *white-balancing* and is done by showing a white object to the camera and calibrating all three color values to their maximum value. As the camera is white-balanced to a specific illumination, a change in lighting will result in a wrong calibration and

possibly in an inadequate dynamic range of the individual RGB color values. Therefore, state-of-the-art video cameras are capable of performing automatic white-balancing.

For compatibility between different setups, the image acquisition step outlined above is the most important. Depending on the sensitivity of subsequent processing steps with respect to variations in the RGB values, it may therefore be necessary for an image processing system to be adapted to the specific spectral characteristics of the camera used in a given setup.

Image Enhancement

The image enhancement step is usually carried out to improve the quality of the image data, i.e., it modifies the color values of the individual pixels. Besides the elimination of measurement artifacts introduced by the image sensors like, e.g., pixel noise, this step is used in color image processing to reduce the effects of shades or varying reflectances. As the lighting conditions in real world environments are far from being constant, especially if the camera or the observed object moves, the color values measured by the camera exhibit large variations. Figure 4.3 illustrates schematically several aspects that influence the light spectra observed from a specific object: the spectral characteristics of the illumination sources; the distances and angles between sources, object and camera; the object's reflection and absorption characteristics; the light reflections from other objects in the scene.

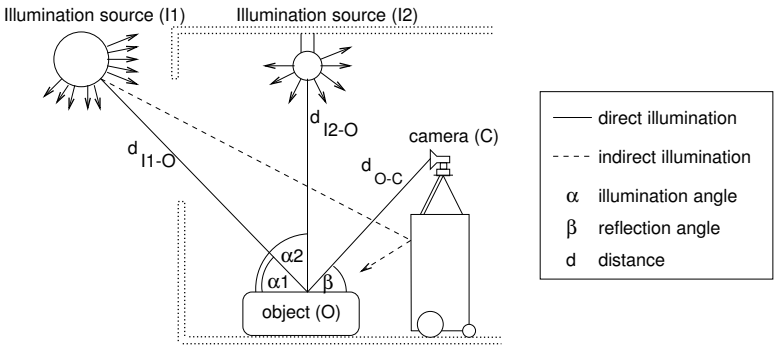


Figure 4.3: A simplified illustration depicting different influences on the appearance of an object.

Obviously, a change in the position of the object or the observer will very likely result in a changed spectrum of the object as it is observed by the camera. However, a human observer often does not notice this changed appearance: Under different lighting conditions an object surface is observed as having a constant (i.e., the same) color. Several different processes in human image processing are believed to be responsible for this capability. With respect to color perception, the low-level stages of the human visual system perform white-balancing and adapt the sensitivity of the receptors in the

human retina (see [100], for example). Besides these physiological processes in the low-level visual system of the human, color perception is believed to also consist of higher level processes in the brain. One aspect is that the human observer extracts the color of the illuminant out of the stimulus and automatically discounts it [24]. Another aspect is the tendency of humans to see the color of familiar objects as we expect them to be. Together with the poor color memory of humans, an introspective view therefore leads to the assumption that humans are able to perform chromatic adaptation. However, while the low-level processes are indeed resulting in a certain ability to perform chromatic adaptation, there seems to be a substantial influence from the high-level processes on how humans perceive color that depends heavily on the cognitive context.

To enable image processing systems to cope with variations in lighting conditions, researchers have proposed a large variety of *color constancy* algorithms that aim at mapping the observed color description to an illumination-independent description (cf., e.g., [28, 34, 84]). However, due to the different processes active in the human visual system, it is an open question whether perception-based color constancy algorithms are sufficient to cope with lighting variations [35].

Besides the qualitative aspects of color constancy algorithms, the computational effort to perform the color correction of images is still too high to apply color constancy algorithms in real-time systems. Additionally, many color constancy algorithms need certain assumptions to be satisfied or need additional input data, for example the knowledge of the illumination conditions. Therefore, real-time image segmentation approaches try to convert the raw color data into a color space that is less sensitive to changes in lighting. Depending on the application domain and the processing steps performed on the color data, a large number of different color space representations has been proposed. A perceptually uniform color representation is the Lu^*v^* *color space* [105] where the Euclidean distances between different color tuples are proportional to the differences observed visually by a human. One transformation that has proven to be well-suited for reducing skin color differences due to lighting changes is the r-g color space (see 4.1). However, a static color space transformation into a new color space is not sufficient to cope with arbitrary changes in illumination. Therefore, additional measures have to be taken in subsequent processing steps to cope with lighting variations.

Iconic Feature Extraction

Based on the enhanced image representation, the extraction of iconic features of the individual pixels can be carried out. This is an optional processing step, as the subsequent segmentation step may be based directly on the enhanced image representation. For example, the Lu^*v^*-representation of an image can be used directly in the segmentation step to find regions of homogeneous color.

However, the example for direct color segmentation based on the Lu^*v^*-representation does not apply to situations where specifically colored objects need to be found. In this case, a model for the color of interest can be used to assign to each pixel a label indicating

whether it belongs to this color model based on some distance criteria for the color value. Such an iconic feature extraction transforms a color image into a *label image*. Instead of directly assigning discrete labels to color pixels, it can be more advantageous to use continuous values for every pixel indicating the likelihood that it belongs to a specific color model. This results in a *probability image* for the specific color.

Note that a static color model for the color of interest is not sufficient to cope with lighting variations. It is therefore necessary to either perform some adaptation of the color model or to use several different color models depending on the current lighting situation.

Segmentation

In the image segmentation step, the iconic features of the input image are spatially grouped together to form some kind of high-level representation of the image contents. Obviously, this processing heavily depends on the type of iconic features provided by the previous processing step. If we concentrate on the color cue, the iconic features of the individual pixels can be their color values in a certain color space, some labels indicating specific color models, or continuous values indicating the likelihood of a color model.

For example, the Lu^*v^* color values present in an image can be grouped into classes based on their Euclidean distances in the color space (e.g., [20]). Now each pixel gets a label assigned indicating the color class it belongs to. If every pixel has been assigned to a color class resulting in a label image, a connected components analysis ([39], Ch. 2) yields regions of homogeneous color.

Instead of such a data-driven image segmentation, a model-driven image segmentation using specific color models representing the colors of interest can be carried out. In this case, each iconic feature represents how the pixel color relates to the specific color model resulting in a probability image. This probability image is converted in the segmentation step into a label image by thresholding the probability values. However, such a thresholding applies only to situations where there is only a single color of interest. Given several probability images for different color models, i.e., a probability vector for each pixel, more sophisticated classification approaches are needed. A typical classificator used for color segmentation is *Maximum-Likelihood classification* that selects the model giving the highest probability value. To remove spurious outliers from the label image and generate a smoothed segmentation result, median filtering is typically applied to the label image. Applying a connected components analysis to the smoothed label image provides the image regions with the colors of interest.

Image Feature Extraction

After the segmentation step, the homogeneously colored image regions in an image are available. Which image features are extracted from the segmented color regions depends on the given application. Typically, the region size, its center of mass, the surrounding

polygon, and some additional data like, e.g., the eccentricity are calculated for every image region. These features provide a description for the homogeneously colored areas contained in the image. For the analysis of isolated images to perform, for example, object recognition, these image features are sufficient.

For our goal of recognizing gestures from image sequences, the features extracted from an individual image can be linked to the features extracted in the previous images of an image sequence. The exploitation of the temporal information is often accomplished through applying Kalman filtering (see Section 3.2.1) to the positions of the regions segmented in every image of an image sequence. In this way, trajectories that represent the motion of homogeneously colored objects in image sequences can be extracted from the individual image features. The extraction of these *image sequence features* will be considered in more depth in the next chapter dealing with the actual recognition of gestures. In this chapter, we will concentrate on the extraction of image features describing human hands from isolated images. For this purpose, we will now introduce the developed method for adaptive skin color segmentation that allows us to extract image features representing human hands.

4.4 Adaptive Skin Color Segmentation

As pointed out on page 56, there are currently no color constancy algorithms available that allow to *tolerate* arbitrary lighting changes [35]. Current approaches to color image processing try to constantly *adapt* internal models of the interesting colors to the observed changes in the image data. An adaptive algorithm needs to decide for every image whether the image contains objects of interest that have changed somehow and to whose changed appearance the algorithm should adapt. In other words, the question that needs to be answered is: 'What parts of the new image belong to the previously observed objects that are now illuminated with a different lighting and therefore have a different appearance?'.

A major challenge in the actual implementation of adaptive strategies is the lack of 'ground truth' in the color signal: it is impossible to decide whether a pixel still shows the same object that now exhibits a different appearance based only on iconic information, i.e., the color of an individual pixel. Consequently, an adaptive algorithm may adapt the color models to 'wrong' values. For example, if a tracked human hand moves over a wooden desk with a color slightly different from skin color, the desk could be accidentally segmented to belong to the hand region. Subsequently, the color model may be updated not only with the pixels belonging to the hand but also with the pixels exhibiting wood color.

In order to avoid such distractions, we propose to use domain-dependent context knowledge to restrict the adaptation to 'valid' color values. For example, in the assembly construction domain introduced in Section 2.2, the hands manipulate baufix® parts while at the same time wooden parts may be present on the table. In this domain, the hand

regions are moving while wooden parts are static. Therefore, adaptation can be restricted to those segmented regions that exhibit motion. In a scenario where a mobile robot is intended to track human faces based on skin color, a face detection algorithm can be used to verify that a segmented skin-colored region actually represents a face [31]. Only if a face is detected at the position of the skin-colored region, an image patch of face size is used for adapting the skin color model.

Previous adaptive approaches do not apply any verification step and use a single rectangular update area based on the COM of the segmentation result [82, 90]. Consequently, in case of a wrong image segmentation the color model is adapted to this wrong image patch. In contrast, our approach only adapts its color model to skin colored image areas that exhibit motion or have been identified as face regions by a face detection algorithm. Consequently, a wrong adaptation is avoided and the approach is suitable for segmenting hands and faces of humans acting in a wide range of environments.

The principal processing steps of our approach for segmenting input images using an adaptive skin color model are as follows (see Fig. 4.4):

Based on a domain-dependent initialization step, an initial skin color model is generated. From now on, every image is processed with the current skin color model to label every pixel as either skin or non-skin pixel. For this purpose, the input image is first transformed into the r-g color space. Based on the skin color model, the probability of every pixel for being skin color is calculated. A *classification threshold* is applied to the probability values to obtain a binary label image. An example label image is visible in Fig. 4.4 on the left beneath the input image. This label image is smoothed through applying a median filter to eliminate spurious pixels with skin color. Skin-colored regions are extracted from the smoothed label image by carrying out a connected components analysis. These skin-colored regions represent image-specific information about the objects in the scene having skin color. Tracking the segmented skin-colored regions over time provides the trajectories of the objects in the image sequence for further analysis, e.g., for recognizing the performed gestures.

To select the skin-colored pixels that can be used for updating the skin color model, two different types of context knowledge for determining the 'true' skin areas are applied:

- If a tracked region exhibits motions, i.e., it is not a skin-colored background object, it is considered an 'interesting' region and an update region R_{update} is constructed based on the segmentation result (see Section 4.4.6).

- If a region segmented in an individual image exhibits a face-like structure, an elliptical update region R_{update} is constructed based on the size of the face as determined by the face detection. If this face region is also an 'interesting' region due to its motions, the elliptical update region replaces the original update region.

Based on a pre-trained global skin locus similar to the one used by Soriano et al. [90], all pixels in the update regions that are not skin-like are discarded. For the remaining skin-like pixels lying inside the skin locus, the skin probability is calculated using the

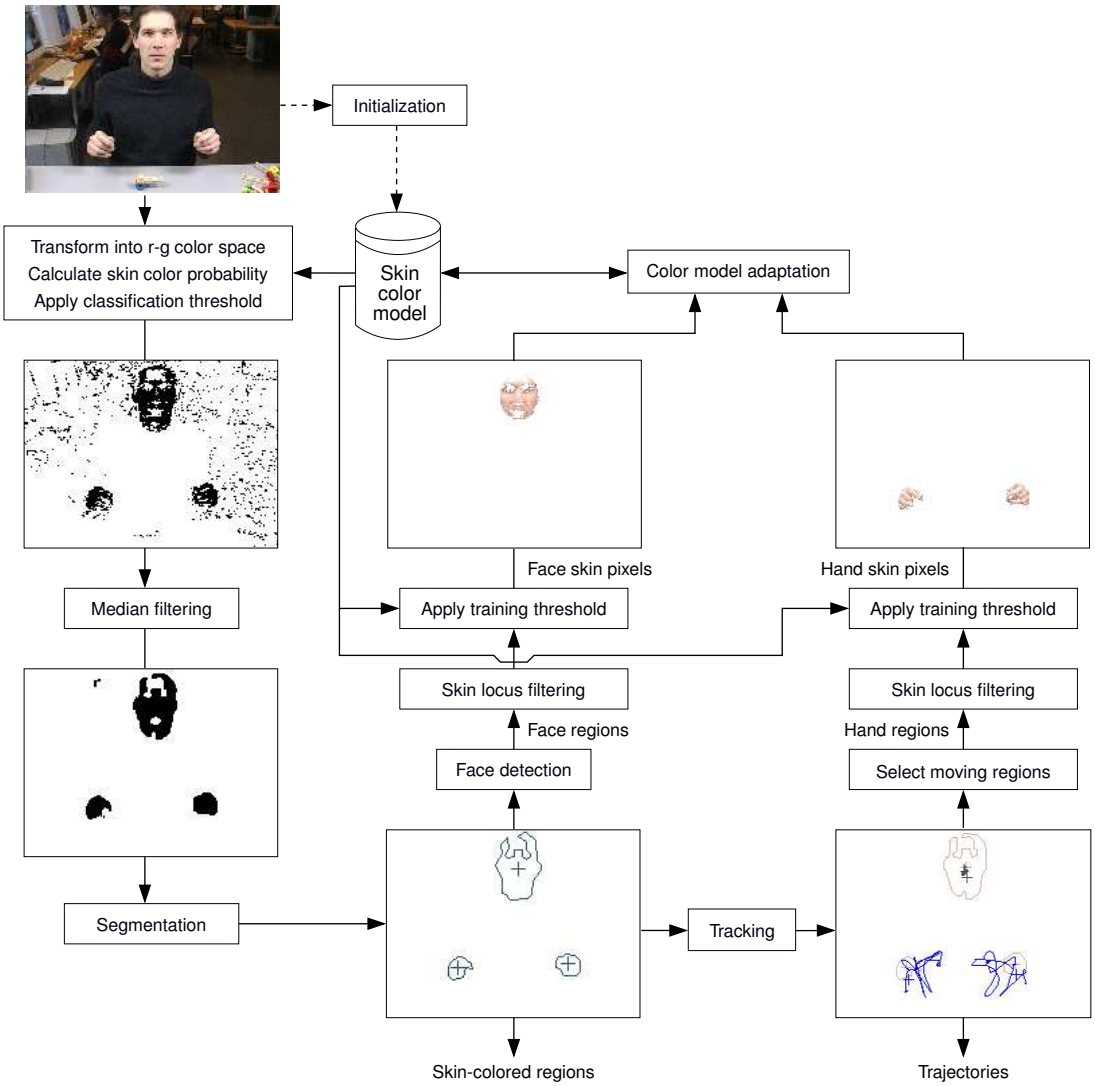


Figure 4.4: The image processing for adaptive skin color segmentation.

current skin color model. This probability is compared to a threshold that was acquired in the previous skin color model training phase. All pixels with a probability value exceeding this *training threshold* are used for adapting the skin color model. The updated skin color model is used for segmenting the next input image.

The realized adaptive skin-color segmentation

1. imposes a minimal set of restrictions on the scenario
2. enables skin color segmentation in real-time on standard workstations
3. provides a polygonal region description for all skin-colored regions instead of only the center of mass of a single skin-colored region

A detailed description of the individual processing steps for performing adaptive skin color segmentation is the content of the next paragraphs.

4.4.1 Modeling the Skin Color Distribution

Based on the findings by Yang et al. [107] described in Section 4.1, we chose to use the normalized r-g color space for representing skin color. The empirical study by Yang et al. suggests that a Gaussian mixture is well-suited to model skin color distributions. We pointed out that more recent publications dealing with the representation of skin color distributions given large amounts of training data argue for the use of histogram models [51, 16] or rule-based classifiers [38]. However, histogram models exhibit a relatively high rate of 20 % false positives, i.e., background pixels classified as skin [16], and the low rate of 6 % false positives published by Gomez et al. has not been verified on other test sets. Even a low rate of false positives can prohibit a successful segmentation in scenarios with uncontrollable conditions with respect to background and human skin types.

Another important aspect is the fact that the test sets used consist mainly of photographs that exhibit 'good' lighting conditions, e.g., sufficient illumination. In contrast, images from an image sequence captured with a video camera performing automatic exposure control are likely to be of inferior quality due to, for example, insufficient illumination. Therefore, the skin and non-skin color distributions of image sequences will differ from the test sets used in the published studies.

Given the differences between single photographs and image sequences with respect to background and lighting conditions, we argue for the use of adaptive, *local* models of the skin color distribution instead of a global skin color model trained offline. Given the small amount of training data available in an adaptive approach and the problem of selecting the appropriate number of histogram bins to model the skin distribution [51], the use of an adaptive histogram model is difficult. Instead, Gaussian models are used very often in adaptive color segmentation approaches [103, 73, 82] as they allow to model color distributions with a small parameter set.

Consequently, we use Gaussians to model skin color distributions. The skin likelihood for a pixel with color value $\mathbf{x} = (r, g)$ can be calculated for a Gaussian $G(i)$ with mean $\boldsymbol{\mu}_i$ and covariance $\boldsymbol{\Sigma}_i$ using:

$$p_i(\mathbf{x}) = \frac{1}{\sqrt{2\pi \det \boldsymbol{\Sigma}_i}} \exp\left\{-\frac{1}{2}[\mathbf{x} - \boldsymbol{\mu}_i]^T \boldsymbol{\Sigma}_i^{-1} [\mathbf{x} - \boldsymbol{\mu}_i]\right\} \tag{4.2}$$

The remaining design decision is whether an *unimodal Gaussian* or a *mixture of Gaussians* should be used to model a skin color distribution. Obviously, this depends on the properties of the distribution, i.e., on the variations in lighting and skin types contained in the distribution.

Selecting the Appropriate Gaussian Model

The simplest modeling of a skin color distribution consists of an unimodal Gaussian. For the special case of modeling the skin color distribution of a person's face, a single Gaussian has been shown to give good results [80]. Using an unimodal Gaussian has implications for the goal of real-time performance, as calculating the Gaussian parameters of mixtures increases the computational load for performing adaptation. More important, however, is the increase in the computational load in the classification step which must be carried out for every image. The skin color probability of a pixel needs to be calculated for every mixture component of a Gaussian mixture instead of once for the single Gaussian. Therefore, the relation between the computational load for classification and the number of mixtures is nearly linear, i.e., a Gaussian mixture with two components needs twice the computational power for classifying an image pixel.

While modeling a skin color distribution with a single Gaussian is computationally less expensive, it results in a reduced ability to deal with color inhomogeneities. Color inhomogeneities are encountered within a single skin-colored object (intra-region) due to, e.g., partial shading as well as between objects (inter-region) due to, for example, different skin types or lighting conditions. The most important situation where a skin-colored hand or face exhibits large intra-region color variations are partial shading effects. Figure 4.5 gives an example for intra-region color inhomogeneities. Here light comes through a window on the left resulting in a partly shaded right side of the face and a darker hand visible on the right compared to the hand on the left that is very bright. For such situations, a good segmentation result can only be achieved through modeling the skin color distribution with a mixture of Gaussians.

Figure 4.5: Example image depicting the inhomogeneous skin color due to shading.

Due to different lighting conditions at different locations in the scene, an image may contain both, intra-region and inter-region color inhomogeneities. Consequently, if several skin-colored regions are present in an image, the optimal solution is to model each skin-colored region individually by a mixture of Gaussians. While this is the best solution for the representation of different skin-colored regions, it is computationally too expensive for the goal of developing an adaptive segmentation approach working in real-time.

Therefore, we implemented two different methods for modeling skin color distributions that are suitable for different environments:

Unimodal Gaussian for each region: Separate unimodal Gaussians allow for different lighting on the individual hands or faces but not for color inhomogeneities within a single skin-colored region as depicted in Fig. 4.5. If after initialization a portion of a skin-colored object changes appearance due to, e.g., shades, only the part of the object with stable visual appearance that is modeled by the Gaussian skin color distribution will still be segmented correctly. While the boundary of the hand or face region is therefore inaccurate, it is still possible to track this region based on its unchanged part. However, updating of the skin model based only on the segmented image area will result in a Gaussian not capable of modeling the changed appearance. Therefore, modeling skin color with a single Gaussian for each region is only applicable if the update area for a region can be determined based on context, e.g., from face detection results. An important disadvantage of using a single Gaussian for modeling the color of skin-colored regions containing intra-region color inhomogeneities is the large variance resulting from the inhomogeneities. Classifying an image with an unimodal Gaussian having a large variance results in labeling background pixels as skin due to the broad skin color distribution.

One mixture of Gaussians for complete image: The quality of modeling several skin colored regions in an image with one mixture of Gaussians usually increases with the number of mixture components used. The mixture allows for a better modeling than the unimodal Gaussians if the number of components M is at least equal to the number of skin-colored regions in the image and the different regions have partially a similar color. For a single human, the potential differences between the hands and the face will be related primarily to different lighting conditions. Consequently, with a smaller inter-region variation due to similar skin appearance of the hands and the face of a single human, a mixture of three Gaussians allows us to also model intra-region variations due to shading. More important, however, is the more precise modeling of the skin color distribution that reduces the number of false positives in the background.

Selecting the appropriate modeling of the skin color distribution is depending on the application domain and the available processing power. For example, in a domain with a single acting human like the Situated Artificial Communicator, the mixture of Gaussians is more appropriate as it allows for intra-region variations and a more precise skin color modeling. Using standard workstations, the associated computational cost can be handled in real-time.

4.4.2 Measuring the Skin Locus

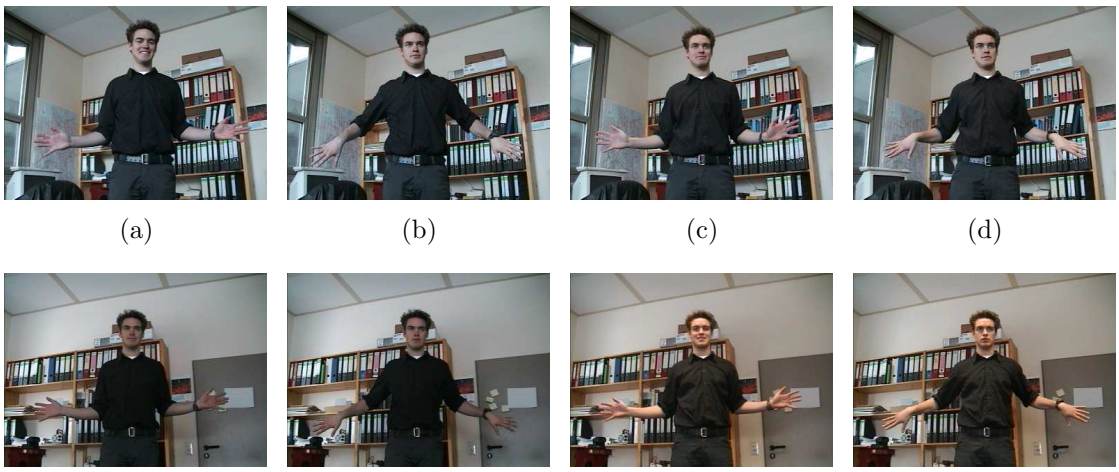
As pointed out in Section 4.1, the skin color of a large number of different human subjects is distributed in normalized color space in a restricted area, the skin locus. This skin locus can be used to ensure that only skin-like pixels are used for adapting a skin color model [90]. The exact shape and placement of the locus depends on the camera characteristics and on the lighting conditions used for acquiring the training images. To generate the skin locus for our setup, we collected images containing skin patches under different lighting conditions with a Sony EVI-D31 camera.

The first training set contained images of five different subjects. Figure 4.6 shows original images of the five subjects before manual segmentation of the skin-colored areas.

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

Figure 4.6: The first training set consisting of images showing five different persons.

For every person and every lighting condition, two pictures were taken showing the two different sides of the hands. Fig. 4.7 shows the complete set of images for one person.



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

(e) (f) (g) (h)

Figure 4.7: For each lighting condition two images were taken showing the two different sides of the hands: (a),(b) camera white-balanced for outdoor conditions; (c)-(h) camera white-balanced for indoor conditions.

A total of four different lighting conditions was used:

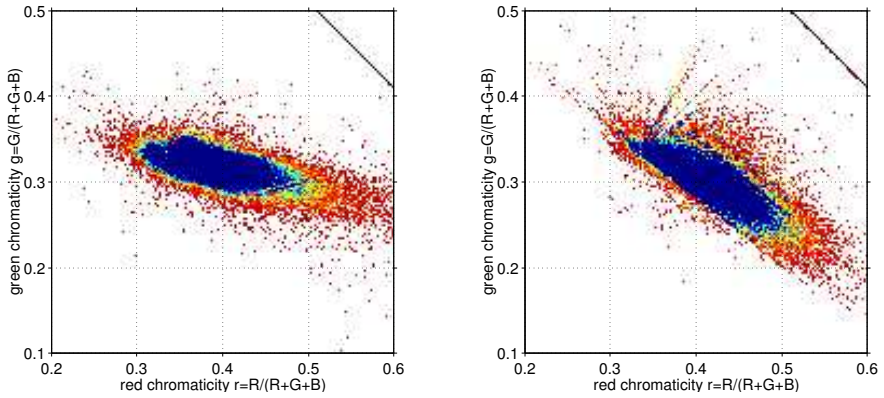
The camera white-balanced for outdoor conditions :

1. Fig. 4.7 (a),(b): light coming through the window on a cloudy day

The camera white-balanced for indoor conditions:

2. Fig. 4.7 (c),(d): direct light from outside
3. Fig. 4.7 (e),(f): light from outside with partial shading due to blinds
4. Fig. 4.7 (g),(h): pure indoor lighting

All skin patches in the images were manually segmented to generate the global skin color distribution in normalized color space that is depicted in Fig. 4.8(a).

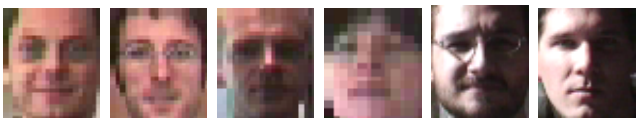


(a) (b)

Figure 4.8: The measured skin locus distributions for (a) the first training set consisting of faces and hands of five subjects and (b) faces of six persons partially containing bright sunlight.

We obtained a second set of 440 face images from six different people with the help of a face detection method (see Section 4.4.3). Here a bounding box around the detected face position was used to automatically cut the image patch showing the face out of the overall image. As can be seen in the images in Fig. 4.9, this resulted in a small amount of background pixels contained in the training set. The images were taken in an office environment with typical lighting conditions, some faces exhibited partial exposure to bright sunlight. No time-consuming manual segmentation has been carried out and the images were used directly for constructing the skin distribution depicted in Fig. 4.8(b).

Despite the different lighting conditions present in the two training sets, the overall skin color distribution constructed by merging the individual distributions is still very focussed (see Fig. 4.10). Similar to [90] we have fitted two quadratic functions



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

Figure 4.9: Image patches obtained from face detection.

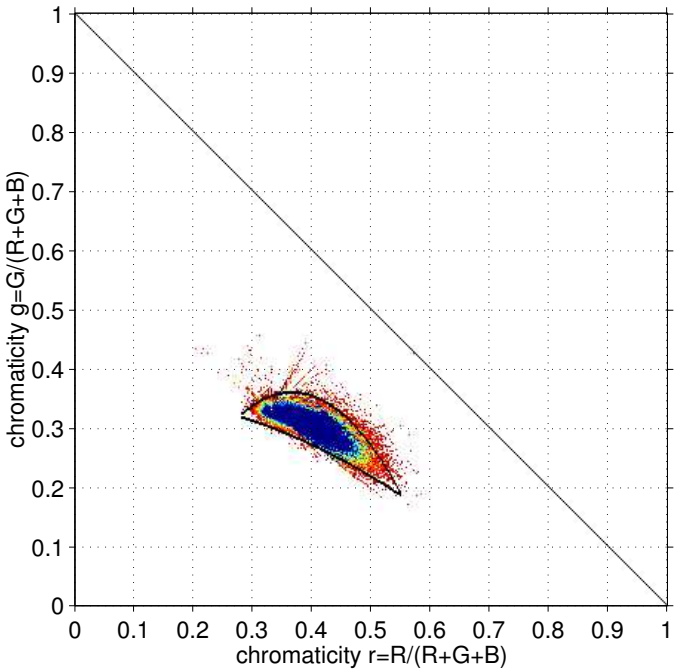


Figure 4.10: Histogram of the global skin color distribution with the skin locus fitted to the distribution.

to this distribution to obtain our setup-specific skin locus. The parameters of the two quadratic functions that enclose 95% of the pixels in the empirically determined skin color distribution are:

$$A_u = -5.05 \quad b_u = 3.71 \quad c_u = -0.32 \tag{4.3}$$

$$A_d = -0.65 \quad b_d = 0.05 \quad c_d = 0.36 \tag{4.4}$$

The decision whether a specific pixel $\mathbf{x} = (r, g)$ is contained in the skin locus is now readily available by calculating the values of the two quadratic functions based on the r value

$$F_u = A_u r^2 + b_u r + c_u \tag{4.5}$$

$$F_d = A_d r^2 + b_d r + c_d \tag{4.6}$$

and checking for the g value to lie between the two calculated function values:

$$\text{PixelInSkinlocus}(r,g) = \begin{cases} 1 & \text{, if } (g < F_u) \ \wedge \ (g > F_d) \\ 0 & \text{, else} \end{cases} \tag{4.7}$$

With this simple classification rule, a preprocessing step to discard all pixels with colors that are not contained in our training set can be realized. One remaining problem is the fact that the skin locus contains also the color values for white colors resulting from highlights in the skin-colored areas of the training images, i.e., reflections of the illumination from the light sources. Often a scene contains several artificial objects with a white color like, e.g., a door, a table, or a computer display. To avoid including the color white in the skin locus, a small region around the *white point* in r-g color space at $(0.33, 0.33)$ is excluded from the skin locus. For this purpose, a rule checking whether a pixel is white is used

$$\text{PixelIsWhite}(r,g) = \begin{cases} 1 & \text{, if } (0.327 < g < 0.339) \wedge (0.327 < r < 0.339) \\ 0 & \text{, else} \end{cases} \tag{4.8}$$

leading to the overall rule for classifying a pixel as skin-like:

$$\text{PixelIsSkin}(r,g) = \text{PixelInSkinlocus}(r,g) \wedge \neg \, \text{PixelIsWhite}(r,g) \tag{4.9}$$

This simple classification rule can now be used as a preprocessing step to select all pixels with a skin-like color. Figure 4.11 gives a visual impression of the area in r-g color space that is occupied by the skin locus. The area around the white point excluded from the skin locus is visible as black rectangle.

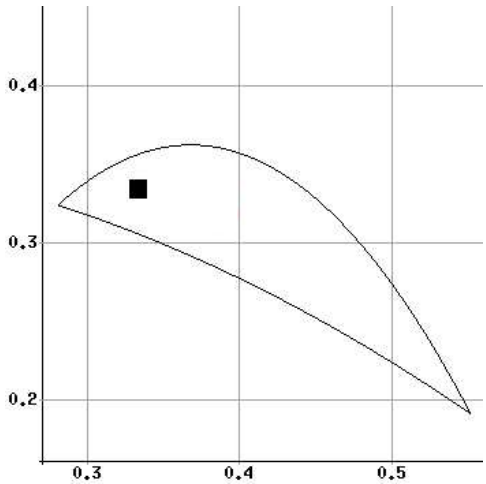


Figure 4.11: The area of the skin locus in r-g color space with the white point excluded.

4.4.3 Detecting Faces in Images

The face detection algorithm used to initialize and update the skin color model was implemented by S. Lang [31] and is based on the *eigenface* method developed by Turk et al. [96]. The underlying idea is that each graylevel image with a size of $n \times m$ pixel can be considered as a point in an nm-dimensional space. Images of faces occupy only a part of this space. By performing Principle Component Analysis (PCA) on a set of training images depicting faces, the principle components of the distribution of face images can be found. These principle components are called eigenfaces and span a subspace of the image space representing possible face images. Any image of the same size as the training images can be approximated by a linear combination of the eigenfaces. The residual error occurring during the reconstruction of an image with eigenfaces is an indicator for faces. It is small for images that are faces, large otherwise. Consequently, an empirically determined threshold can be used to classify images into face and non-face images.

The training set used to generate the eigenfaces for the face detection algorithm consisted of face images with a size of 37×43 pixels. To compensate varying lighting conditions, the images were preprocessed with a histogram equalization. With the obtained eigenfaces after applying PCA, the face detection can be performed by repeatedly extracting sub-images from the input image. A histogram equalization is carried out for each sub-image before it is reconstructed by a weighted sum of eigenfaces. Depending on the residual error, the image is classified as face or non-face. Several implementational details have to be solved for the extraction of sub-images and the search for a face in a skin-colored region, these details can be found in [31].

4.4.4 Initializing the Skin Color Model

Before skin-colored regions can be tracked in unknown lighting conditions, the Gaussian skin color model has to be initialized to model the current appearance of the human skin. Although the learned global skin locus restricts the normalized color space, its area is so large that in most standard environments not only human skin but also some other background image areas can posses 'skin-like' pixel chromaticities. Therefore, filtering a complete image with the skin locus is not sufficient to obtain skin-colored image patches usable for initializing the skin color model. Instead, a more sophisticated approach is needed to obtain the pixels in the image that are skin-colored and actually belong to a human body. In the segmentation system, three different initialization methods are available to acquire image pixels likely to represent human skin:

1. **Skin example at fixed position (needs visual feedback):**
   This initialization method is the most simple as it only requires the human to hold his face or hand at a fixed initialization area within the field of view of the camera and activate the initialization process. To position the face or hand correctly within the initialization area, a visual feedback is needed. Alternatively, the image

region containing skin can be selected manually through a graphical user interface. All image pixels within the initialization area that lie inside the global skin locus are used for initializing the skin color model.

2. **Waving hands (needs static camera/fixed background):**
   This method has been implemented by F. Lömker and is based on analyzing all pixels that posses motion information derived by subtracting two consecutive images acquired with a static camera. During an initialization phase with a human waving his hands for several seconds, all pixels with motion information are collected for generating the initial skin color model. For the calculation all pixels outside the skin locus are discarded in order to remove the background pixels that contain motion due to their 'appearing' after a hand occluding them has been removed. Consequently, if the background contains objects with skin-like color that are temporarily occluded by a hand, these pixels will also be used for generating the skin color model. As a result, a skin color model is acquired that represents the pixels contained in the skin locus and positioned in image areas exhibiting motion. However, since this method requires a static camera as well as a human waving his hands for a few seconds in front of a background without skin-like colors, its use is limited to restricted environments.

3. **Frontal view of a face (needs human face):**
   The appearance of a human face is a feature which can be utilized to realize face detection systems. The face detection algorithm (see Section 4.4.3) based on the face texture needs only gray level images to detect faces. Therefore, this approach can be used to bootstrap a skin color model. Searching for faces in a complete image takes several seconds. However, the huge computational load imposed by searching a face in a complete image without any spatial restrictions is tolerable, since this search is only carried out during the initialization phase. If a face is found, the approximate face size can be deduced from the scaling factor of the face template used for matching the face. Subsequently, all pixels within an elliptical region of similar size at the detected face position can be used to generate an initial skin color model. An additional advantage of this method is its ability to initialize different individual skin color models simultaneously if several people are in the scene.

The choice of the initialization method depends on the application domain. Initialization based on (1) a fixed image patch or (2) waving hands requires interaction between the segmentation algorithm and the user for constructing an initial skin color model. Only the (3) structure-based initialization allows us to automatically generate a skin color model in arbitrary environments if the user looks one time at the camera. As soon as the face has been recognized, an acoustic feedback, e.g., a 'beep', can be used to inform the user that the initialization was successful. Other characteristics of the

human appearance besides the face structure could also be used for initialization, e.g., a specific hand posture or the head-shoulder-contour.

4.4.5 Performing Skin Color Segmentation of Input Images

For segmentation of the input image, the processing steps outlined in Section 4.3 are performed using the modeling and initialization methods described in the preceding sections. Image acquisition is performed with a Sony EVI D31 video camera and a Hauppauge WinTV multimedia TV card under Linux. The Video for Linux Two (V4L2) framework is applied for accessing the video data from the TV card. Image enhancement is carried out through calculating the normalized r-g color values (see Section 4.1) from the RGB color values. For the iconic feature extraction, the skin likelihood is calculated for each individual pixel based on its color and the current color model. Depending on the type of modeling, the overall skin probability value for the pixel x is determined:

Unimodal Gaussian for each region: With an unimodal Gaussian for each individual skin-colored area, the overall probability of a pixel is calculated as the maximum of the skin likelihood of the individual Gaussians from Eq. 4.2:

$$p(\mathbf{x}) = \max_i p_i(\mathbf{x}) \tag{4.10}$$

In this calculation, no information about the previous position of a specific region is incorporated. In a domain where the individual regions are tracked, the search for the maximum value is restricted to the Gaussians belonging to regions in the vicinity of the pixel position.

One mixture of Gaussians for complete image: If a mixture of Gaussians is used for modeling the skin color, the M individual mixture components (see Eq. 4.2) have to be added up with their weights c_i to give the overall pixel probability:

$$p(\mathbf{x}) = \sum_{i=1}^{M} c_i \, p_i(\mathbf{x}) \tag{4.11}$$

As a result, both types of modeling provide for every pixel its skin likelihood as iconic feature giving a skin probability image for the complete input image. This probability image is binarized using a classification threshold S_{class} to obtain a label image containing skin and non-skin pixels.

The value of the threshold S_{class} has to be chosen carefully, a low threshold will potentially classify a larger number of non-skin pixels with skin-like color (false positives) while a high threshold may not classify all skin-colored pixels correctly (false negatives) due to inhomogeneities caused by, for example, shades. The probability values $p(\mathbf{x})$ that are encountered in the probability image depend on the type of modeling. Especially

when using a Gaussian mixture, the variances of the individual mixture components influence the range of probability values that is occupied by skin pixels. It is therefore advantageous to use an adaptive threshold that is based on the actual probability values present in the training set. To obtain the classification threshold S_{class}, the training pixels used for constructing the Gaussian model are also used as test set for building a histogram of the probability values generated by the Gaussian model. The probability value histogram is analyzed starting at the bin representing the largest probability value p($\mathbf{x}$). The bin counts are successively added up until the sum contains more than 98.5 % of all N_{train} training pixels (see Eq. 4.12). Setting the threshold to classify a fraction of 98.5 % of the training pixels correctly has been determined empirically to ignore spurious outliers contained within the last 1.5 % of the training pixels. The probability value represented by the last histogram bin added is taken as the threshold S_{class} for skin color classification, i.e., all probability values below this threshold are classified as non-skin.

$$Pr(Y > S_{class}) = 0.985 * N_{train}, \qquad Y = p(\mathbf{x}) \tag{4.12}$$

To remove isolated pixels classified as skin and provide a more homogeneous result, a median of size 5×5 is applied to smooth the label image. Next, a connected components analysis is carried out in the segmentation step to obtain the region segmentation result. Subsequently, the feature extraction step calculates polygonal descriptions of the image regions and region features like, e.g., compactness, pixel size, center of mass.

For the example image depicted in Fig. 4.12, we trained an unimodal Gaussian and a Gaussian mixture using the face initialization method (see page 69).

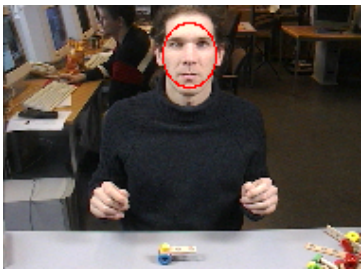


Figure 4.12: Example input image with face detection results.

Figure 4.13 and 4.14 depict the Gaussian model, the resulting label image, and the median filtered label image for the unimodal Gaussian and a Gaussian mixture with three components, respectively. Note that the unimodal Gaussian is a coarser model for the actual skin color distribution, therefore the probability image contains more pixels of the background classified as skin compared to the probability image for the mixture of Gaussians. In the depicted example containing no larger skin-colored background objects, the median filtering removes these isolated background pixels. However, if the background contains areas with a color similar to skin (e.g., a wooden desk), an unimodal Gaussian is more likely to classify many non-skin pixels as skin. If too many background

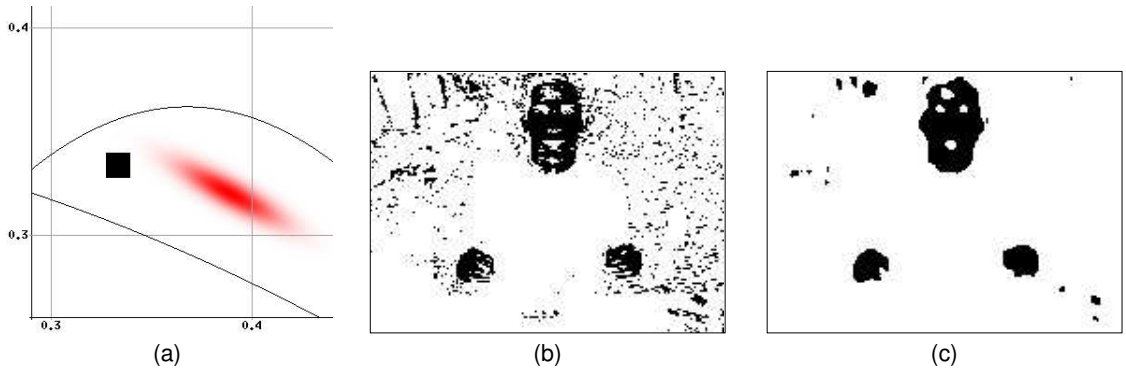


Figure 4.13: Example of pixel classification using a single Gaussian: (a) the unimodal Gaussian in r-g color space; (b) the label image; (c) the label image after median filtering.

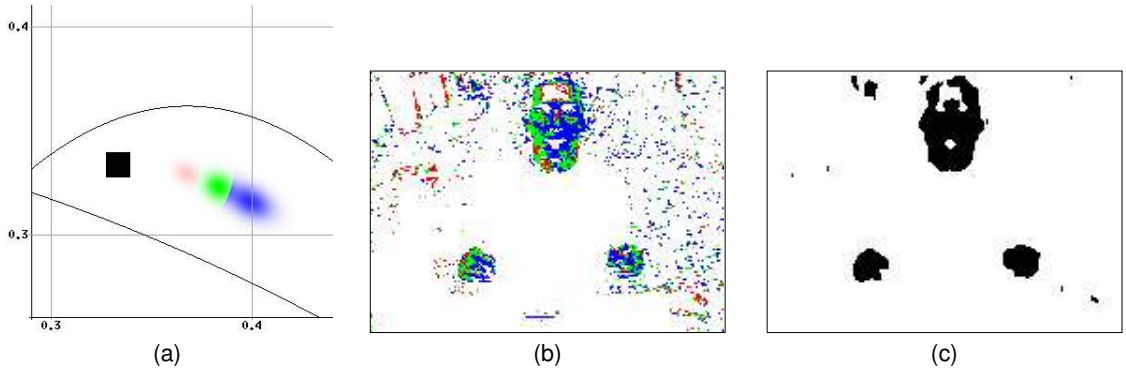


Figure 4.14: Example of pixel classification using a mixture of Gaussians: (a) the three mixture components in r-g color space; (b) the label image with the color indicating the component having the highest probability; (c) the label image after median filtering.

pixels classified as skin are close to each other, median filtering does not result in removal of this area but instead in smoothing of the area leading to a wrong label image.

The segmentation scheme described above operates on the iconic level to construct the label image. Applying a threshold to the iconic probability value ignores information that may have been present in the overall probability image. For example, a face or hand results in an area of high probability in the probability image. This spatial relation between the iconic probability values can be used for a segmentation scheme that focuses on finding elliptical regions representing human hands and faces [98, 57]. In such a scheme, a pixel that lies in an area with many pixels having high probability values and forming an elliptical shape is assigned the 'skin' label even if its probability value is below the threshold. In this way, the influence of partial shadings or non-skin objects (e.g., a ring on a finger, glasses) can be reduced and the form of the segmented region is closer to

the 'expected' elliptical form. Especially for performing face detection (see Section 4.4.3), such a top-down segmentation scheme is advantageous as it provides segmented regions whose form and COM is more suitable to initialize the search process. However, the associated computational cost of fitting ellipses of arbitrary size and orientation to the probability image is too high for real-time processing. Only in domains where the ellipses can be restricted, for example to 'face' ellipses with a fixed orientation and size, this processing scheme may be applicable in real-time segmentation approaches.

4.4.6 Updating the Skin Color Model

To realize an adaptive skin color segmentation, the skin color model generated in the initialization step has to be adapted to the current appearance of the skin-colored object. In case of a perfect segmentation, all pixels belonging to a skin-colored object would have been segmented correctly despite of a changed appearance. In this case, the segmented object region $R(i)_{segment}$ and the image region for updating $R(i)_{update}$ would be identical, i.e., all segmented pixels could be used to update the skin color model.

Under realistic circumstances, however, the appearance variations within a skin-colored area are often so large that not all pixels are correctly segmented. Consequently, the segmented region $R(i)_{segment}$ represents only those pixels that are sufficiently close to the current skin color model. Therefore, it is advantageous to construct a larger update region $R(i)_{update}$ that contains most or all of the pixels belonging to the object:

- If a face detection algorithm is active, every tracked skin colored region $R(i)_{segment}$ is used as a search area for recognizing faces. If a face has been found at the position of a skin-colored region, $R(i)_{update}$ has the form of an ellipse with the size equal to the approximate face size that is available from the face detection algorithm. The position of the detected face is taken to be the position of $R(i)_{update}$.

- For skin-colored regions that exhibit motion, the update region $R(i)_{update}$ is generated by enlarging the segmented region $R(i)_{segment}$. If the shape of the object is known, e.g., a hand with fingers stretched out, this shape model could be matched to the segmented region to construct $R(i)_{update}$. For all skin-colored objects with unknown or flexible shape, such a priori knowledge is not available and therefore the polygon describing $R(i)_{segment}$ is stretched to describe a region with an area two times the size of the segmented region. This enlarged region forms the update region $R(i)_{update}$.

For the example image shown in Fig. 4.12, the two different types of update regions are shown in Fig. 4.15.

All pixels within $R(i)_{update}$ that do not lie inside the skin locus (Eq. 4.4-4.7) are discarded. This step removes all pixels that are contained in the update area but have no skin-like color like, e.g., glasses, rings, or parts of the background resulting from

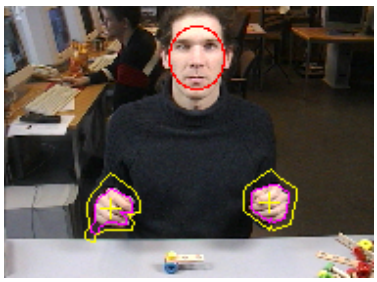


Figure 4.15: Example of the update regions R_{update}. The ellipse originating from the recognized face is drawn in red while the segmented hand regions are drawn in purple and the stretched update regions are drawn in yellow.

the update ara stretching. However, pixels from image areas having a color similar to skin, e.g., a wooden desk, are not removed by skin locus filtering. To avoid adapting the color model to background areas with skin-like color, we use a training threshold S_{train} to select only those pixels that exhibit a skin color close to the current skin color model. Similar to the classification threshold S_{class} (see Eq. 4.12), the training threshold is calculated from the probability value histogram of the previous training set. The threshold value is chosen such that the probability values p($\mathbf{x}$) of all N_{train} training pixels from the previous update step are above the threshold:

$$Pr(Y > S_{train}) = 1.0 * N_{train}, \qquad Y = p(\mathbf{x}) \tag{4.13}$$

Only those pixels in the current update area that have a skin probability above the training threshold S_{train} are considered for updating the skin color model. In this way, we enforce a smooth adaptation of the skin color model and can cope with hands and faces in front of wooden furniture and other skin-like background objects. After applying the skin locus filtering and the training threshold to remove all pixels that are not skin-like, the remaining training pixels contained in $R(i)_{update}$ are used for updating the Gaussian model:

Updating the Unimodal Gaussian

The mean and covariance of the unimodal Gaussian model can be calculated directly from the training pixels $\mathbf{x}_{training,j}$ in the update region:

$$\boldsymbol{\mu}_t = \frac{1}{N}\sum_{j=1}^{N}\mathbf{x}_{training,j} \tag{4.14}$$

$$\boldsymbol{\Sigma}_t = \frac{1}{N}\sum_{j=1}^{N}(\mathbf{x}_{training,j} - \boldsymbol{\mu}_t)(\mathbf{x}_{training,j} - \boldsymbol{\mu}_t)^T \tag{4.15}$$

Note that the values for $\mathbf{\Sigma}_t$ have to be constrained to lie between a minimum and a maximum variance to prevent the Gaussian model from becoming either too specific or too generic. We determined the variance boundaries empirically to $\sigma_{min} = 0.0002$ and $\sigma_{max} = 0.0008$ for both, the variance Σ^{rr} of the r values and the variance Σ^{gg} of the g values.

Updating the Mixture of Gaussians

For updating the mixture of Gaussians, the training pixels of all update regions $R(i)_{update}$ are collected. Using this training set, the parameters of the individual components of the Gaussian mixture are calculated using a *k-means* clustering algorithm from the ESMERALDA toolset [25]. The number of mixture components M is selected based on the application domain (see page 63). Clustering is done by choosing an initial set of cluster centers based on the first M color values. Subsequently, the remaining color values are processed one-by-one to update the initial cluster centers. For every training pixel $\mathbf{x}_{training,j}$, the following three steps are performed:

1. assign the pixel $\mathbf{x}_{training,j}$ to the closest cluster center.
2. calculate the new center of mass for this cluster.
3. update the cluster center with the calculated center of mass.

After processing all color values, the final clusters are used to compute the mean and variance of the Gaussian mixture components similar to Eq. 4.14 and Eq. 4.15.

4.4.7 Faster Segmentation based on Regions of Interest

To reduce the time necessary for image processing, the segmentation can be restricted to image areas known to contain skin-colored objects. To facilitate such a Region of Interest (ROI) segmentation, all interesting skin-colored objects have to be selected during the initialization phase. After initialization, these objects are tracked over time and only the image areas in the vicinity of objects are analyzed in the segmentation step. For this purpose, an image area $ROI(i)_{classify}$ is constructed for every object by calculating the bounding box of the region $R(i)_{segment}$ segmented in the previous image.

To cope with moving objects, the area $ROI(i)_{classify}$ has to be enlarged to ensure that it still contains the complete object. The enlargement is depending on the usual speed of objects in the domain and the segmentation frequency. Only the pixels within $ROI(i)_{classify}$ are classified using the associated skin color model. If object-specific unimodal Gaussians are used, only the Gaussian $G(i)$ belonging to $ROI(i)$ is evaluated and updated. The resulting partial label image is processed in the same way as the overall label image, i.e., median filtering and a connected components analysis are performed to obtain the segmentation result.

While the selection of parts of the input image for segmentation reduces the computational cost, it prevents the system from detecting new skin-colored objects appearing in image areas that are not in a region of interest. To enable the construction of additional ROI's for new objects, a special process would need to regularly scan the complete image and construct new ROI's. If object-specific unimodal Gaussians are used, one of the initialization methods described in Section 4.4.4 has to be performed for this purpose. As such a detection of new objects has not been implemented, the ROI-based adaptive skin color segmentation is currently only applicable in domains complying to a closed-world assumption, i.e., where no new objects are introduced after initialization.

4.4.8 Compromises necessary for Real-Time Image Processing

An important design consideration for real-time skin color segmentation is the available processing power. The performance of current general purpose workstations is not sufficient to process images in full resolution in real-time, i.e., at a frame rate of 25 Hz. In the trade-off between frame rate and image resolution, the more important aspect for trajectory-based gesture recognition is the frame rate. Missing frames result in coarser trajectories and perhaps even in missing details of the gesture. A lower image resolution only poses a problem if the hand performing the gesture is very small and therefore only represented by a few pixels. In this case, the misclassification of a few pixels could already result in loosing the hand and therefore a larger image resolution should be used.

To obtain the complete trajectory, i.e., segment all frames of the image sequence in real-time with 25 Hz, we use subsampled images resulting in a lower resolution. Our approach works with a subsampling of 4 resulting in an image size of 192 x 144 pixels. This is a reduction to 6,25% of the original number of pixels and is sufficient for capturing the movements of hands in images showing a single human close to the camera which is our primary concern for gesture recognition.

For some applications like hand posture recognition or in domains with a smaller zoom, working with a higher resolution would be desirable. In order to allow the application of the developed segmentation approach to other domains without real-time constraints or with more computational power, all algorithms described in this chapter are implemented resolution-independent and directly scale up to better resolutions. Last but not least, it should be noted that not only the available computational power but also the transfer of the image data from the frame grabber to the main memory is an important aspect relevant for the real-time ability of the overall system.

The developed segmentation system processes images with 192 x 144 pixels in real-time with 25 Hz using a Gaussian mixture with three components on a 1.133 GHz Pentium® III processor (402 SPECfp, 568 SPECint). Updating the mixture components based on regions exhibiting motion (see Section 4.4.6) is performed in every time step. The processor load performing segmentation and updating at 25 Hz is around 80 %, leaving enough time for distributing the results to other machines for further processing.

Using the face detection algorithm (see Section 4.4.3) to determine the skin-colored

regions that contain a face and are therefore suitable for updating the Gaussian mixture reduces the frame rate due to the computational load of performing face detection. The resulting frame rate depends on the number of skin-colored regions that are examined by the face detection algorithm. Segmentation, face detection, and adaptation are performed for an average number of three skin-colored regions at a frame rate of approximately 6 Hz on the 1.133 GHz Pentium® III processor.

4.5 Results

The proposed adaptive segmentation approach aims at finding in real-time hands and faces in image sequences with varying lighting conditions. In this section, we will present the segmentation accuracy qualitatively as the effort of a quantitative evaluation does not match its limited expressiveness: the often inferior lighting conditions in typical image sequences and the adaptive nature of the proposed processing scheme would require the evaluation of a large number of 'representative' image sequences to obtain a well-founded quantitative result. Even such a quantitative result is not comparable to the evaluations performed on typical photographs (see Section 4.1), as the image quality encountered in automatically acquired image sequences differs substantially from the data used in [16, 51]. Moreover, the processing of image sequences for the extraction of motion trajectories requires small processing times to obtain a high frame rate instead of exact segmentation results.

The dynamic nature of image sequences makes the presentation of results difficult, as only 'snapshots' of the performance of the adaptive skin color segmentation can be shown. Four images of a sequence depicting a human constructing an assembly in the Situated Artificial Communicator domain are shown in the first row of Fig. 4.16 with the segmentation results overlaid. In the image depicting the scene in the first column, the red ellipse from face initialization is visible that was used to generate the initial Gaussian mixture with three components that is depicted in the first column in the second row. The third row shows the label image generated from assigning each skin-like pixel the color of its associated mixture component. The label image after binarization and applying a median filtering is provided in the fourth row.

A second example for the segmentation quality is given in Fig. 4.17 for a scene depicting a human drinking a cup of coffee in a typical office setting. Although the human hand is not completely labeled as skin in the smoothed label image, the algorithm is capable of keeping track of the hand and the variations in the skin color distribution of the hand during the drinking action. Despite of the color of the wooden desk looking similar to skin, the adaptation of the skin color model is not distracted. Comparing the skin classification at the pixel level in this domain to the results in the assembly construction domain depicted in Fig. 4.16 shows that here only a few background pixels are classified as skin. This is due to the different lighting conditions in the office and causes the skin color model to be located at a different position within the skin locus.

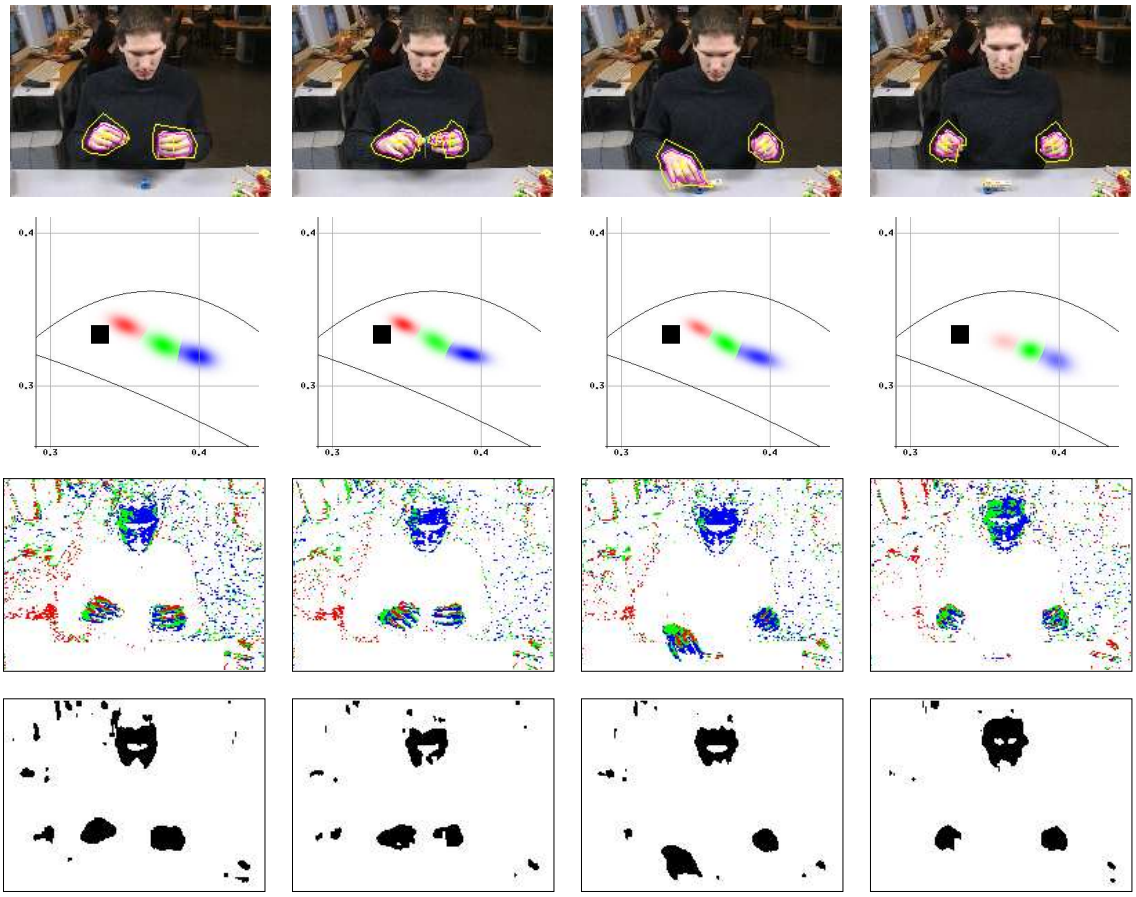


Figure 4.16: Exemplary skin segmentation results for the assembly construction domain. First row: input images with update areas from face detection (red) and motion (blue). Second row: resulting skin color model. Third row: label image showing best mixture component. Fourth row: label image after median filtering.

The adaptive skin color segmentation was also applied for the detection of persons moving in an office environment [31]. In this domain, a mobile robot is used to observe the persons and therefore the processing power is limited. Consequently, unimodal Gaussians are used to model the skin color distributions of the individual faces. The mobile robot is equipped with a special small-size computer containing a 500 MHz Pentium® III processor. On this system, the frame rate is roughly 3 Hz if on average two skin-colored regions are present in the input images. For tracking humans moving in front of the mobile robot, this frame rate is sufficient to perform skin color adaptation and to realize a human-machine interface for interaction between humans and the robot. We have successfully integrated the adaptive color segmentation into a framework that allows the robot to track several humans [53] and detect the communication partner of

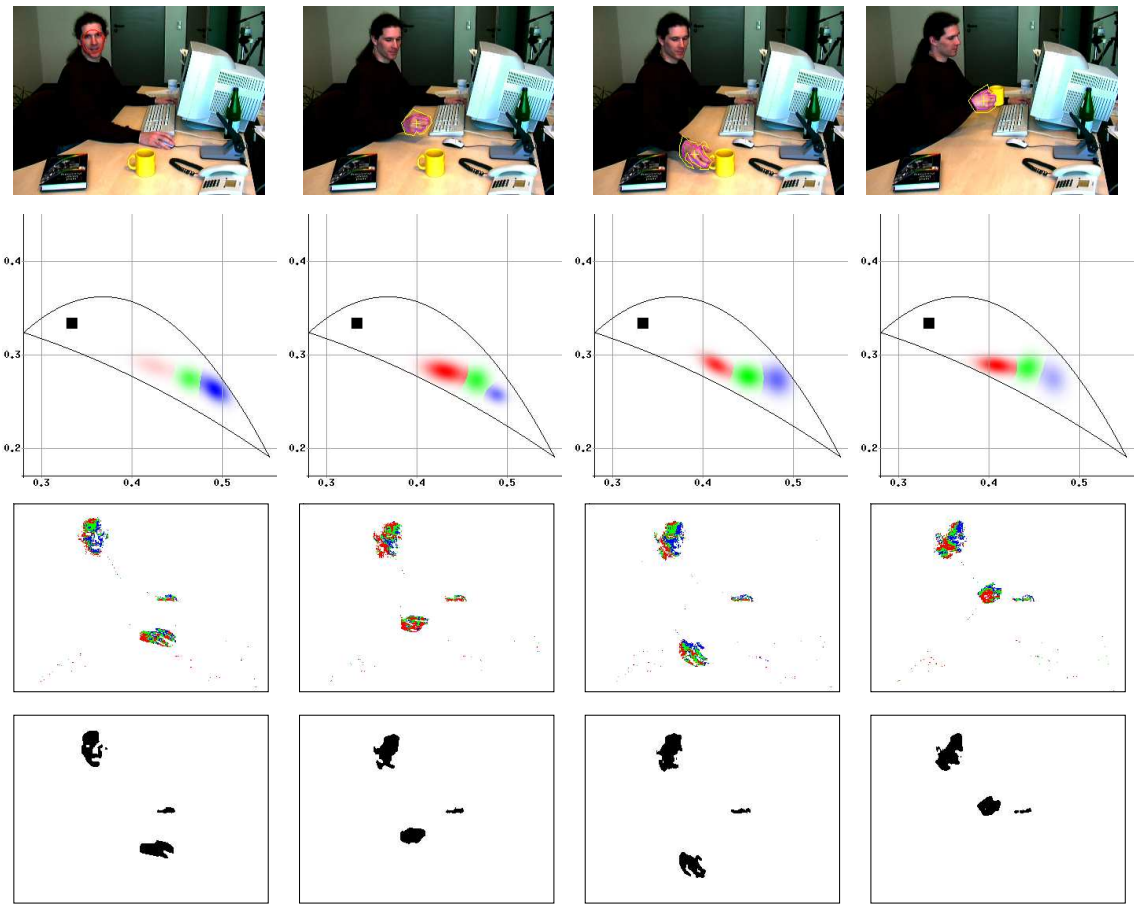


Figure 4.17: Exemplary segmentation results for a scene depicting a person drinking a cup of coffee.

the robot [61]. The evaluation of this integrated system for human-robot-interaction demonstrates the relevance of segmenting skin color and subsequently detecting faces in environments encountered by a mobile robot [30].

4.6 Summary

In this section, we presented our adaptive approach for segmenting human skin color under varying lighting conditions. The properties of human skin that are relevant to its modeling have been pointed out and the basic processing steps necessary for image segmentation were outlined. The realization of the adaptive approach was described in terms of the basic processing steps. The implemented segmentation system allows us to switch between two different methods for skin color modeling: (1) using an unimodal Gaussian for each image region and (2) adapting a mixture of Gaussians for the

complete image. The differences in segmentation quality for the two modeling methods were demonstrated with example segmentation results. Depending on the properties of the application domain, the appropriate modeling technique can be selected. We presented the segmentation results for two different domains qualitatively: (a) segmenting the hands and face of a single human acting in the baufix® construction domain and (b) segmenting a human acting in an office environment. With the presented adaptive approach we therefore have now the basis for approaching the recognition of hand motions in arbitrary domains with varying lighting conditions.

5 Visual Activity Recognition

For the recognition of construction actions in the Situated Artificial Communicator domain, the symbolic action detection presented in Chapter 2 relies on the reliable detection of appearing and disappearing parts in the scene. The human must obey the restrictions imposed by the rule-based approach to enable the recognition of his actions. For example, placing parts outside the observed area directly leads to errors in the action detection, as the symbolic data is extracted based on the observation of the table scene.

Most of the restrictions imposed by the rule-based approach can be avoided with a method to visually monitor the construction actions. While in the symbolic approach rules operate on symbols that have been extracted previously, the visual activity recognition approach applies pattern recognition techniques that operate directly on the sensor data. This direct analysis of sensor data allows us to incorporate the influence of noise in the input data in the recognition process and therefore makes a data-driven activity recognition more robust.

In this chapter, we will present an activity recognition for the Situated Artificial Communicator domain that is based on visually observed motions of the hands to recognize 'pick/place', and 'connect' activities. The positions of all skin-colored objects in a single image are obtained using the adaptive skin color segmentation method introduced in the previous chapter. Tracking the segmentation results over time to construct the motion trajectories is done using Kalman filters (see Section 3.2.1). Out of all tracked skin-colored objects, the trajectory representing the hand motion is selected based on the distance and duration of successful tracking. Using a recursive Bayesian filter to recognize the activity based on the continuous trajectory data can be done either with Hidden-Markov Models or with particle filtering. Reaching good recognition quality with an HMM-based recognition approach requires the state pdf to exhibit Gaussian modes if the underlying process has a continuous nature (see Section 3.2.2). In contrast, particle filtering techniques do not impose any restrictions on the state pdf (see Section 3.2.3). In order to realize a generic activity recognition approach, we therefore chose to apply a variation of particle filtering for the analysis of the hand trajectory. The overall system for the recognition of construction activities is designed to enable the easy application of the approach in different domains.

Note that although the human performs construction actions, their classification based only on sensory data allows us to recognize just his 'generic' activities. For example, moving the hand down to the table and then moving it up again is a generic activity and only the knowledge of the construction domain allows us to call this motion a 'pick/place'

activity. Additionally, due to the missing symbolic information, the activity recognition does not provide the information *what* part is being manipulated and, consequently, whether it is a 'pick' or a 'place' activity.

The contents of this chapter are organized as follows: We will start in Section 5.1 with a review of the research work on activity recognition, concentrating on the recognition of gestures executed by the hands. The tracking of the skin-colored regions extracted with the adaptive image segmentation is the topic of Section 5.2. Based on the obtained motion trajectories, the application of the particle filtering algorithm to recognize two-handed construction activities will be presented in Section 5.3. Results for the Situated Artificial Communicator domain are given in Section 5.4. The Chapter concludes with a summary of the visual activity recognition.

5.1 Related Work

The increasing processing power of standard workstations has resulted in a growing interest in research on image sequence processing to develop systems that can reason about dynamic scenes.

One research direction deals with the observation of large scale scenes. The list of domains varies from the monitoring of people in an office environment [77, 3] through the understanding of football scenes [46] to the tracking of interacting pedestrians in an outdoor scene [74] and the analysis of large traffic scenes [55]. Due to the different domains and the diverse aims of the analysis, a wide variety of vision techniques and pattern matching algorithms is used in these approaches.

Another research direction is the recognition of gestures executed by a single human through moving only his arms and hands, i.e., without changing his overall position in the scene. As we are interested in recognizing gestures of a single human manipulating the environment, we will concentrate here on these approaches. The majority of the published work deals with gestures of the movement and activity type as they are characterized by motion of the hands only. In this section, some well-known publications highlighting different approaches towards visual activity recognition will be reviewed. For more details on the large variety of algorithms for hand and body motion recognition, the reader is referred to one of the review articles (cf. e.g. [54, 76, 104, 68]).

An early work by Starner and Pentland concentrated on recognizing the gestures of hands performing American Sign Language (ASL) [91]. Their method for recognizing ASL with Hidden-Markov-Models (see Section 3.2.2) reaches a recognition accuracy of over 91% on a set of 40 gestures. This rate can be increased to above 99% by using a grammar for ASL sentences. In this approach, the image processing part is greatly simplified to make the feature extraction easy. The subjects have to wear a yellow glove on the right hand and an orange glove on the left hand. The segmentation and tracking of the colored gloves is accomplished with the Pfinder system ('person finder', see [103] and also Section 4.2). Based on the trajectory data of the two hands, the activities are

recognized with standard HMM-techniques.

An approach utilizing spatio-temporal feature trajectories for matching observed motions to learned motion models is the VIGOUR system by Gong and Sherrah for tracking multiple people and recognizing their activities [87]. In the presented examples, the system can deal with three persons sitting in front of a camera. The head and the hands are tracked for each person based on regions extracted with an adaptive skin-color segmentation approach [81]. Gesture recognition is performed in a person-centered way based on the x and y offsets of the tracked hands relative to the tracked head of a person. The examples present the gestures 'waving' and 'pointing' to the left or right.

The approach by Black and Jepson [12] aims at recognizing gestures which are drawn on a whiteboard. They concentrate in their approach on the trajectory recognition aspects and simplify the image processing by requiring the user to make the gestures by drawing with a distinctively colored 'phicon' (physical icon). For the recognition of the gestures a particle filtering method is applied. A detailed explanation of the recognition algorithm can be found in Section 5.3, as we have applied an extension of the recognition framework proposed by Black and Jepson for recognizing the two-handed construction gestures.

The previously outlined publications demonstrate that there exists a large variety of different techniques that have been applied for reaching a similarly large variety of aims. However, only systems operating in very limited environments reach a performance sufficient for their use in real application domains. Therefore some researchers consider the field to be still in its early stage of development [68]. The direct comparison of the different approaches is not possible, as they have been tailored to the specific applications and use distinct preprocessing techniques. Especially the dependence of the different techniques on certain properties of the preprocessing algorithms and the used example domains results in the conclusion that there is no single approach that is well-suited for a larger range of applications.

To enable gesture recognition in a wide range of domains, our approach is based on the adaptive skin color segmentation presented in the previous chapter. The assembly construction domain is chosen as this domain allows us to compare the results of the activity recognition with the rule-based symbolic action detection and emphasizes the need for the integration of symbols and sensory data to recognize manipulative gestures.

5.2 Tracking Skin-Colored Objects

For obtaining the motion trajectories of the hands in image sequences, we first obtain the hand regions in individual images using the adaptive skin color segmentation outlined in Chapter 4. The skin color of a single human constructing assemblies is modeled using a Gaussian mixture with three components (see Section 4.4 for details). To extract the motion of the hands in an image sequence, the regions segmented in the actual image have to be linked to the regions in the previous image. This can be done by simply taking the

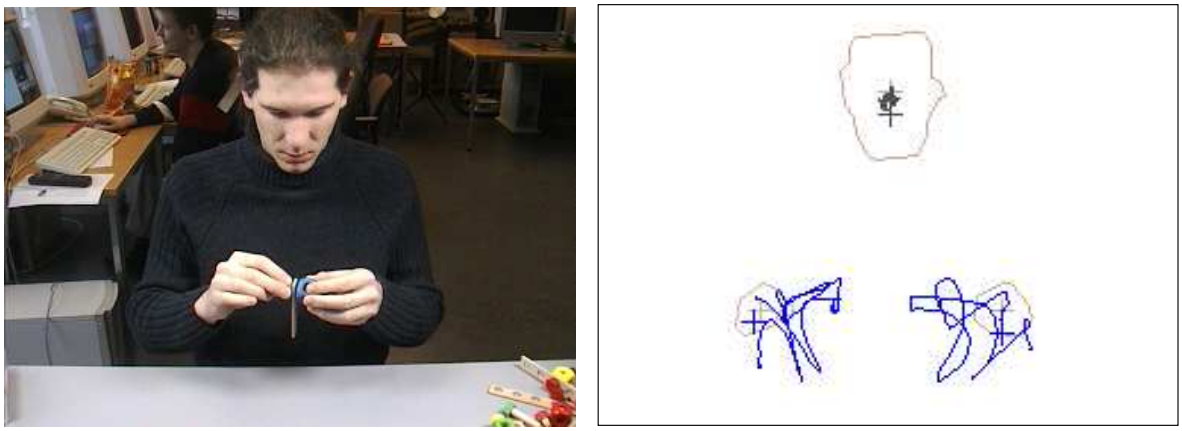
last position of a region as prediction for the current position. However, incorporating higher-order information about the speed or acceleration of a region allows us to better predict its position in the actual frame and thereby resolve possible ambiguities.

For tracking all segmented regions over time, we developed a tracking module [32] based on Kalman filtering (see Section 3.2.1). The system model of the Kalman filter represents the kinematic motion equation of one point using a constant acceleration model. The COM of each region is tracked individually using this system model. Tracking all segmented regions with Kalman filters requires the following two steps for each new image segmentation result:

1. All previously initialized Kalman filters are associated with the region which is closest to the predicted new position. A distance threshold is used to avoid assigning a region which is too far away.

2. For every untracked region a new Kalman filter is initialized. There is no spatial restriction to allow skin-colored regions to appear anywhere in the image. This is especially important if skin-colored regions may be temporarily occluded, e.g., hands can be occluded by some objects between them and the camera.

As an example, consider the construction activity depicted with a snapshot in Fig. 5.1(a). A bolt and a bar have been taken from the table by the two hands and have



(a) snapshot image (b) trajectories

Figure 5.1: Activity example and obtained trajectories for taking two parts, connecting them together, taking a third part, and establishing a bolted connection to form an assembly.

been put together. Subsequently, a cube has been taken and is now being connected to the bolt to form an assembly. The trajectories resulting from tracking the skin-colored regions during the complete construction activity including the placing of the assembly

on the table are shown in Fig. 5.1(b). Note, as the face of the constructor is slightly moving its region is tracked by a third Kalman filter. Different from the trajectory of the face region, the trajectories of the two hand regions exhibit large motions. These trajectories are drawn in blue to indicate that they represent potential hand candidates. Selecting the trajectories of the hands will be covered in more detail in Section 5.3.

Coping with Merged Regions due to Interacting Hands

For monitoring human hands, a special problem is the joint manipulation of parts by two hands. If the hands are next to each other or overlap, the image segmentation may extract a single skin-colored region containing the area of both hands. Consequently, the two Kalman filters previously tracking two individual regions are assigned to this merged region. However, this assignment of two Kalman filters to the same region is not restricted to the case of two adjacent hands. Other cases include a hand taking a skin-colored object or coming close to a face.

In order to allow for individual tracking of two objects while their regions are merged, an imaginary COM is calculated for each Kalman filter [101]. The two imaginary COMs are obtained by dividing the merged region into two individual regions based on the perpendicular bisector of the line between the two COMs from the previous image. For each individual region its COM is calculated and used for updating the Kalman filter. Fig. 5.2 shows an example for two merged hand regions.

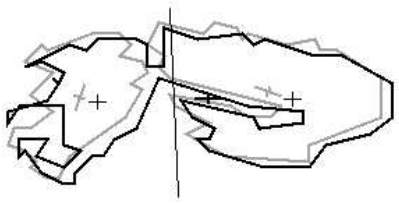


Figure 5.2: A merged region (black) divided into two parts based on the perpendicular bisector of the line between the two COM's of the regions in the previous time step (brown).

5.3 Recognition of Construction Activities

In the following we will show how the recognition of construction activities in the baufix®-scenario is carried out based on the hand trajectories extracted in the tracking step.

First of all, we need to select those two skin-colored image regions that represent the two hands of the human constructor. In the field of view of the camera can be the face of the constructor or body parts of additional people or, last but not least, objects with skin-like color. Consequently, the regions found in the segmentation step and tracked by

the Kalman filters have to be analyzed to decide whether they represent hand regions. To select the two trajectories representing the hand motions of the human constructor, the 'history' of all trajectories is analyzed. For the recognition of construction activities, we consider the history to be comprised of

- the duration of successful tracking of a region, i.e., its lifetime
- the distance between the initialization point and the most distant point reached during tracking

Using this formulation of history, we are able to incorporate the dynamic nature of the hands for selecting the correct hand regions. Assuming that hands are acting in the domain while all other skin-colored regions including the face are not moving very much, the two regions with longest lifetime and largest distance from the initialization point are selected to represent the hands. After obtaining the two trajectories representing the hand motions, this data has to be analyzed to recognize activities. This is done with the CONDENSATION-based trajectory recognition algorithm introduced by Black and Jepson [12] for recognizing commands drawn on a blackboard with a specially colored marker. As pointed out in Section 3.2.3, the CONDENSATION algorithm as introduced by Isard and Blake [49] is a version of particle filtering where the importance density for sampling is chosen to be the prior density.

We will use in the following the notation as introduced in Section 3.2.3. Each gesture model μ is represented by a parameterized *activity model* $M^{(\mu)}$ consisting of a set of feature vectors $\mathbf{x}_t$ describing the motions of the hands during execution of the gesture with duration T:

$$M^{(\mu_i)} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T) \tag{5.1}$$

For the recognition of two-handed construction actions, each feature vector contains the x- and y-velocities of both hands and the distance d between them:

$$\mathbf{x}_t = (x_L, y_L, x_R, y_R, d) \tag{5.2}$$

For the generation of a new model trajectory, several activity trajectories are recorded and the mean trajectory as well as its deviation are calculated after manual segmentation. To enable the recognition of several different complex gestures comprised partly of identical basic gestures, *parent models* $\mathbf{R}^{(\nu)}$ are used for complex gestures which consist of concatenations of child models $M^{(\mu_k)}$ for basic gestures:

$$\mathbf{R}^{(\nu)} = \{M^{(\mu_1)}, \ldots, M^{(\mu_k)}\} \tag{5.3}$$

The child models are linked together by *transition probabilities* $a_{ij}^{(\nu)}$ that are used to determine the next child model after recognizing the current child model i:

$$a_{ij}^{(\nu)} = Pr(\mu_i \rightarrow \mu_j) \qquad \forall \quad 1 \leq i, j \leq k \tag{5.4}$$

The transition probabilities a_{ij} are set by hand but could be learned automatically given sufficient training material. Defining the transition probabilities enables the construction of different complex gestures with partly identical basic gestures. For the recognition of construction activities, the transition probabilities are only used to define the order of the child models resulting in:

$$a_{ij}^{(\nu)} = \begin{cases} 1 & \text{, if} \quad j = i+1 \\ 0 & \text{, else} \end{cases} \qquad \forall \quad 1 \leq i, j \leq k \tag{5.5}$$

The overall recognition system is implemented to recognize the three complex activities $Pick^L/Place^L$, $Pick^R/Place^R$, and *Connect*. The distinction between the left and the right hand is necessary as the two hands exhibit different trajectories. Instead of vertical trajectories pointing down on the table, the movement of each hand has a slight 'skew' pointing to the center of the table (see also Fig. 5.1(b)). The activities are represented as parent models consisting of the following child models:

$$\begin{aligned} \text{Pick}^L/\text{Place}^L\text{:} \quad & \mathbf{R}^{(1)} = \{Hand\ down^L, Hand\ up^L\} \\ \text{Pick}^R/\text{Place}^R\text{:} \quad & \mathbf{R}^{(2)} = \{Hand\ down^R, Hand\ up^R\} \\ \text{Connect:} \quad & \mathbf{R}^{(3)} = \{Approach, Move\ away\} \end{aligned}$$

Recognition within the particle filtering framework is performed using a set of samples $\mathbf{s}_t^{(i)}$ containing parameterized activity models. With the above defined complex activities, each sample vector must contain a parameter ν denoting the parent model and a parameter μ indicating the current child model. The child model trajectory $M^{(\mu)}$ is parameterized with an amplitude scaling α and a scaling in the time dimension with ρ. The current position within the model trajectory that is used for matching the model with the observed data is defined by the time index ϕ. The influence of these parameters on the model trajectory is sketched in Fig. 5.3.

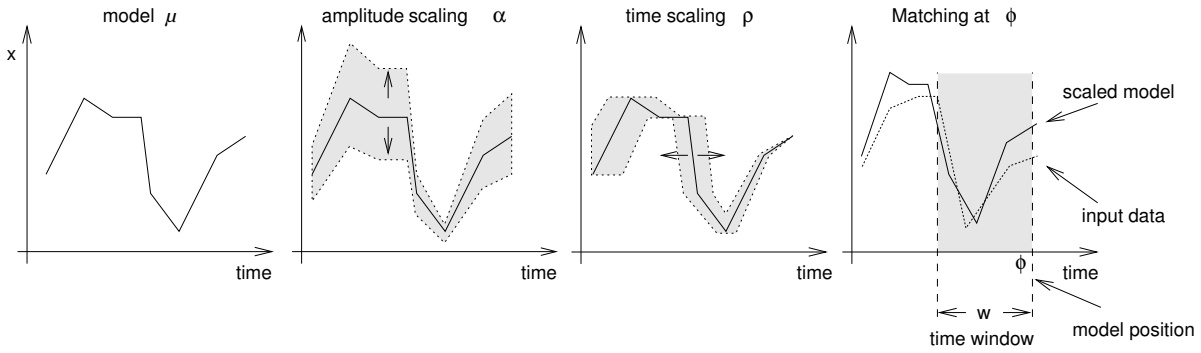


Figure 5.3: Influence of the scaling parameters α, ρ and ϕ in the model μ for matching with the observed data.

The complete representation of a sample is given by:

$$\mathbf{s}_t^{(i)} = (\nu_t, \mu_t, \alpha_t, \rho_t, \phi_t) \tag{5.6}$$

For tracking the observed data, the parameters in N samples of the sample set, having associated weights $\pi_t^{(1)}$, are propagated over time to find the best matching between the observed data and the model trajectories.

$$\left\{(\mathbf{s}_t^{(1)}, \pi_t^{(1)}), \ldots, (\mathbf{s}_t^{(N)}, \pi_t^{(N)})\right\} \tag{5.7}$$

The sample weights have to be calculated at every time step. Following the CONDENSATION algorithm (see page 45), the sample weights are proportional to the conditional observation density. This density is modeled by assuming independent elements in the five-dimensional observation vector $\mathbf{z}_t$ resulting in the calculation of the sample weights:

$$\pi_t^{(i)} \propto p(\mathbf{z}_t|\mathbf{s}_t^{(i)}) \quad with \quad p(\mathbf{z}_t|\mathbf{s}_t^{(i)}) = \prod_{j=1}^{5} p(z_{t,j}|\mathbf{s}_t^{(i)}), \tag{5.8}$$

The conditional observation density of each vector element, i.e., the fit between the model data $M^{(\mu)}$ and the velocities of both hands and the distance, is calculated from:

$$p(z_{t,j}|\mathbf{s}_t) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left\{\frac{-\sum_{u=0}^{w-1}\left(z_{(t-u),j} - \alpha M^{(\mu)}_{(x_{(\phi-\rho u),j})}\right)^2}{2\sigma_j^2(w-1)}\right\}. \tag{5.9}$$

Here w defines the size of the time window for comparing the model with the measured data (see Fig. 5.3). The value $\alpha M^{(\mu)}_{(x_{(\phi-\rho u),j})}$ is the jth component of the trajectory of the model μ interpolated at time $\phi - \rho u$ and scaled by α. The values σ_j indicate the standard deviations of the individual trajectories j of the model.

On initialization of the sample set, the parameters of each sample $\mathbf{s}^{(i)}$ are initialized:

- The parent model ν is sampled uniformly from $[1, \ldots, \nu_{\max}]$.
- The child model μ is sampled uniformly from $[1, \ldots, \mu_{\max}]$.
- The position within the model ϕ is initialized using $\phi = \frac{1-\sqrt{y}}{\sqrt{y}}$ with y sampled uniformly from the interval $[0, 1]$. In this way, the initial value of the position is biased towards small values.
- The scaling parameters α and ρ are sampled uniformly from the interval $[\alpha_{\min}, \alpha_{\max}]$ and $[\rho_{\min}, \rho_{\max}]$ respectively.

For the recognition of assembly construction activities we have three parent models ($\nu_{\max} = 3$) and six child models ($\mu_{\max} = 6$). Similar to Black and Jepson's approach [12], the scaling parameters α and ρ are both limited to the interval $[0.7 \ldots 1.3]$.

After initialization, the sample set is propagated by performing at each time step t the following three steps:

Select: Selection of $N - M$ samples $\mathbf{s}_{t-1}^{(i)}$ according to their respective weight $\pi_{t-1}^{(i)}$ from the sample set $\{(\mathbf{s}_{t-1}^{(1)}, \pi_{t-1}^{(1)}), \ldots, (\mathbf{s}_{t-1}^{(N)}, \pi_{t-1}^{(N)})\}$ of the previous time step. This selection scheme implies a preference for samples with high probability, i.e., they are selected more often. This is the resampling step of the SIR filter (see page 45). A fraction of 10% of the samples ($M = 0.1N$) is initialized at every time step in order to allow for the tracking of new models.

Predict: Prediction of the sample parameters $\mathbf{s}_t^{(i)}$:

$$
\begin{aligned}
\nu_t^{(i)} &= \nu_{t-1}^{(i)} \\
\mu_{i,t}^{(i)} &= \begin{cases} \mu_{i,t-1}^{(i)} & \text{, if } \phi < \phi_{\max} \\ \mu_j \text{ based on } a_{ij}^{(\nu)} & \text{, otherwise} \end{cases} \\
\phi_t^{(i)} &= \phi_{t-1}^{(i)} + \rho_t^{(i)} + \mathcal{N}(\sigma_\phi) \\
\alpha_t^{(i)} &= \alpha_{t-1}^{(i)} + \mathcal{N}(\sigma_\alpha) \\
\rho_t^{(i)} &= \rho_{t-1}^{(i)} + \mathcal{N}(\sigma_\rho)
\end{aligned}
$$

The $\mathcal{N}(\sigma)$ are normal distributions with zero mean and standard deviation σ. The values of α and ρ are limited to the interval $[\alpha_{\min}, \alpha_{\max}]$ and $[\rho_{\min}, \rho_{\max}]$, respectively. If sampling of a new value for α or ρ exceeds the interval consecutively for three times, a new sample $\mathbf{s}_t^{(i)}$ is initialized.

If $\phi \geq \phi_{\max}$ and $\mu_{i,t}^{(i)}$ is the last child model of the parent model ν, a new sample $\mathbf{s}_t^{(i)}$ is initialized.

Update: Determination of the weights $\pi_t^{(i)}$ based on the conditional observation density $p(\mathbf{z}_t|\mathbf{s}_t^{(i)})$ (see Eq. 5.8 and Eq. 5.9).

Figure 5.4: The three steps for iterative propagation of the sample set.

1. Selection of the samples with high weights $\pi_{t-1}^{(i)}$ for propagation.
2. Propagation of the parameters of each selected sample $\mathbf{s}_t^{(i)}$.
3. Calculation of the new sample weights $\pi_t^{(i)}$ based on the observation $\mathbf{z}_t$.

A detailed description of these three steps is given in Fig. 5.4. Using this processing scheme all samples $\mathbf{s}_t^{(i)}$ with a good fit of the model trajectory with the observed data $\mathbf{z}_t$ are propagated over time. By calculating for each model μ_k the mean values of the

sample parameters from all samples belonging to this model the most likely state of this model can be obtained. The actual classification of gestures is realized by calculating for every parent model ν the so-called *end probability* $P_{\text{end},t}$ of its last child model μ_k, indicating how many of the samples belonging to child model μ_k are close to the end of its model trajectory:

$$P_{\text{end},t}(\mu_k) = \sum_{i=1}^{N} \begin{cases} \pi_t^{(i)} & \text{, if } \mu_k \in \mathbf{s}_t^{(i)} \ \wedge \ \phi > 0.9\phi_{\max} \\ 0 & \text{, else} \end{cases} \tag{5.10}$$

Summing only over the sample probabilities belonging to a sample with $\phi > 0.9\phi_{\max}$ assures that the counted samples have been successfully propagated in the sample distribution long enough to match 90% of the model trajectory to the observed data.

At every time step the end probabilities are calculated for all parent models. A parent model is recognized if its end probability exceeds a recognition threshold D_{rec}. Note that the value of the end probability $P_{\text{end},t}(\mu_k)$ equals the fraction of the sample distribution that is populated by samples with ϕ close to $\phi_{\max}$ and belonging to the last child model μ_k of the parent model ν. Therefore, the value of the recognition threshold D_{rec} depends on the total number of child models and the number of 'final' child models. For a recognition system with a total of R parent models with different final child models and a total of S child models, the threshold is calculated from:

$$D_{rec} = 0.2 \times \frac{R(\text{different final child models})}{S(\text{total number of child models})} \tag{5.11}$$

Figure 5.5 shows example trajectories for a sequence of complex construction activities and Fig. 5.6 depicts the hand velocities as well as the end probabilities for this activity. For classifying the activites, a threshold of $D_{rec} = 0.2 \times (3/6) = 0.1$ was used.

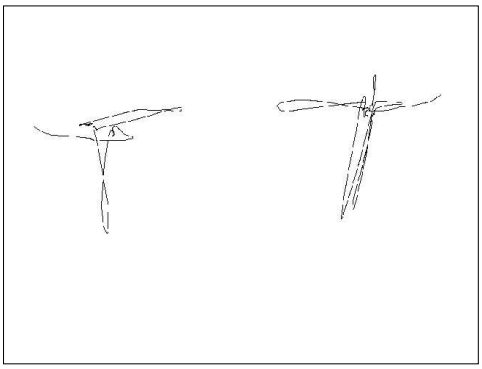


Figure 5.5: Trajectories of the hands taking two parts, putting them together and placing the assembly back on the table.

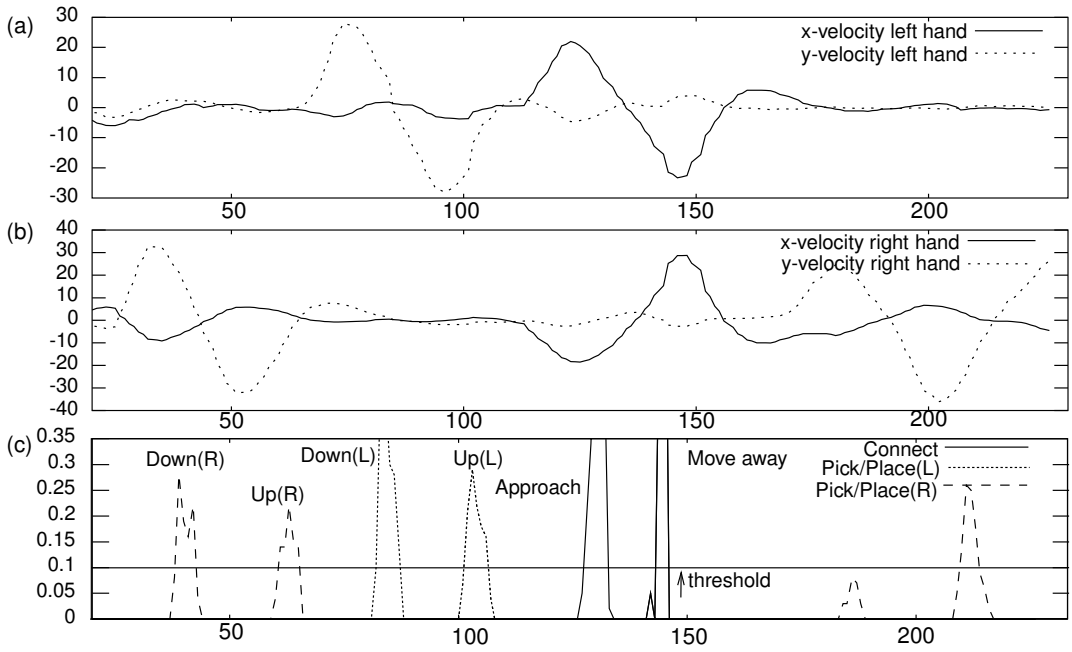


Figure 5.6: Velocities of the (a) left and (b) right hand and (c) probabilities for the model completion for the activity sequence Pick^R, Pick^L, Connect, and Place^R depicted in Fig. 5.5.

5.4 Results

The complete system was tested with several people performing a typical gesture sequence in the baufix® assembly construction scenario. The gesture sequence consisted of picking a bolt and a bar, putting the bar onto the bolt, picking a cube, and securing the parts with a cube screwed onto the bolt before placing the assembly back on the table. The construction sequence therefore contained 6 parent models (3 *Pick*, 2 *Connect*, 1 *Place*) with a total of 12 child models.

For monitoring the construction activities, an image size of 192 x 144 pixels was used. The images were processed in real-time at a rate of 25 Hz by the adaptive color segmentation using an unimodal Gaussian on a DEC Personal Workstation 433au (SPECInt95 13.9). At the time of the experiments, the available processing capacity was not sufficient to use a mixture of Gaussians. To enable a good skin color segmentation with an unimodal Gaussian, no background objects with skin-like color were allowed in the experiments. The resulting skin-colored regions were tracked with Kalman filters. Based on the history of the tracked regions, the two hand trajectories were selected. The trajectory information of the two hands was processed by the activity classification running in real-time on a DEC AlphaServer ES40 (SPECInt95 27.3). To enable real-time operation, a sample set with N=3500 samples was used and the time window for comparing the model trajectories with the observed trajectory was chosen to $w = 10$.

The overall test set consisted of 36 performances of the construction sequence. The gesture sequences were performed by 5 colleagues who were familiar with the capabilities of the system, i.e., the necessity to lift parts up and connect them 'in the air' instead of directly 'on the table'. The classification results for the test set are given in Table 5.1.

	Total Activities	Recognized Activities	
	#	#	%
Child models			
Hand down^L	57	51	89
Hand up^L	57	54	95
Hand down^R	87	78	90
Hand up^R	87	86	99
Approach	72	64	89
Move away	72	70	97
$\sum$	432	403	93
Parent models			
Pick^L/*Place*^{L}	57	50	88
Pick^R/*Place*^{R}	87	77	88
Connect	72	64	89
$\sum$	216	191	88

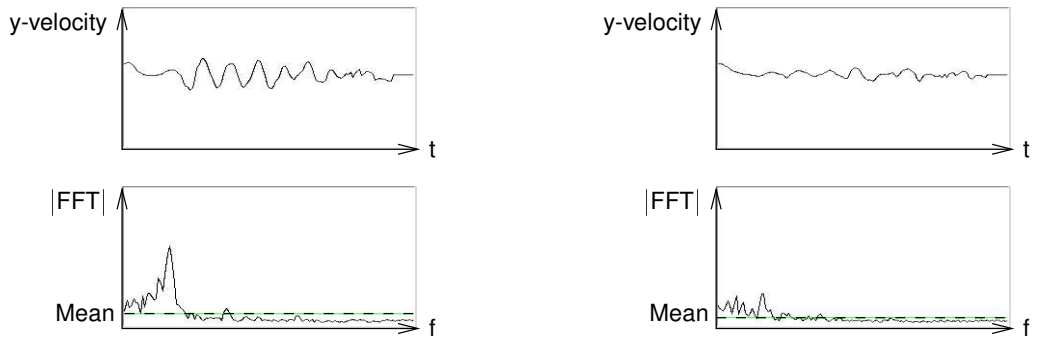
Table 5.1: Results for performing activity recognition on a test set of 36 gesture sequences depicting assembly construction actions.

To differentiate the *Connect* activity into either putting a baufix® element with a hole on a screw (*Put*) or mounting a threaded element on a screw (*Screw*), several experiments were carried out. The varying time scale of the screwing activity (depending on the length of the bolt) was difficult to model in the activity models and resulted in a poor recognition rate [33]. The variations between different persons in the way of turning and regrasping parts were another factor that also made the creation of typical models difficult. Therefore, we realized an additional postprocessing of the *Connect* activity to discriminate between *Put* and *Screw* activities.

Postprocessing for Differentiating Connect Activities

The postprocessing is performed if a *Connect* activity has been successfully recognized. For this purpose, the recording of trajectory data is started as soon as an *Approach* is detected. If the next recognized child model is a *Move away*, a *Connect* is recognized and the recorded data is used for postprocessing. If no complete *Connect* activity is detected, the recorded trajectory data is discarded and no postprocessing is carried out. As the characteristic of connecting a bolt with a thread is the repeated up and down movement of at least one hand during regrasping the threaded bolt, we perform a

Fourier transformation on the derivatives of the two Kalman trajectories, i.e., the hand velocities (see Fig. 5.7). This computation is carried out only for the y-velocity, as the hands execute the *Connect* activity usually in a horizontal orientation as depicted in the example on page 84. During screwing, at least one hand moves slightly up and down in the image while regrasping the bolt or the counterpart every half turn. Therefore, in case of a *Screw* activity the transformation exhibits high amplitudes in the low-frequency range (see Fig. 5.7(a)). For a hand which is not moving, the resulting transformation exhibits no characteristic peak. This is the case when a *Put* activity occurs or the hand holds the counterpart during a *Screw* activity (see Fig. 5.7(b)).



(a) Active hand (b) Passive hand

Figure 5.7: Velocities and their Fourier transformations for a *Screw* activity with (a) an active hand regrasping the bolt and (b) a passive hand just holding the threaded part.

The difference in the FFT-magnitude between the maximum amplitude and the mean value of each hand is used together with the duration of the activity as a three-dimensional feature vector for classifying the activity. Reference feature vectors for the activities *Put*, $Screw^L$, $Screw^R$, and $Screw^{LR}$ (both hands move up and down) were calculated based on 60 examples for each activity. An unknown activity is classified based on the smallest Euclidean distance between its feature vector and the reference feature vectors. An empirically determined threshold is used to reject the activity hypothesis completely. This occurs if the hands move close to each other but do not exhibit trajectory data that is characteristic for a mating activity. With the described approach, the discrimination between *Put* and $Screw^*$ activities was successfully accomplished in all of the 64 *Connect* activities in the test sequences.

Together with this postprocessing step, the results in table 5.1 indicate the possibility to recognize hand motions related to assembly construction actions based only on the sensory trajectory data. No additional information except for the skin-colored image regions is required and, consequently, most of the restrictions of the symbolic action detection (see Section 2.3.4) can be avoided. At the same time, the missing symbolic

information prevents the activity recognition from providing the complete details of the construction actions, i.e., what parts are manipulated and whether they are picked or placed. Therefore, the recognized activities do not allow to construct a hypotheses of the assembly that has been constructed by the observed manipulative gestures. Consequently, the applicability of the extracted information for improving human-machine interfaces is limited.

5.5 Summary

In this chapter, a visual activity recognition for the Situated Artificial Communicator domain was presented. Using Kalman filters to track all hand hypotheses generated by a skin color segmentation allows us to identify the hands out of several hypotheses by analyzing the trajectories. This makes our approach quite stable even in cluttered environments as the decision which regions represent the hands is made on the basis of the motions over time. Based on the hand trajectories, the recognition of construction activities is performed using a variation of the CONDENSATION algorithm. To provide detailed information about the type of *Connect* activity, a separate post-processing of the trajectory data is performed to discriminate between *Put* and *Screw* activities.

With the described activity recognition, the classification of the gestures used for assembly construction is achieved. However, the recognized activities do not allow to obtain a detailed hypotheses of the assembly constructed, as symbolic information indicating the manipulated parts is missing. We therefore propose in the next chapter our approach to an integrated action recognition that combines the data-driven activity recognition algorithm just presented with symbolic scene information to take advantage of the mutual information present in the two different types of information.

6 Integrating Sensory and Symbolic Information for Action Recognition

In the previous chapter, we have presented a data-driven approach using a priori defined trajectory models to perform gesture recognition. Following Bobick's categorization outlined in the introduction, the described data-driven approach realizes an activity recognition. The recognized activities are picking/placing of parts and connecting them together. Especially picking/placing is a very generic activity that is not limited to the baufix®construction scenario in the Situated Artificial Communicator domain but rather occurs in many scenarios.

However, the information that a generic 'pick/place' activity has been recognized is not very informative if there are many different objects that could be picked/placed. In such a situation, the symbolic information *what* object has been manipulated is crucial and needs to be incorporated into the recognition system to obtain a complete description of the manipulative gesture performed, i.e., to recognize the action instead of the activity. Additionally, only the use of both types of information, symbols and sensory data, allows us to detect contradictory results from the individual cues. For example, a hand motion may be recognized as 'picking/placing' based on the trajectory although there is currently no object in the scene that could be picked/placed. It is therefore necessary to integrate both types of information into a single action recognition framework to perform the recognition of manipulative gestures.

For avoiding recognition results contradictory to the scene context, the symbolic information about the context of an action, e.g., the objects currently in the scene, can be integrated with a data-driven trajectory recognition in two ways:

- The symbolic context is used to post-process the results from a data-driven recognition algorithm by filtering out 'impossible' results. This can be seen as a sequential approach.

- The symbolic context is modeled in the data-driven recognition algorithm in such a way that it influences the analysis of the trajectory data. This incorporation of symbolic constraints *during* the recognition of trajectory data resembles a parallel approach.

With the parallel integration of sensory data and symbolic constraints, generating results violating the scene context can be made very unlikely instead of completely ignoring 'impossible' results. This is an important advantage of a parallel approach as

one should avoid rejecting recognition results completely by relying on the results of one type of information. For example, if an object is present in the scene but the object recognition algorithm does not recognize it, the trajectory recognition algorithm should still be able to generate the recognition result 'picking' if this recognition result is the most likely result. Clearly, depending on how the recognition result is used in other algorithms, it may be necessary to verify the correctness of the result with additional processing steps.

For the implementation of an action recognition approach enabling the parallel integration of symbolic and sensory information, we have to develop a suitable framework to integrate both cues. A straightforward method is choosing one approach and extending it with the information from the other. Using a single type of information for gesture recognition has been demonstrated in Chapter 2 with a symbolic action detection, and in Chapter 5 with a data-driven activity recognition for assembly construction actions. Let us shortly get back to the drawbacks of both techniques to identify which one is better suited to provide the basis for an integrated approach.

In symbolic approaches (see Section 2.1), the symbol grounding step is a major difficulty as the extraction of symbols from vision data is challenging, and many domains contain a large number of symbols. An increasing number of symbols leads to an exponentially growing number of different symbolic interpretations for a scene, and prohibits the wide-spread use of symbolic methods for real-time gesture recognition.

Data-driven approaches are frequently used for recognizing gestures in real-time as they can be easily adapted to the available computational power. As these approaches usually lack the incorporation of symbolic constraints in a structured way, they are primarily applied to the recognition of communicative gestures within well-defined environments. Recognition of manipulative gestures is only accomplished by taking advantage of specific properties of the domain in consideration. This has been demonstrated in Chapter 5 for assembly construction actions. Obviously, relying on domain-specific symbolic boundary conditions to recognize manipulative gestures results in approaches that cannot be re-used for the recognition of gestures in other domains.

While the data-driven activity recognition approaches usually do not provide a generic method for representing symbolic context, they are much better for meeting real-time demands and contain intrinsically a reduced scalability problem by focusing only on the motions of the hand and not on the complete scene. Therefore, they are more suited to be used as basis for an integrated action recognition approach. In this chapter, we develop a novel framework for the recognition of actions that incorporates symbolic constraints in the particle filtering approach used for trajectory-based activity recognition. Based on the proposed extension, the actual scene context can be incorporated during the analysis of the trajectory data.

Before presenting our approach, we will review in Section 6.1 related literature dealing with the recognition of manipulative gestures. The basic idea for incorporating symbolic constraints in a particle filtering algorithm is presented in Section 6.2. The symbolic constraints considered for manipulative gestures are the objects currently in the scene.

The different ways in which objects are manipulated and an appropriate representation for the object context of a manipulative gesture are the topic of Section 6.3. The details of our extension to the standard particle filtering implementation for trajectory recognition are outlined in Section 6.4. A quantitative evaluation of the method based on construction actions in the Situated Artificial Communicator domain is presented in Section 6.5 and the evaluation in an office environment is described in Section 6.6. Using the framework to support task-based object recognition is sketched in Section 6.7 before the chapter concludes with a summary of the proposed method.

6.1 Related Work

One of the first approaches exploiting hand motions and objects in parallel is the work of Kuniyoshi [60] on qualitative recognition of assembly actions in a blocks world domain. As already pointed out in Section 2.1, this approach features an action model capturing the hand motion as well as an environment model representing the object context. The two models are related to each other by a hierarchical parallel automata that performs the action recognition. Kuniyoshi's approach represents essentially a symbol-based action recognizer as it does not apply a model for the motion of the hand, i.e., some kind of trajectory model. Instead, the hand motion is used only to guide the focus of attention of the visual feature detectors that recognize specific interactions between the hand and the objects.

An approach dealing with the recognition of actions in an office environment is the work by Ayers and Shah [3]. Here the head of a person is tracked based on detecting the face and/or neck with a simple skin color model. Before operation, an accurate description of the layout of the scene has to be provided to the system. Based on entrance and exit areas defined in image coordinates, tracking of the head is started and stopped. The position of the person is derived from the position of the head. The way in which a person interacts with an object is defined in terms of intensity changes within the object's image area. For this purpose, the image areas containing objects need to be defined before operation, too. By relating the tracked head to detected intensity changes in its vicinity and using a finite state model defining possible action sequences, the action recognition is performed. Similar to Kuniyoshi's approach, no explicit motion models are used.

In a similar approach, Peixoto et al. [77] use two cameras, a static camera and a binocular camera head, to recognize actions in an office environment. The static camera detects and tracks several people based on optical flow, i.e., motion in the image. The active camera head focuses on a single person to track its position. Based on the position of the person relative to 'context cells' defined a priori in world coordinates, the actions are recognized. Again, no explicit motion models are used and an action is simply the relation between some tracked person and a context cell.

An approach that actually combines both types of information, sensory trajectory data

and symbolic object data, in a structured framework is the work by Moore et al. [69]. In their object-centered approach called *ObjectSpaces*, a camera mounted on the ceiling observes a human interacting with different objects. The evaluated domains are an office, a kitchen, and a car. On initialization, a 'background image' showing the static scene is recorded and the image areas showing known articles are manually selected. During operation, background subtraction is used to detect new foreground regions. The motion and size of the regions differing from the background are analyzed over several frames to differentiate between new persons and new objects in the scene. Similar to the above described approaches, foreground regions belonging to new persons are analyzed with a simple skin color model to find the hands and track them subsequently.

The image features of a foreground region representing an unknown object are compared to the features of known objects to obtain *image-based* evidence for the type of the unknown object. The representation of known objects includes for each object the surrounding image area typically occupied by a hand during interaction (called *activity zone*) and the object manipulated previously. If a tracked hand comes close to an unknown object, the knowledge of the previously manipulated object and in what activity zones of known objects the hand is allows us to obtain *object-based* evidence for the unknown object type. Finally, the trajectory of a tracked hand is analyzed with Hidden-Markov-Models trained offline on different activities related to the known objects. Therefore the hand trajectory in the vicinity of a new foreground region provides *action-based* evidence for the type of the unknown object that is combined with image-based and object-based evidence to recognize objects.

While in the approach of Moore et al. the sensory trajectory information is used primarily as an additional cue for object recognition, we present in the following an approach for the oppositional goal of recognizing gestures with the help of symbolic information. Our person-centered approach to gesture recognition is motivated by the goal of extracting information about the acting human to improve the human-machine interface. Different from the object-centered approach of Moore et al., our approach supports the recognition of complex actions that involve a sequence of manipulative gestures interacting with different objects. For example, preparing a cup of coffee involves several gestures that interact with different objects. After the coffee pot is taken to fill a cup of coffee, sugar and possibly also milk are added to the cup of coffee.

6.2 Extending Particle Filtering with Context Information

The application of particle filtering (see Section 3.2.3) to the recognition of hand trajectories has already been demonstrated in Chapter 5 for recognizing the 'pick/place' and 'connect' activities in the Situated Artificial Communicator domain. Based on the particle filtering algorithm as it is used for the activity recognition, we will present in this

section the basic idea of our extension to this approach to enable the incorporation of context knowledge. The implementational aspects of this integrated action recognition approach are covered in the sections following.

Basically, in every iteration of a standard particle filtering algorithm, i.e., in each time step t, there are three steps performed:

1. **Select:** Selection of $N - M$ samples $s_{t-1}^{(i)}$ based on their weights $\pi_{t-1}^{(i)}$ and initialization of M new samples.

2. **Predict:** Prediction of the individual parameters contained in the selected samples $s_t^{(i)}$ based on some model of the system dynamics.

3. **Update:** Determination of the new sample weights $\pi_t^{(i)}$ based on the observations.

The sample vector $\mathbf{s}_t = (\nu_t, \mu_t, \phi_t, \alpha_t, \rho_t)$ as defined in Eq. 5.6 for the recognition of activities contains a reference to a specific activity model (ν and μ) as well as three parameters (ϕ, α, ρ) that relate the static trajectory data defined in the activity model to the actual trajectory data observed until time t. To integrate context knowledge into particle filtering, the definition of an activity model needs to be extended to incorporate symbolic information describing the context of a gesture.

Similar to the rules used for symbolic action detection (see Fig. 2.12 on page 31), a gesture can posses preconditions that must hold for its recognition and it can have an effect on the state of the scene. As a precondition of an action 'take cup', the hand would need to be empty and the effect of recognizing the action would be the hand holding the object 'cup'.

Besides this symbolic information applying to the start and end of a gesture, the trajectory model of the gesture can also posses symbolic context. The context associated with the time steps of the trajectory model is denoted in the following as *symbolic model context* C_T consisting of a context definition $\mathbf{c}_t$ for each time step in the trajectory model (see Eq. 6.1). For example, a model for the action 'take cup' would require the hand to approach an object 'cup' that must be present in the scene.

$$C_T = (\mathbf{c}_1, \ldots, \mathbf{c}_t, \ldots, \mathbf{c}_T) \tag{6.1}$$

Adding the precondition, the symbolic model context, and the effect to the activity model gives the definition of an *action model*:

$$M^{(\mu_k)} = (Precondition, X_T, C_T, Effect) \tag{6.2}$$

The precondition is used in the select step of the particle filtering algorithm to initialize and select only samples whose precondition matches the actual situation. This requires the application-dependent representation of the current situation. For example, the contents of the hand represent symbolic information about the current situation that is relevant for the recognition of manipulative gestures. On recognizing an action, the effect

defined in the action model of the recognized action changes the internal representation of the current situation.

The definition of an action model in Eq. 6.2 yields static information about the symbolic context of an action. However, similar to the parameters (ϕ, α, ρ) relating the trajectory model X_T to the observed motion data (see Fig. 5.3 on page 87), we need to extend the sample vector with information about the relation between the symbolic model context C_T and the observed symbolic data. Using the example of the action 'take cup', the generic model context 'cup' needs to be related to a specific object present in the scene. Especially in a scene containing several cups, the information which cup is manipulated by the gesture needs to be represented in the sample vector. Relating symbolic data of the scene to the symbolic model context is done by adding the *symbolic sample data* Ψ to the sample vector $\mathbf{s}_t$ giving:

$$\mathbf{s}_t = (\nu_t, \mu_t, \phi_t, \alpha_t, \rho_t, \Psi_t) \tag{6.3}$$

The initialization and adaptation of the symbolic sample data Ψ needs to be done based on the symbolic data extracted from the scene. If the symbolic information is time-dependent, it has to be changed accordingly in the prediction step. Using the extended sample vector shown in Eq. 6.3, the representation of symbolic information is added to the particle filtering algorithm. However, we have not yet changed the way in which the propagation is performed, i.e., the recognition is still unaffected by the symbolic information contained in the sample vector.

In the implementation of particle filtering for activity recognition, the weights for the sample set are determined in the update step based on the observed hand trajectories. To incorporate symbolic context information, the observed symbols have to be taken into account in the update step as well. For this purpose, the trajectory-based calculation of the sample weights is extended to also consider how well the symbolic context contained in the gesture models (denoted by μ and ν in the sample vector) matches the symbol-based observations. In this way, samples belonging to models that are consistent with the current scene context are chosen more often in the select step.

In our application, the value of the trajectory-based weight is directly related to the fit between model trajectory and observed trajectory. In contrast, a symbol-based weight is inherently non-proportional as it expresses how well the scene matches the expected symbolic context. To fuse these two different sources of information to obtain a single weight value, we extend the calculation of the sample weight from the observation data $\mathbf{z}$ given in Eq. 5.8 with a multiplicative *context factor* p_{symb} representing how well the observed *symbolic scene information* Θ_t fits the expected context:

$$\pi_t^{(i)} \propto p(\mathbf{z}_t|\mathbf{s}_t^{(i)})\; p_{symb}(\Theta_t|\mathbf{s}_t^{(i)}) \tag{6.4}$$

The proposed multiplicative fusion allows us to calculate the matching of the action model with the observed scene separately for the sensory and symbolic data. Inserting the sample vector definition from Eq. 6.3 into Eq. 6.4 and removing all parameters not

relevant for the calculation of the trajectory-based weight and the context factor leads to the weight update equation:

$$\pi_t^{(i)} \propto p(\mathbf{z}_t|\nu_t^{(i)}, \mu_t^{(i)}, \phi_t^{(i)}, \alpha_t^{(i)}, \rho_t^{(i)})\; p_{symb}(\Theta_t|\nu_t^{(i)}, C(\mu_t^{(i)}), \Psi_t^{(i)}) \tag{6.5}$$

For a successful incorporation of context knowledge into the particle filtering approach, it must be carefully analyzed what values the context factor may take on: A small factor may not increase the weights of the samples with correct context knowledge sufficiently to counteract the high weights of other samples whose model trajectory temporarily fits the observed trajectory better. A large factor leads to a loss of diversity in the sample set after several iterations as the samples that contain the correct context knowledge dominate the sample set and are primarily propagated.

Due to these consequences of small or large context factors, the range of values that is suitable has to be determined empirically based on the application domain. Before we turn to the technical implementation of the proposed extension to particle filtering, we will consider in more detail in the next section what kind of symbolic information can be applied to improve the recognition of manipulative hand gestures.

6.3 Extracting Context Information for Hand Gestures

The data-driven recognition of activities can be fused with a huge variety of symbolic information that is available in every environment. Obviously, not each kind of symbolic information is relevant to recognize a specific action that is made up of an activity in a certain symbolic context. In other words, the actions that have to be recognized determine what context information is needed. Our focus here is on the recognition of manipulative gestures as outlined in Chapter 1. The symbolic context information relevant for recognizing manipulative gestures are the objects that are being manipulated. For the sake of simplicity, we assume that a manipulative gesture interacts with a single object at a time. Before we turn to the definition of context knowledge and its extraction, we will briefly get back to manipulative gestures and take a closer look at how they manipulate objects.

6.3.1 Revisiting the Different Types of Gestures

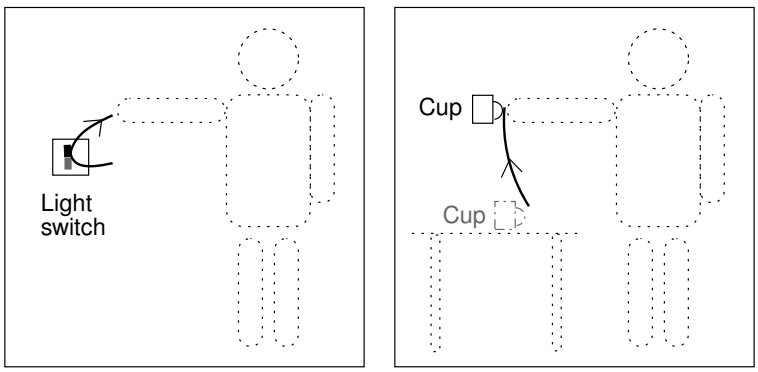
In Section 1.1 on page 3 we have introduced the classification of hand motions into communicative and manipulative gestures. Concentrating on the recognition of manipulative gestures, we pointed out that the recognition of the associated objects is challenging and that we therefore consider only those manipulative gestures that do not change the object's appearance substantially.

Intuitively, manipulative gestures operate with objects by translating or rotating them on a macroscopic scale like, for example, 'take cup' or 'pick up phone'. However, a large number of gestures performed during everyday life manipulate objects on a microscopic

scale. Especially the use of electric and electronic devices requires the user to press buttons or switches. While these devices are operated with manipulative gestures, the change applied to the devices is usually very small and often only temporarily. For example, pressing a button is only visible for a very short period of time. Obviously, the visual detection of the changed object appearance while the button is pressed down is very difficult and may be even impossible, e.g., if the button is totally occluded by the hand.

The scale of the object manipulation is therefore an important aspect for the visual detection of objects and their changes due to manipulative gestures. In order to capture the different nature of microscopic and macroscopic manipulative gestures, we define two subcategories:

- A *Touch Gesture (TG)* denotes a manipulative gesture that operates on the microscopic scale, i.e., the overall object is not moved (see Fig. 6.1(a)). However, a part of the object, i.e., a button or switch, may be temporarily or permanently moved or changed. Note that this formulation of a touch gesture does not imply that the changed object state can be detected visually.
- A *Move Gesture (MG)* is a manipulative gesture that actually translates or rotates an object, i.e., the object is moved on a macroscopic scale (see Fig. 6.1(b)).



(a) Touch Gesture (TG) (b) Move Gesture (MG)

Figure 6.1: The categorization of manipulative gestures into two subcategories for capturing the different nature of gestures manipulating objects either at (a) a microscopic or (b) a macroscopic scale .

The introduced two subcategories for manipulative gestures are relevant for the visual extraction of the symbolic information. For example, in a TG 'dialing numbers on a phone' the hand interacts with the static object 'phone' for dialing the numbers. Here the object 'phone' needs to be recognized just once, as it does not change its position during the gesture. In contrast, in an MG 'take cup' the object 'cup' changes its position

during the gesture. This change in position can be either a translation or a complete disappearance, depending on the quality of the object recognition algorithm and the way in which the hand holds/occludes the 'cup'.

In this view on manipulative gestures, the recognition of touch gestures is much easier as the object context can be extracted more reliably due to the objects remaining static. However, to recognize the numbers that were dialed while executing the above mentioned TG 'dialing numbers on a phone', the internal structure of the object 'phone', i.e., the positions of the individual phone keys, need to be known. Depending on the resolution of the camera images depicting the phone and the occlusions resulting from the hand dialing, the recognition of such small touch gestures is very challenging.

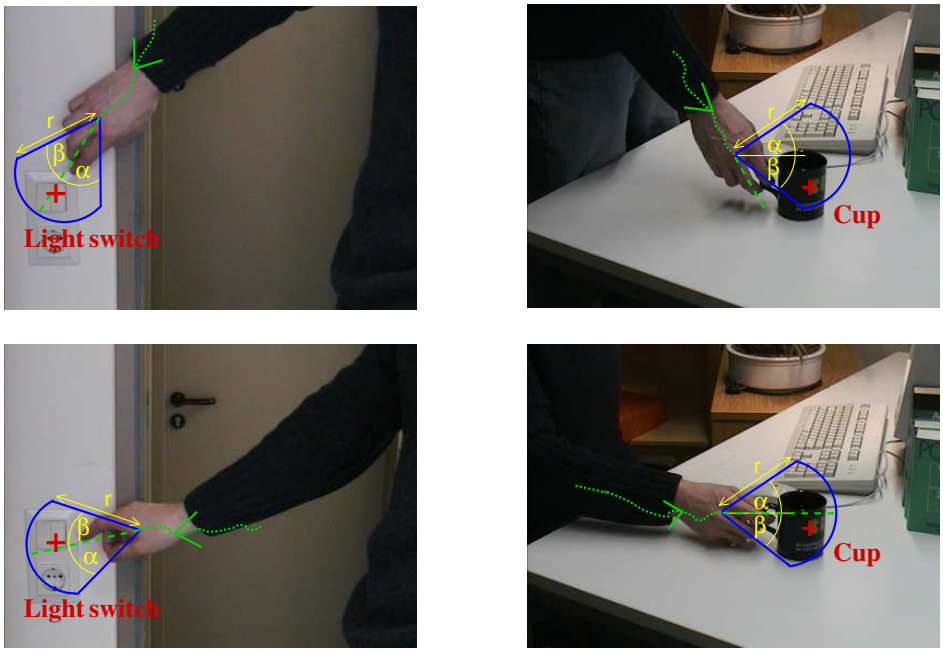
In this thesis we restrict our considerations to move gestures and to touch gestures that interact with objects without an internal structure. This allows us to concentrate on the action recognition aspects and avoids dealing with the problems of insufficient image resolution and occlusion of object details like, e.g., phone keys.

6.3.2 Relations between Hand Motions and Objects

Independent from the subcategory of the manipulative gesture, i.e., whether it is a touch gesture or a move gesture, the hand trajectory has to be related to the object that is manipulated. For this purpose, it is necessary to select those objects in the scene that can be manipulated by the currently observed gesture. Obviously, these must be objects that are close enough to the hand trajectory to be touched or picked for interaction. In limited environments this general formulation of object relevance by having a small distance to the hand trajectory may be sufficient [77, 3], but in more complex environments several objects will fulfill such a simple distance criterion.

In order to avoid the computational complexity as well as the increase in ambiguity associated with a larger number of relevant objects, it is advantageous to restrict the set of relevant objects by a more sophisticated criterion. For this purpose, we define a *context area* to be the image area that contains objects potentially relevant for a specific manipulative gesture. The context area is currently defined as a segment of a circle described by the radius c_r and the start angle c_α as well as the end angle c_β. For interaction with objects that do not have a specific 'handling direction' and can be approached from different directions, defining the *orientation* c_{orient} of the context area relative to the hand direction is appropriate. Most of the touch gestures fall into this category. Two examples for the context area of the touch gesture from Fig. 6.1(a) are given in Fig. 6.2(a). For objects that need to be approached from a specific direction like the cup in Fig. 6.1(b) the context area is independent of the hand direction and therefore has an absolute orientation (see Fig. 6.2(b)).

With the introduction of these two types of context areas it is now possible to define for each point of time along the trajectory of a gesture model the context area in which object context is expected. Defining the complete context including information about the expected object type is the focus of the next section.



(a) Relative orientation c_{orient} of the context area.

(b) Absolute orientation c_{orient} of the context area.

Figure 6.2: The different orientations for defining the context area. The hand trajectory (green) with its direction (dashed green) and the context area (blue) resulting from the parameters c_r, c_α, and c_β (yellow). The red plus marks the object position that is assumed to be detected by the object recognition.

6.3.3 Defining Object Context for Manipulative Gestures

With the definition of context areas given in the previous section, the image area that contains symbolic information relevant to the recognition of a specific gesture is known. Besides the definition of the context area for each point of time having a symbolic context, a specific gesture model also needs to specify what context is expected. Here we focus on objects as the type of the context.

First of all, symbolic context $\mathbf{c}_t$ for a time step of a gesture model μ can be *irrelevant*, *necessary*, or *optional*. Therefore the context contains a parameter c_{imp} representing the context importance. The context importance is essential as an object that is subject to a manipulative gesture may be temporarily occluded by the hand manipulating it. The ability to visually recognize the context with, for example, an object recognition algorithm is expressed by the context importance. For example, in the gesture 'take cup' the cup may be not detected by the object recognition as soon as the hand has taken it because it is now visually occluded. In this gesture the presence of the cup is a necessary context before the hand takes it, but it becomes optional afterwards to account for the potential visual occlusion due to the hand.

Besides defining the context area of a gesture, the *object type* c_{type} that is expected in this context area needs to be described. As an image can contain several identical objects, each object is given an identification number during recognition. This *ID* denotes a specific object while the type identifies what kind of object it is, e.g., a 'cup'. For defining the object context of a gesture model we only need the type but not the ID. However, as we will see in the next section, the ID of an object is needed to identify it during the course of a manipulative gesture. With the previously introduced context parameters describing a context area and the expected context object, the definition of the symbolic model context $\mathbf{c}_t$ for time t is:

$$\mathbf{c}_t = (c_{imp}, c_{orient}, c_\alpha, c_\beta, c_r, c_{type}) \tag{6.6}$$

To give an example, table 6.1 shows the complete symbolic context C_T for all eight time steps of the manipulative gesture 'take cup' manipulating the object of type 'cup'. In the depicted example, the first two time steps (t_1, t_2) do not contain any symbolic context information, i.e., only the trajectory data is relevant for the gesture recognition at these points of time. For the time steps $t_3 - t_5$, the model defines a necessary object context, i.e., the 'cup' must be present in an area oriented in an absolute coordinate system with the current hand position as reference point. After the hand has come close enough to the cup to pick it up, the context is from t_6 until the end optional, as the hand may occlude the cup during handling it.

	Context	Context area data				Context
	importance	orientation	start [deg]	end [deg]	radius	object type
Time	c_{imp}	c_{orient}	c_α	c_β	c_r	c_{type}
t_1	irrelevant					
t_2	irrelevant					
t_3	necessary	absolute	30	90	20	cup
t_4	necessary	absolute	30	90	20	cup
t_5	necessary	absolute	30	90	15	cup
t_6	optional	absolute	30	90	15	cup
t_7	optional	absolute	30	90	15	cup
t_8	optional	absolute	30	90	15	cup

Table 6.1: Example for the definition of object context for the manipulative gesture model 'take cup' consisting of a total of eight time steps.

Figure 6.3 shows images from a sequence depicting a hand taking a cup together with the context areas of the 'take cup' gesture model.

Through defining the symbolic context C_T for a gesture model as described above, we can use the symbolic information of the present scene to check at each point of time of a gesture model whether its expected symbolic data matches the actual contents of the context area. The algorithmic details of using the symbolic context information for action recognition are covered in the next section.

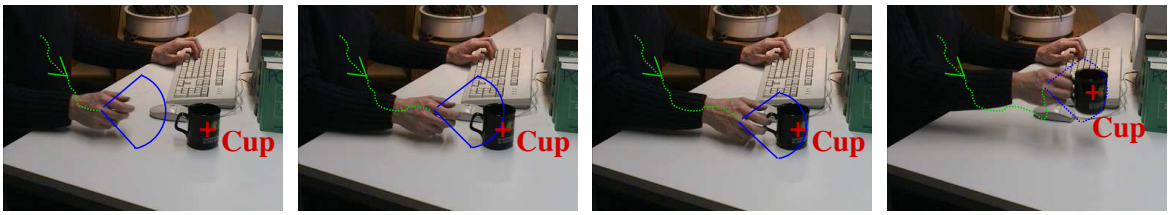


Figure 6.3: The context area of four time steps (t_3,t_4,t_5,t_6) of the 'take cup' gesture.

6.4 Integrated Action Recognition using Object Context

Given symbolic information about the current scene, the two additional steps necessary to perform integrated action recognition instead of trajectory-based activity recognition as outlined in Section 3.2.3 are the calculation of the context factor (see Eq. 6.4) and the update of the symbolic sample data. The principal realization of these two steps will be described in this section while the details of two example implementations are given in subsequent sections.

In the previous section we noted that the model context C_T of a gesture model contains for every time step an object type c_{type} that is manipulated by this gesture. The symbolic scene information Θ_t contains not only the type for each object O_j in the scene but also an additional ID that uniquely identifies this object:

$$\Theta_t = \{O_1, \ldots, O_j, \ldots, O_n\} \quad n \in N \tag{6.7}$$

$$O_j = (\text{type}_j, \text{ID}_j) \quad \forall\, j \in \{1, ..., n\} \tag{6.8}$$

To relate the context object type c_{type} to a specific context object O_j in Θ_t, the symbolic sample data $\Psi^{(i)}$ is used to store the ID of the context object. Initialization of the ID in $\Psi^{(i)}$ is performed the first time that an object context is expected by a gesture model and an appropriate scene object is found. Through this *binding* of the model to a specific object it can be assured that the manipulative gesture defined by this model interacts with the same object in the following time steps by requiring an object with this ID to be in the context area. If the end probability $P_{\text{end},t}(\mu)$ of this gesture model exceeds the recognition threshold D_{rec} (see Eq. 5.10,Eq. 5.11), it is recognized.

To determine the object manipulated by the recognized gesture, all samples in the sample set that were used for calculating $P_{\text{end},t}(\mu)$ are analyzed. For every object O_j, an *object probability* $P_{O,t}$ is calculated indicating the probability that this object has been manipulated by the recognized gesture. The binary binding between a sample and an object does not provide information about the quality of the binding, but the sample weight represents the information how good the overall sample matches the observed gesture. Therefore, the object probability $P_{O,t}$ is calculated for every object O_j based on the weights of the samples belonging to the gesture model μ and containing this object

in the symbolic sample data:

$$P_{O,t}(O_j,\mu) = \sum_{i=1}^{N} \begin{cases} \pi_t^{(i)} & \text{, if } O_j \in \Psi_t^{(i)} \ \wedge \ \mu \in \mathbf{s}_t^{(i)} \ \wedge \ \phi > 0.9\phi_{\max} \\ 0 & \text{, else} \end{cases} \qquad \forall\, j \in \{1,...,n\} \tag{6.9}$$

The object with the highest probability $P_{O,t}(O_j,\mu)$ is selected to be the manipulated object. This probabilistic formulation for determining the manipulated object allows us to recognize gestures that manipulate an object in the vicinity of other objects with the same type. While during binding of an object to a specific sample one of the other objects within the context area may be chosen, the final decision on what object was manipulated is based on the sample weights. These weights indicate which samples model the interaction between the observed trajectory and the binded object best.

Besides binding a scene object to the gesture model μ using the ID, Ψ can be used to represent information about the acting hand and the previously recognized actions. For example, after recognizing a 'take cup' action where the cup has been lifted to the mouth, the recognition of a 'drinking from a cup' action requires the knowledge that the hand holds a visually occluded cup. Additionally, knowledge about the action history, i.e., the previously recognized action 'take cup', allows us to restrict the set of actions that are considered for recognizing the next action. Equation 6.10 gives a generic description of the symbolic information contained in Ψ for recognizing manipulative gestures. Notice that except for the object ID the symbolic information represented by Ψ is application dependent.

$$\Psi_t^{(i)} = (\text{ID}, \text{state of the hand}, \text{history}, \dots) \tag{6.10}$$

In general, updating of the symbolic information is performed in every time step and depends on the currently observed objects in Θ_t, the current parent model ν, the symbolic model context C of the gesture model μ, and the previous symbolic information:

$$\Psi_t^{(i)} = f(\Theta_t, \nu_t^{(i)}, C(\mu_t^{(i)}), \Psi_{t-1}^{(i)}) \tag{6.11}$$

Now for each sample $\mathbf{s}_t^{(i)}$ the value of the context factor p_{symb} can be calculated from the symbolic scene data, the symbolic model context, and the updated sample-specific context:

$$p_{symb} = g(\Theta_t, \nu_t^{(i)}, C(\mu_t^{(i)}), \Psi_t^{(i)}) \tag{6.12}$$

As the symbolic data contained in $\Psi_t^{(i)}$ is application-dependent, the implementation of the weight value calculation depends on the application, too. In the most simple case, the symbolic context of a gesture is only the manipulated object and the symbolic sample data Ψ therefore contains only the ID. The calculation of the context factor p_{symb} for this simple context model is shown in pseudo-code in the algorithm on page 108. Here it is assumed that for initial binding of the object ID one object within the context area with correct type is randomly selected (lines 6-10). During normal operation the presence of the expected object within the context area and the context importance c_{imp} determine

1: **if** $c_{imp}(\mu) = Irrelevant$ **then**
2: $p_{symb} \Leftarrow F_{nocontext}$
3: $\Psi_t \Leftarrow \Psi_{t-1}$
4: **else** /⋆ model contains context ⋆/
5: $\Theta_{t,contextarea} \Leftarrow$ objects in context area with correct type
6: **if** $\Psi_{\mathrm{ID},t} \leq 0$ **then** /⋆ initial object binding if first context occurrence ⋆/
7: **if** $\Theta_{t,contextarea} \neq \emptyset$ **then**
8: $\Psi_{\mathrm{ID},t} \Leftarrow$ select randomly one object
9: **end if**
10: **else** /⋆ object already binded ⋆/
11: $\Psi_t \Leftarrow \Psi_{t-1}$
12: **end if**
13: **if** $\Psi_{\mathrm{ID},t} \in \Theta_{t,contextarea}$ **then** /⋆ object with correct ID in context area ⋆/
14: **if** $c_{imp}(\mu_t) = Necessary$ **then**
15: $p_{symb} \Leftarrow F_{necessary}$
16: **else** /⋆ context optional ⋆/
17: $p_{symb} \Leftarrow F_{optional}$
18: **end if**
19: **else** /⋆ object missing ⋆/
20: $p_{symb} \Leftarrow F_{missing}$
21: **end if**
22: **end if**

Algorithm 1: Schematic algorithm for setting the context factor p_{symb}

the value of the context factor (lines 14-22). In the depicted algorithm fixed weights F_* are used for the context factor p_{symb} based on the different context importances.

6.5 Evaluating the Recognition of Assembly Construction Actions

For the evaluation of the proposed action recognition approach, the trajectory data evaluated in Section 5.4 for the visual activity recognition is used. The symbolic scene information Θ_t for the construction scene is manually generated, i.e., all object recognition results are correct to focus on the performance of the action recognition algorithm. While in the activity recognition approach the gesture models contained the trajectory data for both hands, we here use two individual recognizers to analyze the actions of the two hands independently. This separate modeling and recognition is appropriate for most everyday manipulative gestures, as only a single hand is usually interacting with an object. In the construction actions in the Situated Artificial Communicator,

however, the two hands interact during connecting two parts together. To also recognize these cooperative manipulative gestures, the symbolic scene information Θ_t used in the recognition algorithms contains baufix® objects as well as the left (L) and right (R) hands:

$$\begin{aligned} \Theta_t &= \{O_1, \ldots, O_j, \ldots, O_n, \mathrm{L}, \mathrm{R}\} \quad n \in N && (6.13)\\ O_j &= (\mathrm{type}_j, \mathrm{ID}_j) \quad \forall\, j \in \{1, ..., n\} && (6.14)\\ \mathrm{type}_j &\in \{\text{round\_bolt, hexagon\_bolt, 3\_h\_bar, 5\_h\_bar, 7\_h\_bar,} && (6.15)\\ &\quad \text{felly, socket, ring, cube, rhomb\_nut}\} \qquad \forall\, j \in \{1, ..., n\} \end{aligned}$$

In this way, a *ConnectLR* action can be modeled as a trajectory performed by the left hand containing an object O_j and approaching the right hand R. Similarly, *ConnectRL* can be modeled and if both connect actions are recognized at approximately the same time, a *Connect* is recognized.

To keep track of the current contents of a hand, the symbolic sample data $\Psi_t^{(i)}$ for recognizing construction actions contains not only the ID of the object manipulated by the gesture model $\mu^{(i)}$ of this sample, but also the *hand contents* (HC) similar to one of the states in the two-hand model (see page 28):

$$\Psi_t^{(i)} = (\mathrm{ID}_t^{(i)}, \mathrm{HC}_t^{(i)}) \tag{6.16}$$

Updating the hand contents $\mathrm{HC}_t^{(i)}$ of an individual sample is performed on completion of the gesture model $\mu^{(i)}$ of this sample. Note that the completion of the child model of an individual sample i leads to updated hand contents only in the symbolic sample data $\Psi^{(i)}$ of this sample. The new child model assigned subsequently to this sample inherits the modified hand contents. Due to this local representation of the hand contents in each individual sample, several competing hypotheses for the actual hand contents can be present in the overall sample set.

For answering queries from other system components about the current hand contents and for initializing the hand context $\mathrm{HC}_t^{(i)}$ of new samples i, it is necessary to have a global representation of the hand contents. For this purpose, we introduce the global hand contents (GHC) that are, just like the sample-specific hand contents, similar to a state of the two-hand model. Only on a successful recognition of an action model, i.e., if the parent end probability $P_{\mathrm{end}}(\mu_k)$ (see Eq. 5.10) of the child model μ_k belonging to the action model ν exceeds a threshold, the GHC are updated with the effect (see Eq. 6.2) defined in the action model of the recognized action:

$$\mathrm{GHC}_t = Effect(\mu_k) \tag{6.17}$$

The set of recognized actions is essentially the same as for the activity recognition, i.e., picking and placing of objects and connecting them together. However, in the trajectory-based activity recognition it was not possible to differentiate between *Pick*

and *Place* as the trajectories of these two gestures are similar. With the global hand contents that are updated each time an action is recognized, the information whether a hand is currently empty or holds an object is available *while* analyzing the trajectory. This knowledge can be used in the integrated action recognition to separate between picking and placing. Following Eq. 6.2, the action models differ from the models used for activity recognition by containing additional symbolic information besides the trajectory model X_T. An action model can posses symbolic model context C_T associated with the trajectory model, in this case the context object type c_{type}, as well as a precondition P on the GHC and an effect E changing the GHC. For recognizing assembly constructions performed by the left hand, the following action models $\mathbf{M}^{(\mu)}$ are used:

$$
\begin{aligned}
\mathbf{M}^{(1)} &= (\text{P:GHC} = \emptyset,\ \textit{Hand down},\ c_{type} = \text{X},\ \text{E:GHC} = \text{X}) \\
\mathbf{M}^{(2)} &= (\text{P:GHC} = \text{X},\ \textit{Hand up}) \\
\mathbf{M}^{(3)} &= (\text{P:GHC} = \text{X},\ \textit{Hand down},\ c_{type} = \text{X},\ \text{E:GHC} = \emptyset) \\
\mathbf{M}^{(4)} &= (\text{P:GHC} = \emptyset,\ \textit{Hand up}) \\
\mathbf{M}^{(5)} &= (\text{P:GHC} = \text{X},\ \textit{Approach}^L,\ c_{type} = \text{R},\ \text{E:GHC} = \texttt{unknown}) \\
\mathbf{M}^{(6)} &= (\text{P:GHC} = \texttt{unknown},\ \textit{Move away}^L,\ c_{type} = \text{R})
\end{aligned}
$$

The action models for the right hand are identical except for $\mathbf{M}^{(5)}$ and $\mathbf{M}^{(6)}$ having different trajectory models and the left hand L as context object type. The *Hand down* trajectory model is contained in two different action models, model $\mathbf{M}^{(1)}$ with context object type c_{type}=X for picking a baufix® object X and model $\mathbf{M}^{(3)}$ without context for placing an object X. Similarly, two action models $\mathbf{M}^{(2)}$ and $\mathbf{M}^{(4)}$ contain the *Hand up* trajectory model for the associated hand motion following picking/placing. The action models $\mathbf{M}^{(5)}$ and $\mathbf{M}^{(6)}$ for connecting parts with the left hand have as model context not an object but the symbol denoting the right hand. Together with the precondition on the global hand contents, the action model $\mathbf{M}^{(5)}$ is only recognized if the left hand containing an object X approaches the right hand R. The context for the action model $\mathbf{M}^{(6)}$ defines a trajectory that points away from the right hand. The different trajectory models of the left hand together with the context areas are depicted in Fig. 6.4. Note that the models are defined by their velocities, for a better presentation these velocities were multiplied with a scaling factor leading to trajectories that are larger than the actual trajectories defined by the models. Therefore, the context areas are spread out over a larger image area as they are associated with the individual points of the scaled trajectories.

Based on these child models, the following three parent models $\mathbf{R}^{(\nu)}$ (see Eq. 5.3) are defined for the left hand:

$$
\begin{aligned}
\textit{Pick}(\text{X}):&\quad \mathbf{R}^{(1)} = \{\, \mathbf{M}^{(1)}, \mathbf{M}^{(2)} \} \\
\textit{Place}(\text{X}):&\quad \mathbf{R}^{(2)} = \{\, \mathbf{M}^{(3)}, \mathbf{M}^{(4)} \} \\
\textit{ConnectLR}(\text{X}):&\quad \mathbf{R}^{(3)} = \{\, \mathbf{M}^{(5)}, \mathbf{M}^{(6)} \}
\end{aligned}
$$

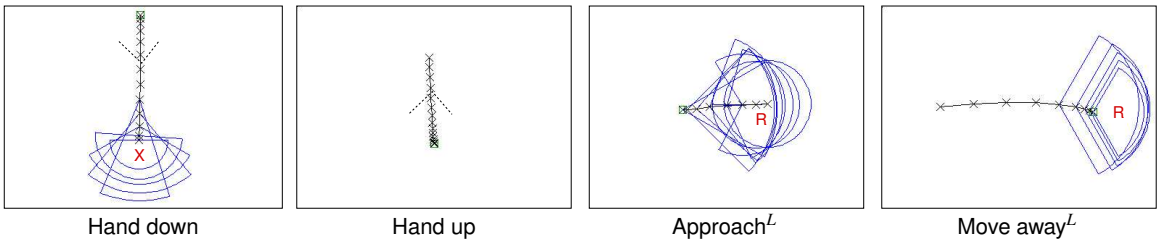


Figure 6.4: The child models of the left hand together with the context areas.

Except for different actions models $\mathbf{M}^{(5)}$ and $\mathbf{M}^{(6)}$, the parent models for the right hand are the same.

As recognizing the objects during the construction actions is very difficult, the state of the GHC after the *ConnectLR* action is unknown. One of the two hands, i.e., one of the two GHC's in the different action recognizers, holds the connected parts afterwards and the other one is empty. Different from the symbolic action detection in Section 2.3.3, we can recognize the *ConnectXX* action based on the visual observation of the hand motions. However, due to the visual occlusions it is not possible to extract the information what hand holds the (partial) assembly resulting from connecting two parts. Therefore, we can only update both GHC's if the next *Pick* or *Place* occurs. To accomplish this, we set each GHC to the state `unknown` after each hand executed the first child model of the *ConnectXX* action to allow initialization of both child models $\mathbf{M}^{(1)}$ and $\mathbf{M}^{(3)}$ for *Pick* and *Place*. If one of these two child models is recognized subsequently, the information whether an object was binded to the model, i.e., whether a *Pick* or *Place* is being performed, is used to update both GHC's accordingly (see also page 30).

For the evaluation of the integrated action recognition in the assembly construction domain, we used the empirically determined parameters $\mathrm{F}_{necessary} = 1.05$, $\mathrm{F}_{optional} = 1.00$, $\mathrm{F}_{missing} = 0.95$ for the context factor p_{symb}. All parameters were similar to the parameters used for activity recognition (see page 91) except for the size of the sample set. To account for the additional computation of the context factor, we used only 3000 samples instead of 3500. The recognition results for the proposed action recognition using the data set from the visual activity recognition (see Section 5.4) are shown in Table 6.2. To compare the results quantitatively, the right column depicts the results from the trajectory-based activity recognition for comparison.

The table indicates that the gesture recognition quality of the integrated approach incorporating symbolic context is comparable to the results obtained without the extended sample weight calculation (see page 92). More important, however, is the symbolic information associated with the recognized actions: the proposed approach provides the recognized actions with the objects that are manipulated, e.g., *Pick*(bolt).

	Total Actions	Recognized Actions		Recognized Activities
	#	#	%	% (see page 92)
Child models				
Hand down^{L}	57	52	91	89
Hand up^{L}	57	53	93	95
Hand down^{R}	87	82	94	90
Hand up^{R}	87	85	98	99
Approach$^{L\|R}$	72	67	93	89
Move away$^{L\|R}$	72	66	92	97
$\sum$	432	405	94	93
Parent models				
Pick^{L}	47	41	87	88
Place^{L}	10	9	90	
Pick^{R}	61	57	93	88
Place^{R}	26	24	92	
Connect	72	61	85	89
$\sum$	216	192	89	88

Table 6.2: Action recognition results obtained on the test set containing 36 gesture sequences depicting construction actions.

Comparison with Symbolic and Data-driven Recognition Approaches

Compared to the symbolic action detection, the combination of symbolic and sensory data for recognition avoids relying completely on changes in the symbolic data of the overall scene. While the integrated approach incorporates the actual trajectory of the hand for determining the manipulated object, the purely symbolic approach is based on the detection of new and disappeared objects anywhere in the observed scene for inferring the actions. Obviously, objects can only appear/disappear in the vicinity of an acting hand, but this information is not exploited in the symbolic action detection.

In contrast to the activity recognition, the integrated approach allows us to maintain symbolic information about the observed gestures. As noted above, a *Pick* action is recognized with the object that was picked, while the activity recognition did not provide this information. Therefore, only the use of the proposed action recognition in the assembly construction domain provides information about the constructed assembly structures similar to the symbolic action detection presented in Chapter 2. As will be shown in Chapter 7, this symbolic information is highly relevant for improving the human-machine interface of the Situated Artificial Communicator.

6.6 Recognizing Manipulative Gestures in an Office Environment

To demonstrate the applicability of the proposed framework for a wider range of manipulative gestures, we developed an action recognition system for an office environment containing the manipulative gestures 'pick up phone', 'hang up phone', 'take cup', 'stop drinking', 'pick book', 'stop reading', and 'type on keyboard'. Figure 6.5 depicts the setup of the office scenario that was used in the evaluation. In this setup, we concentrate on recognizing the above mentioned manipulative gestures performed with the right hand.

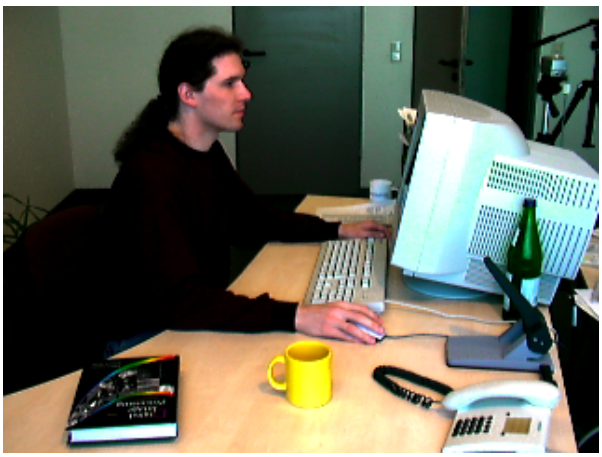


Figure 6.5: The scenario used for recognizing manipulative gestures performed in an office environment.

The symbolic scene information Θ_t for the office domain contains the objects that can be manipulated:

$$\Theta_t = \{O_1, \ldots, O_n,\} \qquad n \in N \tag{6.18}$$

$$O_j = (\text{type}_j, \text{ID}_j) \qquad \forall\, j \in \{1, ..., n\} \tag{6.19}$$

$$\text{type}_j \in \{\text{cup, phone, keyboard, mouse, book}\} \quad \forall\, j \in \{1, ..., n\} \tag{6.20}$$

Similar to the symbolic sample data used for recognizing construction actions, $\Psi_t^{(i)}$ contains the ID of the manipulated object and the current hand state (HS) that can be either empty or hold an object:

$$\Psi_t^{(i)} = (\text{ID}_t^{(i)}, \text{HS}_t^{(i)}) \quad with \quad \text{HS}_t^{(i)} = \{\emptyset | O_j\} \tag{6.21}$$

For the office domain, the precondition P and the effect E of the child models (see Eq. 6.2) operate on the global hand state (GHS) that is defined similar to the hand state in Eq. 6.21. The child models are defined as follows:

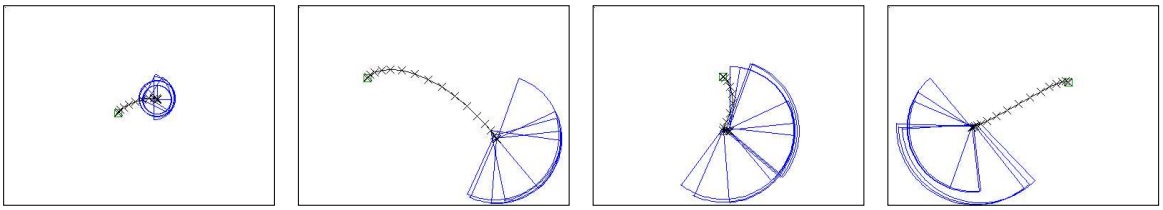
$$
\begin{aligned}
\mathbf{M}^{(1)} &= (\text{P:GHS} = \emptyset,\ \textit{reach left},\ c_{type} = \text{phone},\ \text{E:GHS} = \text{phone}) \\
\mathbf{M}^{(2)} &= (\text{P:GHS} = \text{phone},\ \textit{lift left}) \\
\mathbf{M}^{(3)} &= (\text{P:GHS} = \text{phone},\ \textit{place down left},\ \text{E:GHS} = \emptyset) \\
\mathbf{M}^{(4)} &= (\text{P:GHS} = \emptyset,\ \textit{back from left}) \\
\mathbf{M}^{(5)} &= (\text{P:GHS} = \emptyset,\ \textit{reach middle},\ c_{type} = \text{cup},\ \text{E:GHS} = \text{cup}) \\
\mathbf{M}^{(6)} &= (\text{P:GHS} = \text{cup},\ \textit{lift middle}) \\
\mathbf{M}^{(7)} &= (\text{P:GHS} = \text{cup},\ \textit{place down middle},\ \text{E:GHS} = \emptyset) \\
\mathbf{M}^{(8)} &= (\text{P:GHS} = \emptyset,\ \textit{back from middle}) \\
\mathbf{M}^{(9)} &= (\text{P:GHS} = \emptyset,\ \textit{reach right},\ c_{type} = \text{book},\ \text{E:GHS} = \text{book}) \\
\mathbf{M}^{(10)} &= (\text{P:GHS} = \text{book},\ \textit{pick right}) \\
\mathbf{M}^{(11)} &= (\text{P:GHS} = \text{book},\ \textit{place right},\ \text{E:GHS} = \emptyset) \\
\mathbf{M}^{(12)} &= (\text{P:GHS} = \emptyset,\ \textit{back from right}) \\
\mathbf{M}^{(13)} &= (\text{P:GHS} = \emptyset,\ \textit{reach front},\ c_{type} = \text{keyboard}) \\
\mathbf{M}^{(14)} &= (\text{P:GHS} = \emptyset,\ \textit{back from front},\ c_{type} = \text{keyboard})
\end{aligned}
$$

Based on these child models, the actions performed by a person sitting at an office desk are defined as parent models $R^{(\nu)}$ (see Eq. 5.3). Different from the previously used parent models, the transition probabilities $a_{ij}^{(\nu)}$ (see Eq. 5.4) for some of the parent models allow transitions in two different childs. Therefore, the transition probabilities that are not zero are given explicitly in the definition of the action models:

$$
\begin{aligned}
\textit{pick up phone}: &\quad \mathbf{R}^{(1)} = \{\, \mathbf{M}^{(1)}, \mathbf{M}^{(2)}, \mathbf{M}^{(6)}\}, &\quad & a_{12}^{(1)} = 0.5,\ a_{16}^{(1)} = 0.5 \\
\textit{hang up phone}: &\quad \mathbf{R}^{(2)} = \{\, \mathbf{M}^{(3)}, \mathbf{M}^{(4)}, \mathbf{M}^{(8)}\}, &\quad & a_{34}^{(2)} = 0.5,\ a_{38}^{(2)} = 0.5 \\
\textit{take cup}: &\quad \mathbf{R}^{(3)} = \{\, \mathbf{M}^{(5)}, \mathbf{M}^{(6)}, \mathbf{M}^{(2)}\} &\quad & a_{56}^{(3)} = 0.5,\ a_{52}^{(3)} = 0.5 \\
\textit{stop drinking}: &\quad \mathbf{R}^{(4)} = \{\, \mathbf{M}^{(7)}, \mathbf{M}^{(8)}, \mathbf{M}^{(4)}\} &\quad & a_{78}^{(4)} = 0.5,\ a_{74}^{(4)} = 0.5 \\
\textit{pick book}: &\quad \mathbf{R}^{(5)} = \{\, \mathbf{M}^{(9)}, \mathbf{M}^{(10)}\} &\quad & a_{9,10}^{(5)} = 1.0 \\
\textit{stop reading}: &\quad \mathbf{R}^{(6)} = \{\, \mathbf{M}^{(11)}, \mathbf{M}^{(12)}\} &\quad & a_{11,12}^{(6)} = 1.0 \\
\textit{type on keyboard}: &\quad \mathbf{R}^{(7)} = \{\, \mathbf{M}^{(13)}, \mathbf{M}^{(14)}\} &\quad & a_{13,14}^{(7)} = 1.0
\end{aligned}
$$

The trajectory models of several actions are depicted together with the context areas in Fig. 6.6. As the trajectory models are defined in terms of x- and y-velocities, the depicted trajectories have been obtained by multiplying the velocity values of all models with the same constant to scale the models for better visual presentation. These trajectories therefore do not match with the actual length of the trajectory resulting from performing such an action.

Note that, except for *type on keyboard*, the overall office actions consist of two manipulative gestures, one fetching and interacting with the object, and the second placing the object back on the table. For example, the overall 'drinking' action consists of *take cup* and *stop drinking*. Through recognizing the two gestures defining the start and end of the overall action individually, the information about the currently performed action can be used by other modules. Getting back to the example given in the introduction,



(a) *reach front* (b) *reach left* (c) *reach middle* (d) *reach right*

Figure 6.6: Several activity models together with the context areas for the actions recognized in the office environment.

the information that the human *is currently* drinking can be used to trigger his personal computer to read out loud new e-mails.

We have evaluated the recognition quality of the integrated system for the above given office actions with six test subjects. Each of the subjects performed every action ten times, leading to a total of 60 examples per action. The parameters were identical to the parameters used for the activity recognition (see page 91) using a sample set of 3000 samples to account for the calculation of the context factor. The results obtained are given in Table 6.3.

	Total Actions	Recognized Actions	
	#	#	%
take cup	60	58	97
stop drinking	60	53	88
pick up phone	60	59	98
hang up phone	60	58	97
pick book	60	60	100
stop reading	60	52	87
type on keyboard	60	52	87
$\sum$	420	392	93.3

Table 6.3: Recognition results for 60 gesture sequences depicting office behaviors.

The obtained results clearly demonstrate the suitability of the proposed integrated recognition approach for recognizing manipulative gestures in the office domain. The measured recognition quality has been obtained using a test set of six different people, demonstrating that the approach can cope with the differences in the execution of the gestures by different persons.

The three actions *take cup*, *pick up phone*, and *pick book* consisting of action models with context objects are recognized with a high recognition rate. A similarily high recognition rate was obtained for the *hang up phone* action that contains a very characteristic

trajectory model. The *type on keyboard* had a lower recognition rate than the preceding actions as the distance between the rest position of the hand and the keyboard was small resulting in a short trajectory model. Therefore, most recognition failures resulted from a recognition of the *back from front* child model while the person was still typing on the keyboard. A similar problem caused the lower recognition quality of the two move gestures *stop drinking* and *stop reading*. Both trajectory models are defined without a context object, because the hand already holds the object that is going to be placed. Therefore, the trajectories of these manipulative gestures compete with the other gesture models that have the same precondition. Both, holding a cup for drinking and holding a book for reading, are behaviors that are characterized by the hand performing some motions during interacting with the object, i.e., taking a small sip or flipping pages. The low recognition rates of *stop drinking* and *stop reading* were mostly due to an too early recognition of these actions while the person was still interacting with the object.

If the motions that are performed during using the object would have characteristic trajectories, the recognition quality could be improved by adding appropriate action models, for example, a 'flipping page' action. However, experiments revealed that the differences between persons in the way they handle objects are quite large prohibiting the construction of generic models. Additionally, the small scale of the object interactions, i.e., how the pages are flipped or the hand holds the cup, does not result in characteristic trajectories of a good quality. Consequently, interactions with office objects occuring at a small scale like, e.g., 'flipping pages' cannot be modeled appropriately with trajectory models. To cope with such actions more successfully, an object recognition algorithm would be needed that is capable of recognizing the object while the human is interacting with it.

6.7 Task-based Object Recognition

In the description of the integrated action recognition given above, we assume the hand trajectory data $\mathbf{z}$ and the symbolic scene information Θ_t to be available for processing. In describing our recognition approach, the term 'integrated' denotes the parallel use of symbolic and sensory data for analyzing the observed manipulative gestures. The extraction of the symbolic object information necessary for recognizing manipulative gestures is not covered in this thesis, but in this section we will take a closer look at the restrictions that the proposed action recognition framework provides to simplify the object recognition task.

For the symbolic action detection we noted on page 15 that the extraction of symbols for arbitrary scenes is computationally very difficult due to the potentially large number of objects and similar appearances of different objects. It is therefore advantageous to perform a *task-based object recognition* that aims at recognizing the objects that are needed in the current task.

Other approaches to an integrated action recognition outlined in Section 6.1 use sim-

ple visual feature detectors to extract symbolic information about the scene. In these approaches, the trajectory of the hand or head indicates where to perform the visual feature extraction. The visual routines are either identical for all objects [60] or are directly related to the positions of the interesting image areas defined a priori [3, 77]. Obviously, the small sets of actions chosen for these approaches help to avoid the scalability problem.

Similar to these approaches, we could use the trajectory data $\mathbf{z}$ to perform object recognition in the vicinity of the hand. However, this would require the object recognition to search in the image area for any object out of the complete set of objects modeled in the action recognition. Only if objects can be related to specific areas in the image similar to [3, 77], the problems of scalability and appearance ambiguity can be avoided. However, in many domains actions can not be related to specific scene areas. For example, nearly all objects in an office that are manipulated by gestures can be positioned anywhere on a table. It is therefore necessary to have a more generic definition of the restrictions following from the task context. The context areas used in the proposed action recognition framework for deriving symbolic context information could form the basis for such a task-based object recognition approach. Different from analyzing the complete area in the vicinity of the hand, the context area enables the analysis of an even smaller image area based on the associated action.

For an individual sample $\mathbf{s}$ of the sample set, the model context $\mathbf{c}_\phi(\mu)$ at the model position ϕ defines the context area and the type of the expected object. In this local view on the action recognition approach, a task-based object recognition would consist of searching only for the expected object type c_{type} in the context area. This local approach would solve the ambiguity problem as only the presence of the expected object is verified. But looking at the sample set containing typically several thousand samples reveals that performing local task-based object recognition is usually not adequate for real-time action recognition as it requires to execute an object recognition algorithm for every individual sample.

Instead, we propose to perform global task-based object recognition using *unified context areas* that are constructed by merging context areas of individual samples. Merging can be done either object-specific or classifier-specific: If objects need specific classifiers, an individual unified context area is constructed for each object. For object recognition approaches that are capable of extracting several object labels at once, a classifier-specific unified context area associated with all the objects known by this classifier can be constructed.

The advantages of the suggested task-based object recognition using unified context areas are the reduction of the analyzed image area and the possibility to perform object-specific analysis of small image areas. It should be noted, however, that applying different classifiers may lead to contradictory object recognition results for a single image area. This is an important aspect that needs to be addressed in the implementation of a task-based object recognition on the basis of unified context areas that applies several different object classifiers.

6.8 Summary

In this section the integrated action recognition approach to recognize manipulative gestures was introduced. The extension to the particle filtering algorithm as used for activity recognition consists of an additional context factor for the sample weight calculation and the addition of symbolic context information to the sample representation. For manipulative gestures, we presented a representation of the context in terms of the object type that is manipulated and its position with respect to the acting hand. The application of this object context for calculating the context factor to update the sample weights was described. An evaluation of the proposed action recognition approach was presented using the same test set of assembly construction actions that was evaluated for activity recognition. As a second domain containing a wider range of manipulative gestures, the recognition of actions in an office environment was demonstrated.

The proposed approach provides a 'complete' recognition of manipulative gestures including *which* objects were manipulated. In contrast, the activity recognition recognizes gestures based only on the trajectory data and relies on assumptions about the domain to match recognized activities to actions performed with objects. The proposed framework can be used to perform task-based object recognition for the extraction of the symbolic information from the scene by taking advantage of the defined context areas for searching context objects.

7 Action Recognition in the Situated Artificial Communicator

In this chapter, we will demonstrate that the recognition of manipulative gestures can substantially improve a human-machine interface. For this purpose, we have integrated the action recognition within the assembly construction domain of the Situated Artificial Communicator described in Section 2.2. The system components realize the perceptual front-end of the Situated Artificial Communicator to allow the human to interact with the system through using speech and gestures. However, no robotic manipulator is present in the perceptual front-end for the automatic construction of the toy assemblies based on the user's instructions. Therefore, in our setup a human constructor executes the instructions that have been given by the instructor. The manipulative gestures performed by the human constructor are recognized with the action recognition framework presented in Section 6.5.

In the following, we will demonstrate how the results from recognizing the construction actions can be used to improve the capabilities of the overall system. First, we will show in Section 7.1 how the observation of the acting human can be used for improving dialog interaction by providing information about the hand contents. Besides this direct improvement of the human-machine interface through the action recognition results, the recognized actions can also be used as context knowledge for other system components to indirectly improve the human-machine interface. In the Situated Artificial Communicator, the observation of the construction actions provides relevant context information for the vision algorithms aiming at recognizing the baufix® objects and assemblies. How the action recognition results are combined with these algorithms is demonstrated in Section 7.2 for the recognition of assemblies and in Section 7.3 for the recognition of objects. The realized applications presented in this chapter for improving the human-machine interface of the Situated Artificial Communicator are summarized in Section 7.4.

7.1 Improving the Human-Machine Interface

The architecture of the perceptual front-end of the Situated Artificial Communicator is shown in Fig. 7.1. The components of the action recognition are drawn in black and its interactions with other modules to improve the human-machine interface are drawn in blue. All other components not relevant at this point are drawn in gray. In the following sections, other versions of this figure will be used to depict the interaction

between action recognition and object/assembly recognition. Figure 7.1 contains on the left side the modules that are responsible for processing the speech input from the user instructing the system. On the right, the components related to the image processing tasks are depicted. The central module relating the spoken input to the visually observed scene is drawn in the middle and denoted 'interrelated perceptions'.

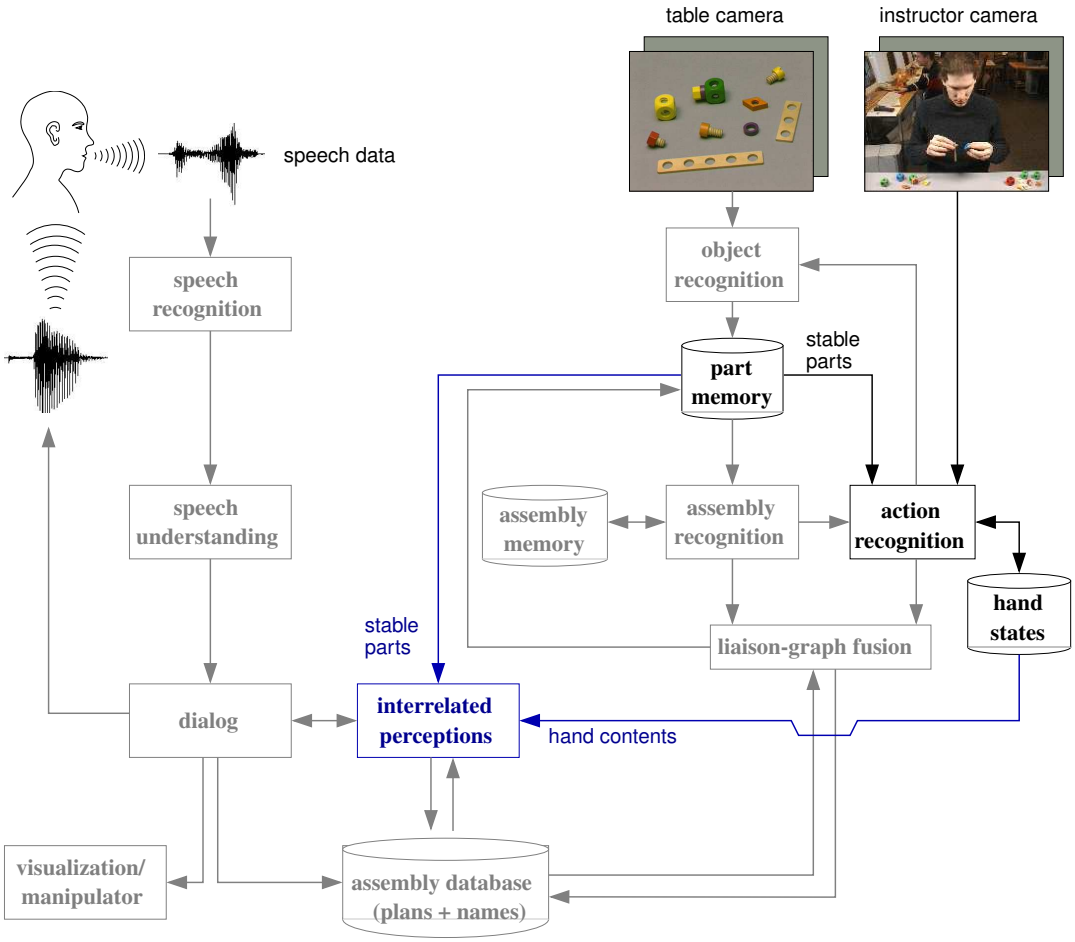


Figure 7.1: The basic system architecture of the Situated Artificial Communicator with the action recognition components (black) and the module interactions for improving the human-machine interface (blue).

Given an acoustic signal, the speech recognition extracts partial syntactic structures that are transferred to the speech understanding component. Based on the syntactic structures, a linguistic interpretation is generated that is passed to the dialog component. For all instructions referring to parts, e.g., 'Put the bar on the red bolt', the dialog queries the component interrelating the vision data with the spoken input.

The parts currently laying on the table are provided to the interrelation module by the part memory introduced in Section 2.3.2. The part memory is updated with information

from the object and assembly recognition procedures each time a new image has been processed. Instructions that refer to parts that are already in the hands of the human constructor cannot be resolved by the interrelation module using only the information from the part memory. For example, if the red bolt has already been taken, it is no longer on the table and therefore it is not contained in the part memory. Through analyzing the images captured by an additional camera observing the human constructor, the construction actions carried out can be recognized with the proposed action recognition system and the hand states can be updated accordingly. Similar to the information from the part memory about the current table scene, the information from the hand states about the current hand contents is provided to the interrelation module for resolving instructions. In this way, the interrelation module is able to relate the red bolt from the instruction to the part contained in the hand.

If an object or assembly matching the description is found either in the information from the part memory or the hand states, it is returned to the dialog. If all parts contained in the instruction have been related to parts on the table or in the hands, the dialog generates an appropriate system output. In the realized system this amounts to visualizing the intended action in a virtual environment [7] as can be also seen in Fig. 2.13 on page 32. If the dialog cannot completely interpret an instruction, it asks the user for more information.

Besides referencing elementary baufix® parts, it is also possible to assign names to assemblies. In the Situated Artificial Communicator domain, these names usually denote a toy-airplane and its components. Naming assemblies can be done either implicitly ('Take the bolt in front of the airplane') or explicitly ('This is the airplane'). The names given to assemblies are stored together with the structural descriptions from assembly recognition in the assembly database. From now on, references to parts on the table containing names are resolved using the assembly database, e.g., 'Take the airplane'.

Observing the construction actions leading to an assembly results in the assembly construction plan. If a human is intended to learn how to construct an assembly whose construction plan is contained in the assembly database, the action recognition can be used for comparing the construction actions executed in the scene with the learned construction plan [29]. In such a setup, the dialog can provide feedback about whether the performed actions fit the given construction plan, i.e., whether a human constructor constructs the assembly correctly.

In another setup used in the Situated Artificial Communicator project, several of the above mentioned modules of the perceptual front-end are integrated with a robotic manipulator [66]. In this setup, the dialog sends the interpreted instructions directly to the robot for execution.

In the perceptual front-end of the Situated Artificial Communicator described above, the interaction quality is *directly* improved through using the results from action recognition, i.e., the contents of the hand states, for resolving instructions. In the next two sections we will present the integration of action recognition results into the algorithms for observing the assemblies and elementary parts contained in the visual scene.

Through enabling the extended vision algorithms to provide better scene recognition results, the action recognition serves in the following to improve the overall interaction quality *indirectly*.

7.2 Fusing with Visual Assembly Recognition

An important functionality needed in the Situated Artificial Communicator is the ability to recognize assemblies made of elementary baufix®-parts. In Section 2.2.3 we introduced the syntactic assembly recognition based on the functional model for assemblies. Based on the results of an object recognition algorithm, the details of the connections can be determined by visually analyzing the relations between the individual objects. However, the information about the objects in the assembly can be incomplete due to perspective occlusions. For example, some small parts may have been visually occluded and therefore have been missed by the object recognition algorithm.

An alternative method to recognize an assembly follows from the fact that constructing a complex part from a set of separate parts usually takes a *sequence* of construction actions, i.e., in most cases constructing an assembly is a *process* of *sequentially* connecting several components. This assembly construction process can be observed with the action recognition approach proposed in this thesis. Recognizing the individual construction actions provides the construction sequence necessary to build the assembly, i.e., the construction plan. This construction plan is quite reliable regarding the types of parts used and their order in the constructed assembly, but no details of the connections are available. This follows from the fact that the action recognition observes only the motions of the hands, but it is not possible to infer which holes or threads are used to establish the connection between two parts. Only by recognizing the individual parts and their orientation in space one could infer which holes or threads are used for a connection, but this would require the object recognition algorithms to cope with the visual occlusions due to the hands holding the parts.

Comparing the two different methods to recognize assembly structures based on the visual structure (syntactic assembly recognition) and the process (action recognition) shows that they provide complementary results. The syntactic assembly recognition provides detailed information about the connections between the recognized individual parts, e.g., the hole of a bar used for connecting it with a bolt and a cube. While there may be some parts missing due to visual occlusion, these parts are contained in the structure generated by the action recognition. However, the connection details are not provided by action recognition as this detailed information is not inferable with current vision algorithms.

Fusing the results allows to take advantage of the benefits of the two distinct approaches and can greatly enhance the quality of the extracted assembly structure. Figure 7.2 depicts how the different modules are connected with each other to implement the merging of the different results. The principal processing steps for fusing assembly

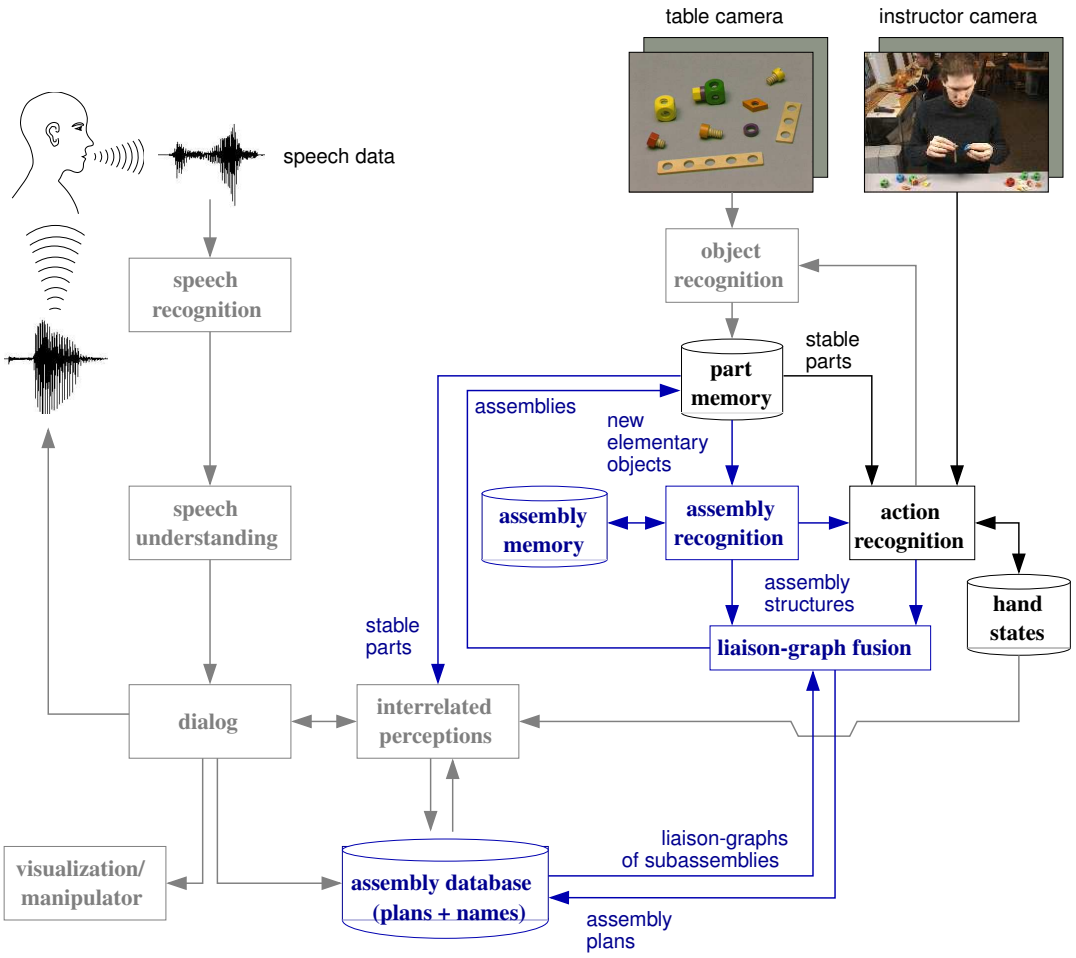


Figure 7.2: Incorporating action recognition results to improve the assembly structures generated by the syntactic assembly recognition.

structures are as follows:
If an assembly is put down on the table, several new objects are detected by the object recognition module. On receiving several new objects at a time, the part memory sends the set of new objects to the assembly recognition. As the syntactic assembly recognition can become very time-consuming if the number of objects is large, the recognition process is only performed if no visually similar assembly has been recognized previously. To enable the search for known assemblies, a hierarchically-organized assembly memory containing different visual representations of all previously recognized assemblies has been developed together with C. Bauckhage [10]. Before the syntactic assembly recognition is carried out, this assembly memory is searched for a match between the stored assemblies and the visual appearance of the new set of objects (for details see [10, 70]). If a match is found, the stored assembly structure is taken as recognition result, otherwise the syntactical assembly recognition is started. If an assembly has been found, the

structural description of the new assembly is sent to the fusion module. At the same time, the complete set of new objects as received from the part memory is sent to the action recognition.

The action recognition compares the set of new objects with the contents of the hand states. If either one of the hand states contains a complete assembly or the parts contained in both hand states can be connected to form a complete assembly, these parts are compared to the set of new objects. Given perfect object recognition results, the parts in the hand states would exactly match the objects recognized in the scene. However, as visual occlusions cause object recognition errors or invisible objects not recognized at all, the (partial) assembly contained in the hand state(s) is assumed to have been put down in the scene if at least 75% of the objects from object recognition are matched in the assembly contained in the hand state(s). In this case, the action recognition sends the assembly structure to the fusion module. If only an incomplete assembly is contained in one state of the two-hand model, a *Connect* action is inferred before sending the assembly structure to the fusion module.

As the recursive assembly model introduced in Section 2.2.1 does not provide unique descriptions of assemblies, it is advantageous for the fusion to first transform an assembly structure into a unique representation. In this way, the comparison in the fusion module between the different assembly structures from action recognition and syntactic assembly recognition can be simplified. One method to represent assembled objects with a unique representation is the *liaison-graph*, a concept introduced by Bourjault [15]. We will first describe in the following the liaison-graph representation before going into the details of the fusion process.

Liaison-graph Representation

For describing the structure of an assembly, the nodes of a liaison-graph represent the parts of an assembly while the edges correspond to certain relations – generally physical contacts – between the assembled objects. Here the liaison-graph is used to represent *visible physical contacts* between the objects of an assembly. The direction of the edges indicates the order in which the objects were put onto the corresponding bolts. Different types of edges are used to determine which object is put onto which bolt.

For the example assembly shown on the left of Fig. 7.3, one structural description based on the functional assembly model is depicted in the middle of Fig. 7.3 and the liaison-graph describing the visible physical contacts is shown on the right of Fig. 7.3.

As pointed out above, the syntactic assembly recognition provides detailed information about the connections between the individual parts, e.g., the hole of a bar used for connecting it with a bolt and a cube. As this information is not provided by the action recognition algorithm and is therefore not used in the fusion process, it is not drawn in the depicted assembly representations for simplicity.

The textual representation for the liaison-graph depicted in Fig. 7.3 can be written as follows:

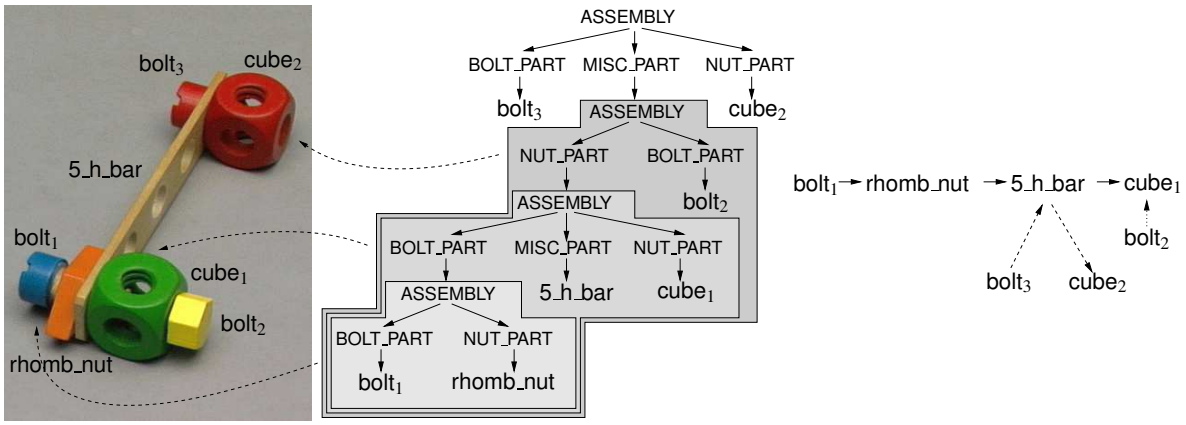


Figure 7.3: An example assembly consisting of several subassemblies with a corresponding structural description derived on the basis of the functional assembly model and the liaison-graph representation.

$bolt_1$ - rhomb_nut - 5_h_bar - $cube_1$
$bolt_2$ - $cube_1$
$bolt_3$ - 5_h_bar - $cube_2$

Transforming a structural description of an assembly into a liaison-graph representation is done as follows: Obviously, all objects of a certain level of hierarchy in the structural description are put onto the same bolt. Therefore, all hierarchical levels of the structural description have to be parsed and subassemblies need to be replaced by their functional parts to obtain the liaison-graphs containing all parts with physical contacts. While MISC_PART and NUT_PART subassemblies must be replaced with the respective part establishing the function, a BOLT_PART subassembly has to be replaced with the complete set of parts from the lower level of the hierarchy, as all parts of a BOLT_PART subassembly have a physical contact with the bolt.

Fusing Different Liaison-graphs

In order to merge the distinct assembly structures from action recognition and syntactic assembly recognition, we use the textual representation of the liaison-graph. In the fusion module the liaison-graphs from assembly recognition and action detection are compared to construct a merged liaison-graph. The merged liaison-graph is obtained by enhancing the liaison-graph from assembly recognition based on the additional parts in the liaison-graph from the action detection. The fusion process consists of three steps:

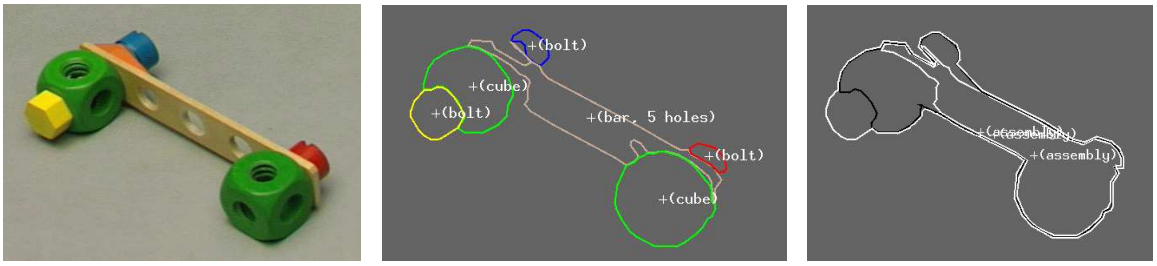
1. Completion of the liaison-graph from action detection by retrieving subassemblies from the assembly database. If assemblies instead of elementary baufix® parts are used for construction actions, the internal structures of the subassemblies have

to be retrieved from the assembly database. These subassemblies have been constructed previously and therefore the retrieved liaison-graphs are the result of a previous fusion step.

2. Identification of liaison-graph lines from assembly recognition that match with previously fused subassemblies retrieved from the assembly database.

3. Enhancement of the remaining liaison-graph lines from assembly recognition based on the liaison-graph lines from action recognition if missing parts are small and therefore may have been not recognized due to visual occlusion.

To give an example for the fusion process, consider the assembly depicted in Fig. 7.4(a). The resulting object recognition results are shown in Fig. 7.4(b). Due to the perspective, the rhomb_nut was not recognized and consequently it is not contained in the recognized assembly structure visible in Fig. 7.4(c).



(a) (b) (c)

Figure 7.4: (a) An example assembly with a rhomb_nut partially occluded due to the perspective; (b) The results from object recognition lacking the rhomb_nut; (c) Visual representation of the results from assembly recognition.

Therefore, the rhomb_nut is also missing in the liaison-graph generated from the assembly structure:

(A1) bolt_a - 5_h_bar - cube_a
(A2) bolt_b - 5_h_bar - cube_b
(A3) bolt_c - cube_b

The following shows the liaison-graph from action detection ($B1$) and the subassemblies retrieved from the assembly database ($B2 - B4$):

(B1) $\text{ASSEMBLY}_4 \rightarrow \text{ASSEMBLY}_1$ - ASSEMBLY_2 - ASSEMBLY_3
(B2) $\text{ASSEMBLY}_1 \rightarrow \text{bolt}_1$ - rhomb_nut
(B3) $\text{ASSEMBLY}_2 \rightarrow \text{bolt}_2$ - 5_h_bar - cube_1
(B4) $\text{ASSEMBLY}_3 \rightarrow \text{bolt}_3$ - cube_2

Comparing the two graphs given above shows that line A3 is identical to the ASSEMBLY_3 in line B4 and therefore bolt_c=bolt_3 and cube_b=cube_2. Now the remaining liaison-graph lines are:

(A1) bolt_a - 5_h_bar - cube_a
(A2) bolt_b - 5_h_bar - cube_2
(B1) $\text{ASSEMBLY}_4 \rightarrow \text{ASSEMBLY}_1$ - ASSEMBLY_2 - ASSEMBLY_3
(B2) $\text{ASSEMBLY}_1 \rightarrow \text{bolt}_1$ - rhomb_nut
(B3) $\text{ASSEMBLY}_2 \rightarrow \text{bolt}_2$ - 5_h_bar - cube_1

ASSEMBLY_2 in line B3 contains cube_1 and therefore only matches with line A1 giving bolt_a=bolt_2 and cube_a=cube_1. and resulting in the remaining liaison-graph lines:

(A2) bolt_b - 5_h_bar - cube_2
(B1) $\text{ASSEMBLY}_4 \rightarrow \text{ASSEMBLY}_1$ - ASSEMBLY_2 - ASSEMBLY_3
(B2) $\text{ASSEMBLY}_1 \rightarrow \text{bolt}_1$ - rhomb_nut

To fuse the remaining line A2 with the lines B1 and B2, the subassemblies in line B1 are replaced by the parts used for the connection. This is accomplished by comparing B1 with A2 and taking into account the knowledge which function a subassembly can have depending on its position as outlined on page 125. This leads to the remaining liaison-graphs:

(A2) bolt_b - 5_h_bar - cube_b
(B1) bolt_1 - rhomb_nut - 5_h_bar - cube_2

For comparison of the remaining liaison-graphs, a heuristic is applied to deal with small parts missing in the liaison-graph from assembly recognition due to perspective occlusions. If a small part is contained in the liaison-graph from action recognition and is missing in the result from assembly recognition, it is added to the fused liaison-graph. Using this heuristic, the rhomb_nut is inserted in line A2 giving the complete liaison-graph for the assembly in Fig. 7.4(a):

(A1) bolt_a - 5_h_bar - cube_a
(A2*) bolt_b - rhomb_nut - 5_h_bar - cube_b
(A3) bolt_c - cube_b

While this graph depicts only the fused part sequence, the connection details generated by the assembly recognition (e.g., which hole of 5_h_bar in line A1 was used in the actual construction step) are attached to the individual parts. This fused liaison-graph is now stored in the assembly database to be used in subsequent processing steps.

7.3 Providing Hypotheses for Object Recognition

The previous section has shown how problems in assembly recognition due to visual occlusion can be reduced by incorporating process knowledge to hypothesize occluded objects within an assembly. The drawback of this fusion approach is the fact that it comes into play *after* the assembly recognition has been carried out. Therefore, this method works well only for small objects that are of the MISC_PART type and do not prevent the overall assembly structure from being recognized. If the elementary object recognition fails to recognize a cube, for example, the assembly structure cannot be detected and therefore the assembly fusion process described in the previous section fails.

One obvious solution to this problem is the incorporation of the process knowledge earlier in the recognition phase. From supervising the process of taking parts, connecting them, and placing constructed (sub)assemblies back in the scene, the set of parts used for building an assembly is available. This context knowledge can be exploited by the recognition procedure as it provides expectations about the objects in the scene if several new objects appear simultaneously, i.e., if an assembly has been put down. In the following, we will briefly sketch the integration of action recognition results into the baufix® object recognition system described in Section 2.2.2. Based on an example assembly, the influence of the context knowledge will be highlighted. More details on the integration of the context knowledge into the object recognition system can be found in [18].

Integrating Context Knowledge

The integration of the object recognition algorithm into the Situated Artificial Communicator architecture to incorporate context knowledge is shown in Fig. 7.5. If a newly constructed assembly is placed in the scene, the 'place' action is recognized by the action recognition. In this case, it retrieves all objects contained in the assembly from the hand states and provides these object hypotheses to the object recognition. If the object recognition detects several new elements in the scene simultaneously, it assumes that an assembly has been placed in the scene and incorporates the expectations from action recognition into the recognition process. Comparing the competing results of the object recognition system with the expectations influences the voting process and the judgment for selecting the best scene interpretation.

Example Results

To demonstrate the integration of process knowledge into the recognition framework, we use the example assembly shown in Fig. 7.6(d). We assume that the assembly was constructed starting with the construction of ASSEMBLY_1 consisting of the yellow bolt connected to the green cube (see Fig. 7.6(a)). This assembly was placed back in the

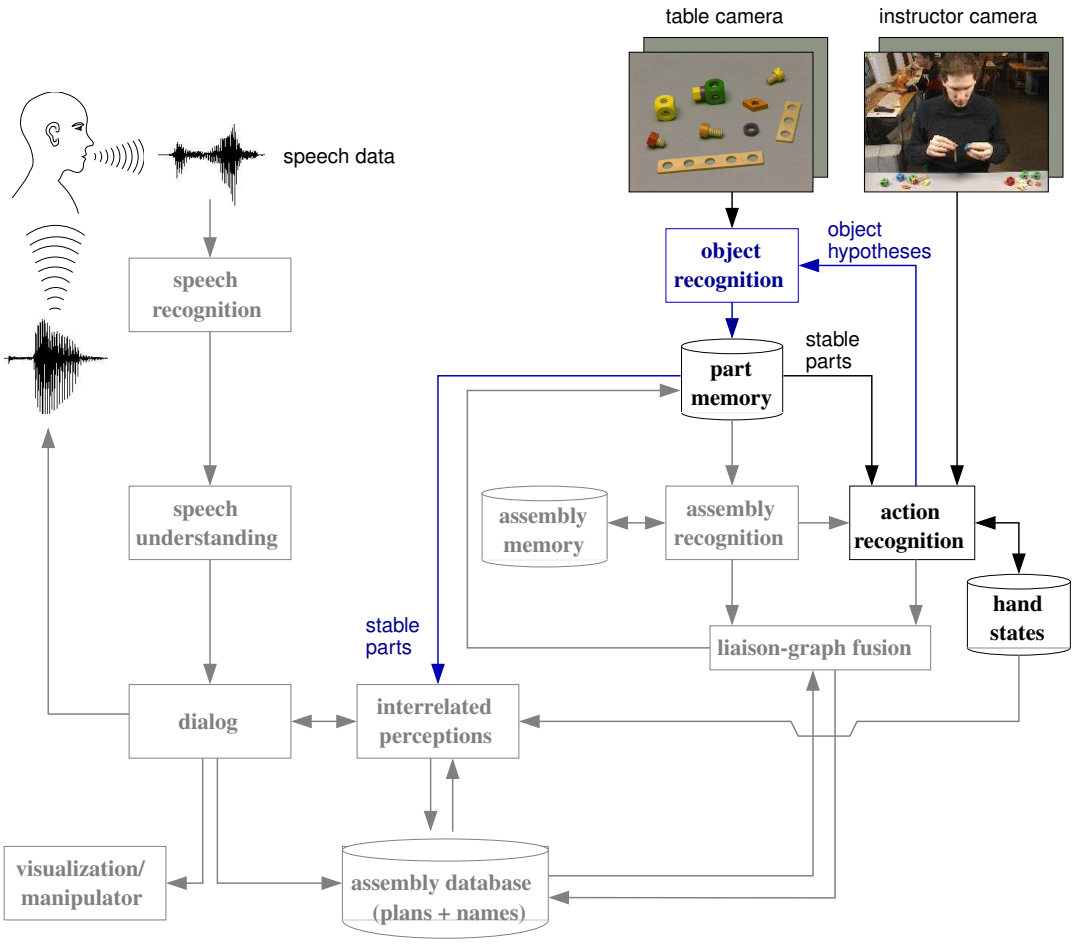


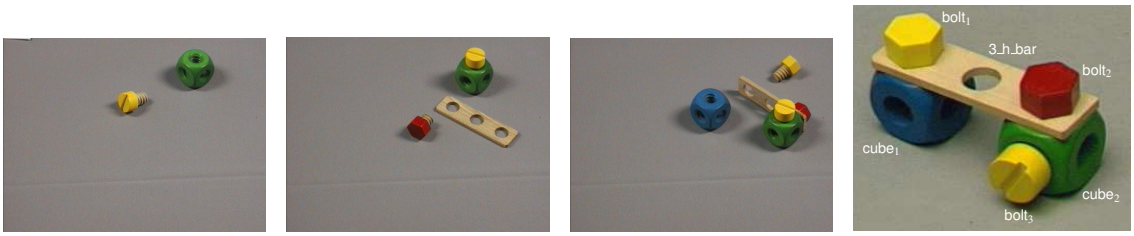
Figure 7.5: The incorporation of action recognition results as context knowledge for object recognition.

scene before picking a 3-hole-bar and a red bolt (see Fig. 7.6(b)). After putting the bar onto the bolt and tightening it with the previously built subassembly ASSEMBLY_1, this new subassembly ASSEMBLY_2 has to be placed back in the scene also (see Fig. 7.6(c)). This step is necessary to regrasp the assembly and to obtain the intermediate assembly structure. Then a yellow bolt was taken to put this new subassembly onto it and tighten it with a blue cube before placing it back on the table.

Constructing the assembly in this way leads to the following assembly structure contained in one hand state:

$\text{ASSEMBLY}_3 \rightarrow \text{bolt}_1\ \text{ASSEMBLY}_2\ \text{cube}_1$
$\text{ASSEMBLY}_2 \rightarrow \text{bolt}_2\ \text{3\_h\_bar}\ \text{ASSEMBLY}_1$
$\text{ASSEMBLY}_1 \rightarrow \text{bolt}_3\ \text{cube}_2$

The objects contained in this assembly structure are the expectations sent to the



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

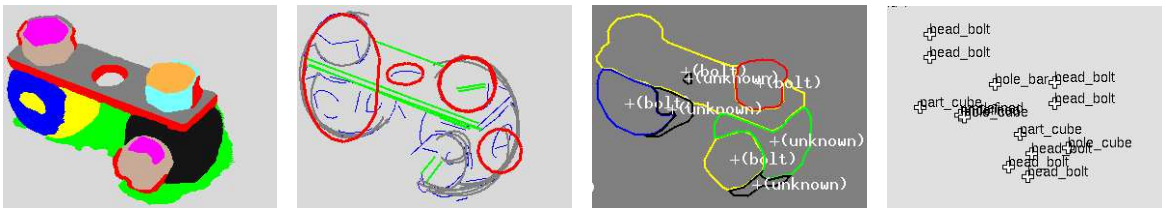
Figure 7.6: Example assembly with intermediate construction steps.

object recognition:

bolt bolt bolt 3_h_bar cube cube

Note that each time an assembly is placed back in the scene, the object recognition is carried out. Therefore, subassemblies in a larger assembly have been recognized several times during intermediate object recognition and assembly structure fusion steps. In the expectation data provided by the hand states, these subassemblies have been retrieved from the assembly database, i.e., the results of previous recognition/fusion steps are used.

For completeness, Fig. 7.7 depicts the results of the individual segmentation and classification results that are used to construct the segmentation hierarchy (see page 24).

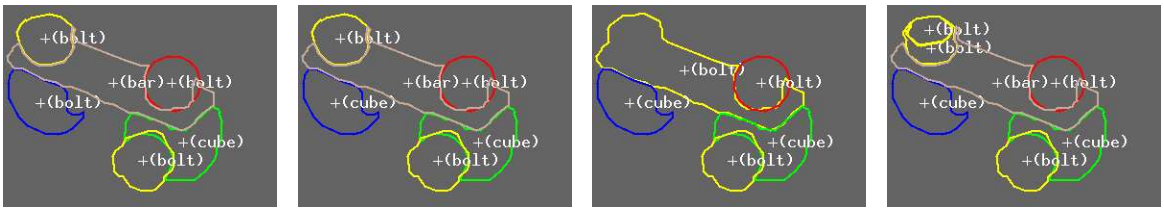


(a) (b) (c) (d)

Figure 7.7: Results for the example assembly: (a) Color-based region segmentation (mean-shift algorithm); (b) Contour-based perceptual grouping; (c) Hybrid object recognition; (d) Holistic detection of object parts;

In Fig. 7.8(a) an intermediate scene interpretation for the example assembly is shown after evaluating the segmentation hierarchy without process knowledge. The blue area has an evidence for *bolt* from the hybrid recognition module and one for *cube* from a classified focus point (see Fig. 7.7(c),7.7(d)). In this situation, voting results in hypothesizing *bolt*. The set of expected elements provided by the action recognition is matched to this scene interpretation. This results in adding a new evidence *cube* for the area with

the blue border while all other areas already contain the correct labels. Repeating the voting procedure produces the label *cube* for the area with the blue border as now this hypothesis has one evidence from the classified focus point and one from process knowledge, while the label *bolt* has only one evidence from the hybrid recognition module. The resulting scene interpretation is depicted in Fig. 7.8(b). Note that the resulting element region does not exactly match the object region due to segmentation errors. Two other scene interpretations generated by the segmentation hierarchy are shown in Fig. 7.8(c) and Fig. 7.8(d).



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

Figure 7.8: Example results. (a) Results without exploiting process knowledge. (b) Results after incorporating expectations from monitoring the construction process. (c), (d) Competing scene interpretations contained in the segmentation hierarchy that have low judgments.

Given the three competing results in Fig. 7.8 (b)-(d), these results are ranked using the judgment procedure. The ranking is determined by comparing the elements in a scene interpretation with the set of expected elements from action recognition. The result shown in Fig. 7.8(c) gets a low judgment because the 3_h_bar is missing. Likewise, the result depicted in Fig. 7.8(d) gets a low judgment because it contains one additional element hypothesis (the yellow bolt) which could not be matched. Only the result depicted in Fig. 7.8(b) contains exactly the same number and type of elements as provided by action recognition in the form of the expectations. Consequently, it gets a high judgment and is selected as recognition result.

For the performance evaluation, a small test set of 26 assemblies with a total of 119 elements was used. To measure the performance of the extended object recognition, the information from process knowledge consisted of correct expectations only. To account for the two stages in the recognition process, we measured the recognition rates for generating object hypotheses and selecting the best scene interpretation individually. For this purpose the best result out of all scene interpretations was manually selected resulting in a total of 72% correctly recognized elements compared to 70% without information from process knowledge. When using this information for selecting the best interpretation, the number of correct elements in the chosen scene interpretation increased from 66% to 71%. Note that the majority of errors is due to wrong segmentations and classifications which cannot be undone by context knowledge. However, using context knowledge

the generation of object hypotheses is slightly improved and the selection of the scene interpretation results in a better overall recognition performance.

7.4 Summary

In this chapter, we demonstrated the use of action recognition results for improving the human-machine interface of the Situated Artificial Communicator. On the one hand, the observation of the acting human can be used for directly improving the dialog interaction by providing information about the hand contents. On the other hand, the observation of the construction actions provides relevant context information for vision algorithms aiming at recognizing the objects and assemblies present in the scene. The correct extraction of the current scene is a highly relevant source of information for the human-machine interface of the Situated Artificial Communicator that shares the interaction domain with the human. Therefore, using the recognized actions as context knowledge results in an indirect improvement of the human-machine interface of the Situated Artificial Communicator.

The applications of the action recognition results presented in this chapter emphasize the need for analyzing non-communicative hand gestures to construct advanced human-machine interfaces. Only the extraction of human actions allows a technical system sharing the domain with the human to provide a sophisticated human-machine interface. We expect that an important application area where the recognition of manipulative gestures is intensively used will be mobile robots. In a mobile robot domain, the robot must observe the acting human and reason about him to react appropriately. The recognition of actions in an office environment has already been presented in Section 6.6, but due to a missing human-machine interface for a mobile robot we cannot demonstrate here the benefit of recognizing manipulative gestures for improving the interaction quality of a mobile robot. Nevertheless, the results obtained in the Situated Artificial Communicator domain already prove the utility of the proposed recognition framework for a system with a sophisticated human-machine interface.

8 Summary and Conclusion

The topic of this thesis is the recognition of manipulative hand gestures. Looking at the literature on gesture recognition shows that most of the published approaches deal with communicative gestures. However, many gestures during everyday life change the position of objects in their surrounding. These manipulative gestures are the primary form of non-communicative interaction between humans and their environment. As manipulative gestures are often considered to be irrelevant for communication between two humans and, consequently, in human-machine interaction, their recognition has attracted little attention up to now. This narrow view on communication ignores the fact that often knowledge about the current situation is required to understand communication acts and is therefore needed by a technical system aiming at enabling a natural human-machine interaction. We argue that the recognition of manipulative gestures in a system realizing a human-machine interface improves the interaction quality substantially by providing information about the current scene to other algorithms.

For the vision-based recognition of gestures, the image sequences depicting gestures have to be analyzed. Communicative gestures are characterized solely by the motion of the hand. Therefore, only sensory information in the form of the hand trajectories is required to recognize communicative gestures. For the recognition of manipulative gestures two types of information are relevant, the sensory hand trajectory data and the symbolic information about the manipulated objects. Only through incorporating the symbolic information it is possible to differentiate between gestures exhibiting similar trajectories like, e.g., 'picking up the phone' and 'taking a cup'. However, if certain restrictions are met by the user and the domain of interest, manipulative gestures can be recognized based only on one type of information.

In order to avoid relying on restrictions imposed by the type of information used, it is advantageous to combine symbolic and sensory information in an integrated recognition approach. We propose in this thesis a generic framework to perform integrated gesture recognition that is based on particle filtering. Through incorporating both, symbolic and sensory information, in the propagation phase of a particle filtering algorithm we achieve a parallel integration of both cues *during* the analysis process. This parallel integration facilitates the concentration of the computational resources on the gesture hypotheses that are likely given the observed symbolic and sensory data. With the proposed incorporation of symbolic information based on context areas, our approach allows us to recognize manipulative gestures independently of the location of the manipulated objects within the image. The developed integration scheme is domain-independent and

can be applied easily to different domains. Additionally, it supports the recognition of complex actions that involve a sequence of manipulative gestures interacting with different objects. For example, preparing a cup of coffee involves gestures interacting with the coffee pot and with sugar and milk.

In developing the integrated approach, we have first studied separately the advantages and limitations of symbolic and sensory information for recognizing manipulative gestures. For this purpose, we implemented two individual recognition approaches in the assembly construction domain. The symbolic action detection is a rule-based method to infer the executed actions based on appearing/disappearing parts and their functional properties. The symbolic information of the parts in the scene is provided by object and assembly recognition algorithms already available for the assembly construction domain. The realized symbolic action detection relies on the correct extraction of symbolic scene changes and provides information about the contents of the hands.

A complementary cue for recognizing construction actions is the sensory data representing the motions of the hands. The extraction of the features describing the hand motions is an important preprocessing step before a pattern recognition technique can be applied to analyze the motions. We developed for this purpose an adaptive skin color segmentation algorithm that allows us to detect hands and faces under varying lighting conditions and in front of cluttered backgrounds. By incorporating face detection results, the adaptation of the skin color model can be restricted to skin-colored faces. Similarily, the motion of skin-colored regions can be used to restrict the adaptation to moving hands to avoid distractions of the color model through skin-like background objects. The hands detected in the images of an image sequence are tracked over time to form trajectories describing the hand motions. These trajectories are analyzed using a particle filtering algorithm that has been used previously to recognize communicative gestures. As this approach is purely data-driven, it represents a visual activity recognition and is only appropriate for recognizing manipulative gestures that have characteristic trajectories. While the approach allows us to discriminate between hand motions with distinct trajectories, it does not provide any additional information about which objects were manipulated and what objects are contained in the hands.

The benefit of using the developed recognition approach integrating symbolic and sensory data is demonstrated by evaluating its recognition quality in the assembly construction domain. The increase in computational cost due to the integration of symbolic information is small. However, the additional information about the objects manipulated by a recognized gesture is highly relevant. It is only this symbolic information that allows us to generate an assembly construction plan from the recognized actions and differentiate between the gestures 'Pick' and 'Place' that exhibit similar hand motions. The implemented gesture recognition approach is therefore capable of recognizing manipulative gestures independent of their specific properties, i.e. without the restrictions imposed by the recognition approaches dealing with the individual information cues. The comparison of the recognition quality with the results of the data-driven activity recognition shows that the advantages of the proposed integration framework are

obtained without reducing the recognition quality of the particle filtering algorithm.

To demonstrate the flexibility of the proposed approach, a recognition system for recognizing manipulative gestures in an office environment was developed. Different from the assembly construction domain, there is a wider range of different gestures interacting with the different objects contained in an office. With the increased ambiguity of the hand trajectories, the symbolic information representing the manipulated objects becomes more important in the office domain. Therefore this domain is well-suited to demonstrate the benefit of combining both information types in an integrated framework. The implemented recognition system based on the integration framework is able to recognize the manipulative gestures despite of the ambiguities in the hand motions.

The relevance of recognizing manipulative gestures for improving human-machine interfaces has been demonstrated with various applications that were realized within the Situated Artificial Communicator setting. On the one hand, the knowledge of the current contents of the hands allows us to understand instructions related to parts that are contained in the hands. Recognizing manipulative gestures results here in a direct improvement of the communicative capabilities. On the other hand, the sequence of recognized construction actions can serve as context knowledge for vision-based object and assembly recognition algorithms leading to an indirect improvement of the communicative capabilities.

The applications in the Situated Artificial Communicator domain using gesture recognition results clearly demonstrate the importance of recognizing manipulative gestures for improving human-machine interfaces. The proposed framework for performing action recognition based on symbolic and sensory data proves the possibility to realize integrated recognition systems capable of real-time performance. In order to apply the proposed framework to the vision-based recognition of manipulative gestures, we developed an adaptive skin color segmentation approach. This approach enables the detection of hands in images captured under varying lighting conditions encountered in ordinary environments like, e.g., offices or homes. With the development of this image processing component to extract the hand motion, gesture recognition can be accomplished in a wide range of domains. We demonstrated the recognition of manipulative gestures in two domains, assembly construction and usual office behavior. In both domains, the gestures are recognized with a good precision and the symbolic context of the gestures, i.e., what object was manipulated, is available for subsequent processing steps. The recognition results demonstrate the suitability of our approach to recognize manipulative gestures. The proposed framework can be applied easily to other domains and, as demonstrated with the applications in the Situated Artificial Communicator, it improves the overall performance of systems interacting with humans who gesture with their hands.

Bibliography

[1] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428–440, 1999.

[2] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Trans. on Signal Processing*, 50(2):174–188, 2002.

[3] D. Ayers and M. Shah. Monitoring human behavior in an office environment. In *IEEE Workshop on Interpretation of Visual Motion, CVPR-98*, Santa Barbara, CA, June 1998.

[4] P. Bakker and Y. N. Kuniyoshi. Robot see, robot do: An overview of robot imitation. In *Proc. AISB'96 Workshop on Learning in Robots and Animals*, pages 3–11, Brighton, UK, 1996.

[5] Y. Bar-Shalom and T. Fortmann. *Tracking and Data Association.* Academic Press, Inc., San Diego, 1988.

[6] C. Bauckhage. *A Structural Framework for Assembly Modeling and Recognition.* Dissertation, Bielefeld University, Technical Faculty, 2002. available at: http://archiv.ub.uni-bielefeld.de/disshabi/2002/0062.pdf.

[7] C. Bauckhage, G.A. Fink, J. Fritsch, F. Kummert, F. Lömker, G. Sagerer, and S. Wachsmuth. An Integrated System for Cooperative Man-Machine Interaction. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 328–333, Banff, Canada, 2001.

[8] C. Bauckhage, J. Fritsch, F. Kummert, and G. Sagerer. Towards a Vision System for Supervising Assembly Processes. In *Proc. Symposium on Intelligent Robotic Systems (SIRS'99)*, pages 89–98, Coimbra, 1999.

[9] C. Bauckhage, J. Fritsch, K.J. Rohlfing, S. Wachsmuth, and G. Sagerer. Evaluating integrated speech- and image understanding. In *Proc. IEEE International Conference on Multimodal Interfaces (ICMI'02)*, pages 9–14, 2002.

[10] C. Bauckhage, J. Fritsch, and G. Sagerer. Memorizing Visual Knowledge for Assembly Process Monitoring. In B. Radig and S. Florczyk, editors, *Pattern Recognition*, Lecture Notes in Computer Science 2191, pages 178–185. Springer-Verlag, 2001.

[11] Y. Bengio and P. Frasconi. An input output HMM architecture. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 427–434. The MIT Press, 1995.

[12] M. J. Black and A. D. Jepson. A probabilistic framework for matching temporal trajectories: CONDENSATION-based recognition of gestures and expressions. *Lecture Notes in Computer Science*, 1406:909–924, 1998.

[13] A. Bobick. Movement, activity, and action: The role of knowledge in the perception of motion. *Phil. Trans. Royal Society London*, B(352):1257–1265, 1997.

[14] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, Reading, MA, 1999.

[15] A. Bourjault. *Contribution à une approche méthodologique de l'Assemblage Automatisé: Elaboration Automatique des Séquences Opératoires*. PhD thesis, Université de Franche-Comté, 1984.

[16] J. Brand and J.S. Mason. A comparative assessment of three approaches to pixel-level human skin-detection. In *Proc. International Conference on Pattern Recognition*, volume 1, pages 1056–1059, Barcelona, 2000.

[17] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Proc. Computer Vision and Pattern Recognition*, pages 994–999, Puerto Rico, USA, 1997.

[18] E. Braun, J. Fritsch, and G. Sagerer. Incorporating Process Knowledge into Object Recognition for Assemblies. In *IEEE International Conference on Computer Vision*, pages 726–732, Vancouver, CA, 2001.

[19] D. Comaniciu and P. Meer. Robust analysis of feature spaces: Color image segmentation. In *Proc. Computer Vision and Pattern Recognition*, pages 750–755, Puerto Rico, 1997.

[20] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proc. Computer Vision and Pattern Recognition*, volume 2, pages 142–149, Hilton Head Island, South Carolina, 2000.

[21] T. Darrell and A. Pentland. Space - time gestures. In *Proc. Computer Vision and Pattern Recognition*, pages 335–340, New York, 1993.

[22] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. B*, 39:1–38, 1977.

[23] M. Ehrenmann, P. Steinhaus, and R. Dillmann. A multisensor system for observation of user actions in programming by demonstration. In *IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, Aug 1999.

[24] F. Fairchild. *Color Appearance Models*. Addison-Wesley, 1998.

[25] G. A. Fink. Developing HMM-based recognizers with ESMERALDA. In Václav Matoušek, Pavel Mautner, Jana Ocelíková, and Petr Sojka, editors, *Lecture Notes in Artificial Intelligence*, volume 1692, pages 229–234, Berlin Heidelberg, 1999. Springer.

[26] G. A. Fink, N. Jungclaus, H. Ritter, and G. Sagerer. A communication framework for heterogeneous distributed pattern analysis. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 881–890, 1995.

[27] G.A. Fink, N. Jungclaus, F. Kummert, H. Ritter, and G. Sagerer. A Distributed System for Integrated Speech and Image Understanding. In *International Symposium on Artificial Intelligence*, pages 117–126, 1996.

[28] G. Finlayson, S. Hordley, and P. Hubel. Colour by correlation: A simple, unifying approach to colour constancy. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(11):1209–1221, 2001.

[29] J. Fritsch, H. Brandt-Pook, and G. Sagerer. Combining Planning and Dialog for Cooperative Assembly Construction. *IJCAI-99 Workshop "Planning and Scheduling meet Real-Time Monitoring in a Dynamic and Uncertain World*, pages 59–64, 1999.

[30] J. Fritsch, M. Kleinehagenbrock, S. Lang, T. Plötz, G. A. Fink, and G. Sagerer. Multimodal anchoring for human-robot-interaction. *Robotics and Autonomous Systems, Special issue 'Anchoring symbols to sensor data in single and multiple robot systems'*, 2003. to appear.

[31] J. Fritsch, S. Lang, M. Kleinehagenbrock, G. A. Fink, and G. Sagerer. Improving adaptive skin color segmentation by incorporating results from face detection. In *Proc. IEEE Int. Workshop on Robot and Human Interactive Communication (ROMAN)*, pages 337–343, Berlin, Germany, September 2002. IEEE.

[32] J. Fritsch, F. Loemker, M. Wienecke, and G. Sagerer. Detecting Assembly Actions by Scene Observation. In *International Conference on Image Processing (ICIP-2000)*, pages 212–215, Vancouver, 2000.

[33] J. Fritsch, F. Loemker, M. Wienecke, and G. Sagerer. Erkennung von Konstruktionshandlungen aus Bildfolgen. In *Mustererkennung 2000, 22. DAGM-Symposium Kiel*, Informatik aktuell, pages 389–396. Springer-Verlag, 2000.

[34] B. Funt. Color constancy in digital imagery. In *Proc. International Conference on Image Processing*, pages 55–59, Los Alamitos, CA, 1999. IEEE.

[35] B. Funt, K. Barnard, and L. Martin. Is machine colour constancy good enough? *Lecture Notes in Computer Science (ECCV 98)*, 1406:445–459, 1998.

[36] D. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[37] Z. Ghahramani and M.I. Jordan. Factorial hidden markov models. *Machine Learning*, 29:245–273, 1997.

[38] G. Gomez and E. Morales. Automatic feature construction and a simple rule induction algorithm for skin detection. In *ICML Workshop on Machine Learning and Computer Vision*, Quebec, Kanada, 2002. to appear.

[39] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (3rd. Ed.)*. Addison-Wesley Reading, 1992.

[40] S. Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

[41] G. Heidemann. *Ein flexibel einsetzbares Objekterkennungssystem auf der Basis neuronaler Netze. Dissertationen zur Künstlichen Intelligenz (DISKI 190). Sankt Augustin: infix-Verlag.* Dissertation, Universität Bielefeld, Technische Fakultät, 1998.

[42] G. Heidemann, F. Kummert, H. Ritter, and G. Sagerer. A Hybrid Object Recognition Architecture. In C. von der Malsburg, W. von Seelen, J.C. Vorbrüggen, and B. Sendhoff, editors, *Artificial Neural Networks – ICANN 96*, pages 305–310. Springer-Verlag, Berlin, 1996.

[43] G. Heidemann and H. Ritter. A neural recognition architecture for composed objects. In B. Jähne, P. Geißler, H. Haußecker, and F. Hering, editors, *Mustererkennung 1996, 18. DAGM-Symposium*, pages 475–482. Springer Verlag, 1996.

[44] K. Ikeuchi, M. Kawade, and T. Suehiro. Assembly task recognition with planar, curved, and mechanical contacts. In *Proc. International Conference on Robotics and Automation*, pages 688–694, 1993.

[45] K. Ikeuchi and T. Suehiro. Towards an Assembly Plan from Observation, Part I: Task Recognition with Polyhedral Objects. *IEEE Transactions on Robotics and Automation*, 10(3):368–385, 1994.

[46] St. S. Intille and A. F. Bobick. Recognizing planned, multiperson action. *Computer Vision and Image Understanding*, 81:414–444, 2001.

[47] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. *Lecture Notes in Computer Science*, 1064:343–356, 1996.

[48] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[49] M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *Proc. International Conference on Computer Vision*, pages 107–112, Mumbai, India, 1998.

[50] H. Fukamachi J.-C. Terrillon, M. N. Shirazi and S. Akamatsu. Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 54–61, Grenoble, France, 2000.

[51] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. In *Proc. Computer Vision and Pattern Recognition*, volume 1, pages 274–280, Ft. Collins, CO, 1999.

[52] Y. Keselman and S. Dickinson. Bridging the representation gap between models and exemplars. In *IEEE Computer Society Workshop on Models versus Exemplars in Computer Vision*, pages 76–83, Kauai, Hawaii, December 2001.

[53] M. Kleinehagenbrock, S. Lang, J. Fritsch, F. Lömker, G. A. Fink, and G. Sagerer. Person tracking with a mobile robot based on multi-modal anchoring. In *Proc. IEEE Int. Workshop on Robot and Human Interactive Communication (ROMAN)*, pages 423–429, Berlin, Germany, September 2002. IEEE.

[54] M. Kohler and S. Schröter. A Survey of Video-based Gesture Recognition - Stereo and Mono Systems -. Technical Report 693, Informatik VII, University of Dortmund, Germany, August 1998.

[55] D. Koller, K. Daniilidis, and H.H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257–281, June 1993.

[56] E. Koller-Meier and F. Ade. Tracking multiple objects using the condensation algorithm. *Journal of Robotics and Autonomous Systems*, 34:93–105, (2-3) 2001.

[57] H. Kruppa, M. A. Bauer, and B. Schiele. Skin patch detection in real-world images. In *DAGM Symposium*, pages 109–116, Zurich, Switzerland, September 2002.

[58] F. Kummert, G.A. Fink, G. Sagerer, and E. Braun. Hybrid Object Recognition in Image Sequences. In *Proc. International Conference on Pattern Recognition (ICPR'98)*, volume II, pages 1165–1170, Brisbane, 1998.

[59] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance. *IEEE Trans. on Robotics and Automation*, 10(6):799–822, 1994.

[60] Y. Kuniyoshi and H. Inoue. Qualitative recognition of ongoing human action sequences. In *Proc. International Joint Conference on Artificial Intelligence*, pages 1600–1609, 1993.

[61] S. Lang, M. Kleinehagenbrock, J. Fritsch, G. A. Fink, and G. Sagerer. Detection of communication partners from a mobile robot. In *Proc. 4th Workshop on Dynamic Perception*, pages 183–188, Bochum, Germany, November 2002. Akademische Verlagsgesellschaft Aka GmbH, Berlin.

[62] A. Leonardis. *Image Analysis Using Parametric Models*. PhD thesis, University of Ljubljana, 1993.

[63] R. Mann and A. Jepson. Towards the computational perception of action. In *Proc. Computer Vision and Pattern Recognition*, pages 794–799, Santa Barbara, CA, 1998.

[64] R. Mann, A. Jepson, and J. Siskind. The computational perception of scene dynamics. In *Proc. European Conference on Computer Vision*, pages 528–539, Cambridge, 1996.

[65] A. Maßmann, S. Posch, G. Sagerer, and D. Schlüter. Using markov random fields for contour-based grouping. In *Proc. International Conference on Image Processing*, volume II, pages 207–210, 1997.

[66] P. McGuire, J. Fritsch, J. J. Steil, F. Röthling, G. A. Fink, S. Wachsmuth, G. Sagerer, and H. Ritter. Multi-modal human-machine communication for instructing robot grasping tasks. In *Proc. IROS 2002*, pages 1082–1089. IEEE, 2002.

[67] J. Miura and K. Ikeuchi. Task-oriented generation of visual sensing strategies in assembly tasks. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(2):126–138, 1998.

[68] T. B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[69] D. Moore, I. Essa, and M. Hayes. Exploiting human actions and object context for recognition tasks. In *Proc. International Conference on Computer Vision*, volume 1, pages 80–86, Corfu, Greece, 1999.

[70] N. Esau and L. Steinborn. Konzeption und Realisierung eines Aggregatgedchtnisses zur Analyse von Konstruktionsprozessen. Master's thesis, Universität Bielefeld, Technische Fakultät, 2001.

[71] H.-H. Nagel. From image sequences towards conceptual descriptions. *Image and Vision Computing*, 6(2):59–74, 1988.

[72] 2001 NEC Press release on March 21st. Nec develops friendly, walkin' talkin' personal robot with human-like characteristics and expressions. http://www.nec.co.jp/press/en/0103/2103.html.

[73] N. Oliver, A. Pentland, and F. Berard. Lafter: a real-time face and lips tracker with facial expression recognition. *Pattern Recognition*, 33:1369–1382, 2000.

[74] N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. In *First Int. Conf. on Computer Vision Systems*, pages 255–272, Gran Canaria, Spain, 1999.

[75] B. Parhami. Voting algorithm. *IEEE Trans. on Reliability*, 43(4):617–629, 1994.

[76] V. Pavlovic, R. Sharma, and T. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):677–695, 1997.

[77] P. Peixoto, J. Batista, and H. Araújo. Real-time human activity monitoring exploring multiple vision sensors. In H. Araújo and J. Dias, editors, *Proc. 7th Int. Symposium on Intelligent Robotic Systems*, pages 187–195, July 1999.

[78] A. Psarrou, S. Gong, and M. Walter. Recognition of human gestures and behaviour based on motion trajectories. *Image and Vision Computing*, 20(5-6):349–358, 2002.

[79] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, January 1986.

[80] Y. Raja, S. McKenna, and S. Gong. Segmentation and tracking using colour mixture models. In *Proceedings of the 3rd Asian Conference on Computer Vision*, pages 607–614, 1998.

[81] Y. Raja, S. McKenna, and S. Gong. Tracking and segmenting people in varying lighting conditions using colour. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 228–233, Nara, Japan, 1998.

[82] Y. Raja, S. J. McKenna, and S. Gong. Colour model selection and adaptation in dynamic scenes. In *Proc. European Conference on Computer Vision*, pages 460–474, 1998.

[83] J. Rehg and T. Cham. A multiple hypothesis approach to figure tracking. In *Proc. Computer Vision and Pattern Recognition*, volume 2, pages 239–245, Ft. Collins, CO, 1999.

[84] Ch. Rosenberg, M. Hebert, and S. Thrun. Color constancy using KL-Divergence. In *Proc. International Conference on Computer Vision*, pages 239–246, Los Alamitos, CA, 2001. IEEE Computer Society.

[85] G. Sagerer, C. Bauckhage, E. Braun, G. Heidemann, F. Kummert, and H. Ritter. Integrating recognition paradigms in a multiple-path architecture. *Lecture Notes in Computer Science*, 2013, 2001.

[86] G. Sagerer and H. Niemann. *Semantic Networks for Understanding Scenes*. Plenum Publishing Corporation, New York, 1997.

[87] J. Sherrah and S. Gong. Vigour: A system for tracking and recognition of multiple people and their activities. In *Proc. International Conference on Pattern Recognition*, volume 1, pages 179–182, Barcelon, Spain, 2000.

[88] H. Sidenbladh, M. J. Black, and D. J. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *Proc. European Conference on Computer Vision*, volume 2, pages 702–718, Dublin, Ireland, 2000. Springer.

[89] J. Siskind. Visual event perception. In K. Ikeuchi and M. Veloso, editors, *Symbolic Visual Learning*. Oxford University Press, New York, 1997.

[90] M. Soriano, B. Martinkauppi, S. Huovinen, and M. Laaksonen. Skin detection in video under changing illumination conditions. In *Proc. Computer Vision and Pattern Recognition*, volume 1, pages 839–842, 2000.

[91] T. Starner and A. Pentland. Visual recognition of american sign language using hidden markov models. In *International Workshop on Automatic Face and Gesture Recognition*, pages 189–194, Zurich, Switzerland, 1995.

[92] M. Störring, H. J. Andersen, and E. Granum. Skin colour detection under changing lighting conditions. In Helder Araújo and Jorge Dias, editors, *SIRS'99 Proc. 7th Int. Symposium on Intelligent Robotic Systems*, pages 187–195, July 1999.

[93] M. Störring, H. J. Andersen, and E. Granum. Physics-based modelling of human skin colour under mixed illuminants. *Robotics and Autonomous Systems*, 35(3-4):131–142, 2001.

[94] G. J. Sussman. *A Computer Model of Skill Acquisition*. Elsevier Computer Science Library, 1975.

[95] J. Triesch and C. von der Malsburg. A gesture interface for human-robot-interaction. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 546–551, Nara, Japan, 1998. IEEE.

[96] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuro Science*, 3(1):71–86, 1991.

[97] S. Ullman. Visual routines. *Cognition*, 18:97–159, 1984.

[98] C. von Hardenberg and F. Berard. Bare-hand human-computer interaction. In *ACM workshop on Perceptive User Interfaces (PUI 2001)*, Orlando, Florida, 2001.

[99] I. Wachsmuth and B. Jung. Dynamic conceptualization in a mechanical-object assembly environment. *Artificial Intelligence Review*, 10(3-4):345–368, 1996.

[100] M. A. Webster. Human colour perception and its adaptation. *Network: Computation in Neural Systems*, 7:587–634, 1995.

[101] M. Wienecke. Erkennung und Klassifikation von Konstruktionshandlungen aus Bildfolgen. Master's thesis, Universität Bielefeld, Technische Fakultät, 2000.

[102] M. Wooldridge and N. R. Jennings. *Intelligent Agents — Theories, Architectures, and Languages*, volume 890 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1995.

[103] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Analysis and Machine Vision*, 19(7):780–785, 1997.

[104] Y. Wu and T. S. Huang. Vision-based gesture recognition: A review. In *3rd Gesture Workshop*, pages 103–115, Gif-sur-Yvette, France, 1999.

[105] G. Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley and Sons, New York, 1982.

[106] J. Yamamoto, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov models. In *Proc. Computer Vision and Pattern Recognition*, pages 379–387, 1992.

[107] J. Yang, W. Lu, and A. Waibel. Skin-color modeling and adaptation. *Lecture Notes in Computer Science*, 1352:687–694, 1998.

[108] M. Yang, N. Ahuja, and M. Tabb. Extraction of 2d motion trajectories and its application to hand gesture recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(8):1061–1074, 2002.

[109] B.D. Zarit, B.J. Super, and F.K.H.Quek. Comparison of five color models in skin pixel classification. In *Proc. of Int. Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 58–63, Corfu, Greece, 1999.

Index