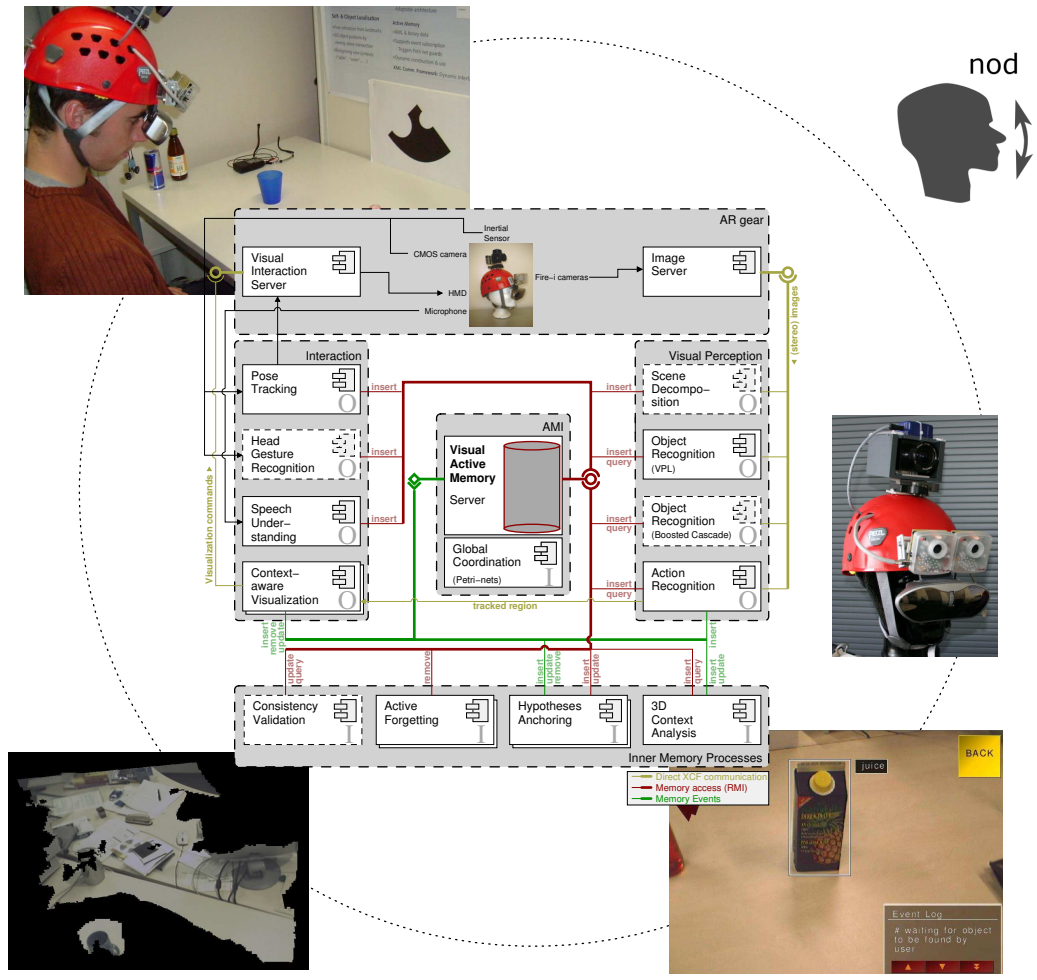
A Cognitive Ego-Vision System for Interactive Assistance

Marc Hanheide



Dipl.-Inform. Marc Hanheide
AG Angewandte Informatik
Technische Fakultät
Universität Bielefeld
email: mhanheid@techfak.uni-bielefeld.de

Abdruck der genehmigten Dissertation zur Erlangung
des akademischen Grades Doktor-Ingenieur (Dr.-Ing.).
Der Technischen Fakultät der Universität Bielefeld
am 25.10.2006 vorgelegt von Marc Hanheide,
am 20.12.2006 verteidigt und genehmigt.

Gutachter:

Dr.-Ing. Sven Wachsmuth, Universität Bielefeld
Prof. Dr. Josef Kittler, University of Surrey

Prüfungsausschuss:

Prof. Dr. rer. nat. Helge Ritter, Universität Bielefeld
Dr.-Ing. Sven Wachsmuth, Universität Bielefeld
Prof. Dr. Josef Kittler, University of Surrey
Dr. rer. nat. Thomas Hermann, Universität Bielefeld

Gedruckt auf alterungsbeständigem Papier nach ISO 9706

A Cognitive Ego-Vision System for Interactive Assistance

Der Technischen Fakultät der Universität Bielefeld

zur Erlangung des Grades

Doktor-Ingenieur

vorgelegt von

Marc Hanheide

Bielefeld – Oktober 2006

Abstract

With increasing computational power and decreasing size, computers nowadays are already wearable and mobile. They become attendant of peoples' everyday life. Personal digital assistants and mobile phones equipped with adequate software gain a lot of interest in public, although the functionality they provide in terms of assistance is little more than a mobile databases for appointments, addresses, to-do lists and photos. Compared to the assistance a human can provide, such systems are hardly to call real assistants.

The motivation to construct more human-like assistance systems that develop a certain level of cognitive capabilities leads to the exploration of two central paradigms in this work. The first paradigm is termed *cognitive vision systems*. Such systems take human cognition as a design principle of underlying concepts and develop learning and adaptation capabilities to be more flexible in their application. They are embodied, active, and situated. Second, the *ego-vision* paradigm is introduced as a very tight interaction scheme between a user and a computer system that especially eases close collaboration and assistance between these two. Ego-vision systems (EVS) take a user's (visual) perspective and integrate the human in the system's processing loop by means of a shared perception and augmented reality. EVSs adopt techniques of cognitive vision to identify objects, interpret actions, and understand the user's visual perception. And they articulate their knowledge and interpretation by means of augmentations of the user's own view.

These two paradigms are studied as rather general concepts, but always with the goal in mind to realize more flexible assistance systems that closely collaborate with its users. This work provides three major contributions. First, a definition and explanation of ego-vision as a novel paradigm is given. Benefits and challenges of this paradigm are discussed as well. Second, a configuration of different approaches that permit an ego-vision system to perceive its environment and its user is presented in terms of object and action recognition, head gesture recognition, and mosaicing. These account for the specific challenges identified for ego-vision systems, whose perception capabilities are based on wearable sensors only. Finally, a *visual active memory* (VAM) is introduced as a flexible conceptual architecture for cognitive vision systems in general, and for assistance systems in particular. It adopts principles of human cognition to develop a representation for information stored in this memory. So-called memory processes continuously analyze, modify, and extend the content of this VAM. The functionality of the integrated system emerges from their coordinated interplay of these memory processes.

An integrated assistance system applying the approaches and concepts outlined before is implemented on the basis of the visual active memory. The system architecture is discussed and some exemplary processing paths in this system are presented and discussed. It assists users in object manipulation tasks and has reached a maturity level that allows to conduct user studies. Quantitative results of different integrated memory processes are as well presented as an assessment of the interactive system by means of these user studies.

Acknowledgments

This is the place to say: Thank You! And I like to say this to many people that have accompanied me on my way, which destination is marked by this thesis. Looking back from where I stand now, I first feel obliged to my supervisor Sven Wachsmuth, who helped me out with his comprehensive expertise several times and inspired my work a lot. Thank you for your most valuable input. I know, that not only I consider Josef Kittler as an outstanding researcher in computer vision and pattern recognition. Therefore, I am grateful that he agreed to review my thesis.

Furthermore, I want especially thank Sebastian Wrede, who was more than a project partner in vampire and an office mate to me. With his enthusiasm, thoroughness, and complementary expertise he mostly inspired and influenced my work, and also broadened my horizon. Thanks for the long nights of joint coding, demo preparation, and many joyful moments in the last years, Sebastian. Of course, I should not forget to mention Christian Bauckhage with his most motivating slogan "Go for it!". He pushed me to where I would have never thought I would like to be. Thank you for the most friendly pressure.

The list of people who contributed to this thesis directly or indirectly is too long too mention all of them. My thanks go to the VAMPIRE project partners in Erlangen, Guildford, Bielefeld and Graz. I very much enjoyed the work in this project. An integrated system as presented in this thesis would have never been possible without their contribution.

Of course, I have to thank all diploma student, I had the pleasure to supervise and to benefit from their work: Nicolas Gorges, Marcel Richter, Viktor Peters, and Hector Lopez. And not to forget the students that helped the project to become a success: First of all, Jan "Hilfskraft" Schäfer, but also Christian Lang, Dinu Niessner, and Marcel Gärtner.

I must also not forget, who initially motivated me to do this thesis and who gave me the opportunity to actually finish it. Gerhard Sagerer made me feel like being at the right place and doing the right job. It is – not only – his merit that working in the applied computer science group was great and still is. I got to know my colleagues as always helpful and cooperative scientists. I would like to mention Nils Hofemann, Britta Wrede, Katharina Rohlfing, Jannik Fritsch, Frank Lömker, and Ingo Lütkebohle especially who have all contributed with their work in one or another way. Thanks also go to the proof readers that have not been mentioned so far; Thomas Käster, Julia Kalb, Dirk Stößel and Timo Thomas. Also the two other members of my examination board, Helge Ritter and Thomas Hermann, deserve my gratitude for the time they offered to spend on my work.

I also thank my parents, who always supported me and gave me the inspiration and liberty to find my way. Finally, of course, I have a deepest gratitude towards Anika. She has always been patient with me even when I was almost close to insanity, cleared my path, and helped me out of valleys of doubts whenever necessary. Anika, I love to share moments of success, happiness, craziness, and also doubts with you further on. Thank you!

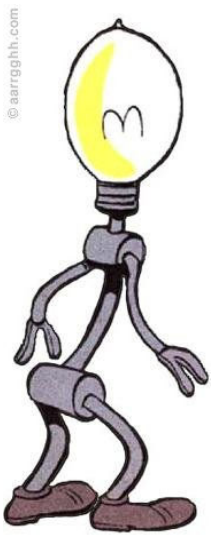
Contents

List of Figures

1 Introduction

If you search for a helping hand, look at the end of your arm.
[proverb]

"Where have I put my keys?", "What's that for?", "And how to do it?"



How often have you asked yourself such questions? And how often did you not receive a proper answer? There are various situations in human life where one seeks for assistance: The well-known annoying situation that you cannot remember where you have put an item, for instance. In yet unknown situations or in case of new tasks to accomplish, you have to learn from either observation or instruction of an expert how to handle objects and how to reach specific (sub-)goals.

Having an assistant who is always available when needed, who takes part at one's activities and can give answers to arising questions can be really helpful to extend limits of memorization capabilities, assist in yet unknown situations, or provide a "helping hand" whenever needed. Think of an assistant who recognizes where one's keys have been put and can tell on request. Or who knows how to assemble some complex furniture, can help you to program the reputed user-friendly coffee machine, or can recall the recipe of your favorite drink. Apparently, assistance is provided to everyone in various situations everyday by teachers, secretaries, friends, or others, who share episodes of our lives with you. All these people around you help, assist, and interact all the time. They observe what you are doing, answer questions you have, but also acquire knowledge from you by questioning and observing. Assistance is therefore also an interaction task involving *perception*, *understanding* and *articulation* of knowledge.

An assistance scenario involving two persons is exemplarily sketched in Fig. 1.1. The knowledge of both agents is usually different and interaction between the two is taking place to exchange and align it. Often one person is considered to be an expert providing assistance to the other, a novice. Generally, both can perceive and act in the environment, but restrictions may apply in specific scenarios, where only one can act in the environment. Although one-to-many or even many-to-many situations are conceivable in assistance situations, we will focus on pair-wise assistance here.

Example 1.1: *Zoe, taking the role of an expert in this example, assists John to assemble a wardrobe by reading the assembly instruction step-wise and referencing currently necessary parts. John can pose callbacks to get additional help.*

Note, that in this specific example, the expert (Zoe) is not directly active in the environment. But on the contrary, the novice can also restrict himself to the role of an observer to, e.g., learn from this observation. But usually roles of actor and observer change dynamically

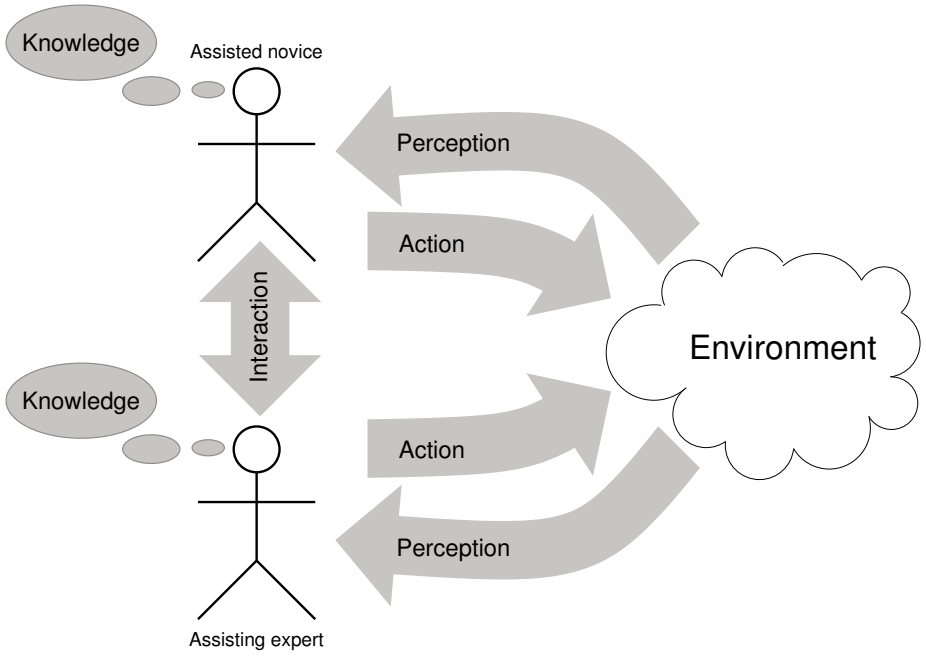


Figure 1.1: A general schema of assistance.

during interaction in assistance scenarios.

But assistants need not necessarily be human. With computers being more powerful and smaller, artificial assistants are becoming realistic. Already today many people carry so-called *personal digital assistants (PDA)* or mobile phones with similar functionality. These are used to organize appointments, take memos, or ToDo lists, etc. in daily life.

1.1 Cognitive Assistants

However, the functionality of recent PDAs is far away from what we expect from a human assistant. They are just portable data storages. Every piece of information stored on such a device must be given by explicit user input. They have very limited input and articulation capabilities for interaction with the user. But the major difference compared to human assistants is the inflexibility and inability to act situation- or context-aware. A human – as long as she or he is kind – would not interrupt you with a notification that you have an appointment when you are just talking to some other person, to give an example. And when accomplishing a task a human assistant would recognize when you have problems and offer proper help only if needed. An assistant would only really be a personal one, if it is able to respect your personal preferences and also to some extend has *anticipatory* capabilities. And if it furthermore is able to *interpret* situations and to act accordingly.

Hence, when artificial assistant systems should ever be able to step in for another human it is crucial that they become able to perceive and understand the situation and know about the goals of the user. They should be *perceptive*, *attentive*, *cooperative*, and *interactive*. As such abilities are usually associated with human cognition, we term artificial assistance systems that develop such abilities ***cognitive assistance systems (CAS)***.

In contrast to the terminology in constellation of a human assisting another human, we use another terminology here. We talk about a *user*, who uses a *system* in the following.

Usually, the system takes the role of an expert assisting the user, but it will turn out, that this roles needs to swapped from time to time, since the system also requires means to obtain certain knowledge itself. The following list gives a definition of some major abilities of cognitive assistance systems already mentioned before.

Attentive An assistant is expected to take care about what the user does. It should pay attention whether she or he needs assistance and response appropriately to requests. It must be situation-aware in two senses: First, it should recognize what is going on around the user. And second, it should also be attentive to the user's wishes and needs. Both capabilities are required to enable another ability, which is introduced next.

Cooperative In order to be helpful a CAS should be highly cooperative. It should provide the knowledge and help needed in a specific situation. Usually assistance is about the accomplishment of dedicated tasks. The system should be cooperative with respect to this task. Being cooperative also means to be attentive with respect to the information required by the user. The selection of information must follow a cooperative strategy. Thus, the system only provides knowledge when it is needed and is refraining in situations where no assistance is requested at all. Cooperativeness also demands for an at most *unintrusive* assistance. The assisted person should not be bothered with unnecessary or distracting information, but should be assisted with focus on the task.

Interactive Interaction between the user and the CAS is crucial in order to allow cooperation and facilitate attentive behavior. It takes part when expressing needs or requests, receiving instructions, directing attention, explaining situations, and exchanging knowledge. CAS have therefore also a strong relation to research in *human-computer-interaction (HCI)* in general. Interaction is often carried out in multimodal fashion using various communication channels like text, speech, gestures, facial expressions, and many more.

Adaptive and Learning In order to allow a CAS to be used in various situations adaptation is fundamental in terms of pro-active and interactive acquisition of new knowledge. This is closely related to communication about situations and attentive behavior. Furthermore, a CAS must be able to adapt to new situations that vary to a certain extend from the situation it has initially been designed for. Not every human always requires the same amount of assistance, and a system should ideally also adapt to these user-specific demands.

This list does not cover all requirements of cognitive assistance systems one can think of, but defines the basic assumptions for the conceptual design of such systems. Taking a closer look at the requirements – especially focusing on a *personal* assistant that closely collaborates with *one user* – unveils two further aspects derived from the above mentioned ones. First, ***perception*** plays a crucial role for situation awareness that allows the system to be *cooperative*. Perception of humans is multi-modal and covers vision, acoustics, and haptics. Of these, vision is known to be the most prominent input cue for perceiving the environment and is also a long studied subject in computer science. Second, close ***collaboration*** between the system and its user is necessary to enable directed attention, cooperative problem solving, and effective communication. Especially when focusing on personal assistance, close interaction and collaboration is essential.

With these definitions in mind, one can ask what typical scenarios do exist, in which artificial assistance systems can be useful? First, such scenarios include teaching applications in

which an assistance system has some expert knowledge that should be taught to the user. As teaching is an iterative process, the CAS should not only articulate its pre-stored knowledge but also supervise and validate the activities of the user in order to correct and intervene in case of errors and to provide individual feedback. Second, picking up the idea of PDAs again, the idea of a *cognitive* personal digital assistance that is able to recall things a user might have forgotten about is a scenario CAS can target at.

Example 1.2: *Zoe is an owner of a CAS that attends her every-day life. As many times before, she is standing in front of her flat and desperately scanning all her pockets for the door keys. She can now ask her system, which will tell her that she left the keys on her office desk. Ideally, the CAS will remember that the keys are needed for that door and will remind Zoe of her keys next time immediately when she leaves her office.*

Such a system is *attentive* in the sense, that it recognizes and anticipates the needs of its user. And is also *cooperative* by means of providing a goal-oriented advice. Another example that outlines how CAS compensate for memory deficiencies is the following:

Example 1.3: *John is very bad in associating faces and names. But luckily he owns a CAS that can display facts about persons John is meeting by recognizing their faces.*

So finally, developing the idea of a cognitive PDA further, a CAS that can step in for specific limited cognitive capabilities of impaired people can be thought of in the future.

Up to now, providing personal assistance by an artificial system has been discussed as a general problem. Accordingly, CAS have been presented as general assistance systems. In fact, systems that can compete with human capabilities at all levels are still a very long way to go. But artificial assistance systems that can be applied in specific scenarios already constitute a much more realistic challenge. They already allow to tackle scientific questions of CAS in general and solve dedicated application tasks in particular. Hence, this work will study general aspects of cognitive assistance systems, but will focus on more specific scenarios to assess the system's capabilities from a scientific point of view.

When investigating towards personal cognitive assistance systems major questions have to be posed and considered in the design of such systems:

- ▷ ***How to perceive the environment?***
  There are numerous approaches for a CAS to perceive the environment inspired from human abilities. The emphasis in this work is on visual perception. But asking *how* to perceive the scene also comprises the question of *where* to perceive it. I propose to take a user's perspective to perceive the scene from her or his point of view exclusively by head-mounted cameras. This unique approach is termed ***ego-vision*** and it will be discussed how it affects the design of the CAS in terms of algorithms, hardware and software.

- ▷ ***How to realize collaboration with the user?***
  Taking the user's point of view also facilitates attentive behavior and close collaboration between the system and its user directly. In an ***Ego-Vision System*** the user and the system share the same view and correspondingly have a shared visual context. Furthermore, it is proposed to integrate the ***human in the loop*** of processing by applying *augmented reality* techniques. By means of these, the system not only perceives the scene from a user's perspective but also can provide visual ***feedback*** and augment the user's view.

- ▷ ***How to acquire, store, use, and articulate (task-dependent) knowledge?***
  The prominent role of knowledge in assistance scenarios has already been stressed. Knowledge is acquired by either perceiving the environment or by user interaction.

Mostly a combination of both – an expert user explaining the perceived situation – enables the system to learn and adapt. In this work, an approach termed ***visual active memory*** is utilized for storage of acquired knowledge and information. This memory is called *active* because it allows to coordinate different processes of a cognitive assistance system following an *information-driven* approach. The prominent role of information is hence directly reflected by the architectural approach.

- ***How to integrate required functionalities?***
  Building CAS requires many different functionalities to be integrated into one system on the basis of an appropriate system architecture and integration concept. This has also to consider issues about how information is fused and how it is interpreted and regarding the specific process flow in the system. The system architecture also effects the design of the individual components. Optimally, the system architecture goes hand in hand with an appropriate software architecture. Therefore, the integration concept for the CAS is developed as a joint work with the developer of the underlying framework [168] that facilitates coordination and integration.

1.2 Contribution

My work takes cognitive assistance systems (CAS) as a challenging scenario to study aspects of computer vision, artificial cognitive systems, and system architectures. It will accordingly also provide answers to the questions posed in the previous section concerning CAS. Focusing on computer vision techniques and interaction with the user, two central paradigms affect the design and development of these systems. First, inspired by ideas of ***cognitive vision systems***, integration and coordination based on a ***visual active memory*** is presented. The central role of some kind of memory is presented as a major assumption for such systems. I will focus on the representation of information and knowledge in such a memory. Second, the idea of taking the user's (visual) perspective and integrating the user in the processing loop leads to the development of ***Ego-Vision Systems*** (EVS) which imposes certain challenges and benefits with respect to involved concepts and algorithms.

On the one hand, this work will present specific findings that are valid for the targeted cognitive assistance systems in particular. But on the other hand, more general insights regarding the selection of algorithms under the given assumptions and paradigms and their integration into an architecture of interactive ego-vision systems will be presented. Thus, the assistance system will also be considered as a case study in order to gather findings about ego-vision systems in general and to explain the proposed concepts. Answers will mainly be provided for the following questions:

- How to deal with limited perception and control capabilities arising from the idea of ego-vision?
- How to integrate and coordinate functionalities in an interactive vision system? How does an adequate architecture looks alike?
- How to make use of the close coupling between user and system by means of ego-vision? How can the user be integrated in the processing loop effectively and efficiently?

Although many different separate answers are possible to these questions, this work proposes a self-contained solution to the outlined challenge, that is "How to construct an interactive

ego-vision assistance system". Targeting at this ambitious goal, we try to find the best trade-off between the two paths towards 'complete' cognitive systems Castelfranchi [21] has identified:

> *(i) obtaining a working, integrated, complete picture of an internal cognitive architecture, and of its components and functions (e.g., decision and reaction, inference and learning, emotions and 'rational' reasoning);*
>
> *(ii) completing the coupling between the agent and its environment in order to have an effective, complete loop: perception, cognitive processing, (re)action, effects or independent changes in the world, perception of changes, and so on.*

Castelfranchi argues a lack of systems that follow both paths in parallel. In this thesis, an architectural point of view is taken on the subject of a closed interactive loop between the system and its user.

The design and development of such integrated systems always demands for a broad spectrum of expertise and can only be realized in acceptable time as a joined project. Therefore, a strong collaboration between different project partners is essential. Many aspects regarding the integration approach and the system architecture have been developed in close cooperation with Sebastian Wrede, who mainly investigated on an appropriate framework for system integration and coordination [172]. His framework constitutes the basis for the system architecture that has been realized on top. In order to make my particular contribution more vivid, every chapter that presents approaches developed to realize a cognitive ego-vision system for interactive assistance is concluded by a summary of the major findings explicitly exposing the contributions.

1.3 Outline

The following chapter introduces the two central paradigms – ***cognitive vision systems*** and ***ego-vision*** – and how these define the general concept of the presented system. Afterwards, I set out to answer the central three questions outlined above. First, I address specific challenges regarding visual perception from an ego-vision perspective in chapter 3. The chapter proposes a number of approaches especially designed to account for these challenges, and to allow the envisioned system to perceive its environment and the user. In particular these are approaches to implement a compact visual representation of the scene using mosaics, recognize objects and actions, and establish a three-dimensional scene model.

An attribute of ego-vision systems is, that they integrate the human user in the processes loop. In order to close this loop the system needs not only to perceive the environment but also has to provide feedback and carry out interaction with the user. Visualization, feedback capabilities and interaction modalities in an *augmented reality* setup of the ego-vision system are therefore subject to chapter 4. Also the hardware setup used for the ego-vision system is introduced in that chapter.

Having the basic functionalities for perception and interaction at hand, chapter 5 presents how the so-called *visual active memory (VAM)* concept is applied to realize a cognitive assistance system following paradigms of *cognitive vision systems*. The different components of a system designed according to the VAM concept constitute *memory processes*. The role, taxonomy and some generic classes of memory processes are discussed in chapter 6.

In chapter 7 the system architecture of an implemented cognitive assistance system is presented. This system architecture is designed on the basis of collaborating memory processes that are coordinated and interrelated using the VAM concept. Afterwards, results of evaluative studies of different system components in particular and also the integrated system will be presented in chapter 8. The thesis closes with a discussion on the insights gathered and a conclusion in chapter 9.

2 Cognitive Ego-Vision Systems

Four eyes see more than two.
[german proverb]

Computer vision has a long tradition in computer science and emerged as an own discipline in the 1970s. Ever since than, images where no longer only captured and processed by computers to be manipulated or stored, but interpreted by (semi-) automatic systems that try to understand the *content* of the vision input. "Understanding" in this sense involves to detect structures in images that carry a meaning, to interpret situations, and to identify entities and objects. In computer vision research, the human visual and cognitive capabilities are the "holy grail", and are often considered as a model and as a measure for artificial vision as well. Marr [102] defined computer vision in 1982 as a

> *process that creates, given a set of images, a complete and accurate representation of the scene and its properties.*

This definition provides a very general notion of computer vision, which is accordingly often referred to as "*general vision*" [4].

However, a human-like general one-and-only solution to computer vision is not available and presumably will never be. In order to enhance the visual capabilities of computers today, research followed several different pathways, often adopting insights of other disciplines. Today, computer vision is closely related to many different fields of computer science like *artificial intelligence*, *signal processing*, *pattern recognition*, and *image processing*.

The focus of research and the applied methods changed several times in the history of computer vision science, targeting at specific challenges derived from the general problem to allow computers to see. The borderlines are often hard to draw and the different methodologies and general approaches converged from time to time.

In the following, a brief overview of different general paradigms and approaches in computer vision shall be given, converging into the definition of ***cognitive computer vision (CCV)*** in section 2.2. Afterwards, a novel paradigm which provides the basis for the work presented here is introduced as "***ego-vision systems (EVS)***".

2.1 Computer Vision Systems

Computer systems that develop capabilities to perceive, interpret, and understand visual input to some extend are generally termed ***computer vision systems***. Such systems are usually constituted of many different functionalities or components in order to pursue some *goal*. This goal can be diverse and of varying complexity, but today's computer

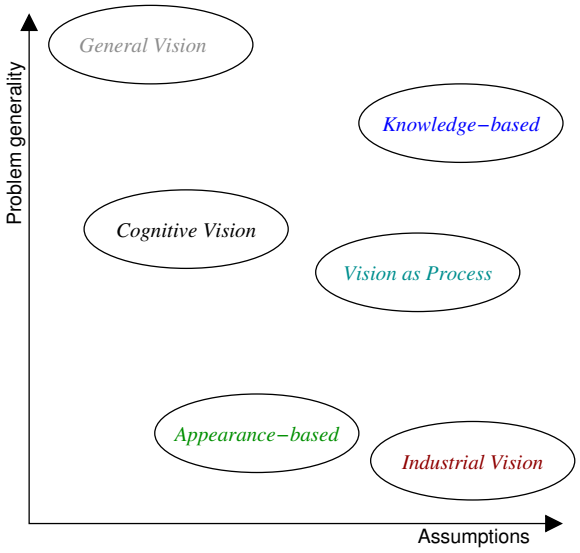


Figure 2.1: Classification of computer vision paradigms according to problem complexity and amount of assumptions made.

vision research has mostly disbanded from Marr's idea [102] of general vision. Instead, the *generality* of the problem to solve is one axis that allows to distinguish different approaches in computer vision. Another axis is the amount *assumptions* or restrictions that have to be applied in order to attain a certain goal.

In Fig. 2.1 a selection of different paradigms in computer vision are contrasted according to these two attributes. ***Industrial computer vision*** is mostly concerned with very specific problems or tasks in controlled environment. Hence, such systems are located on the opposite side of general vision. Other paradigms can be found in between these two extremes.

In order to pursue the goal of computer vision systems, different general paradigms have been followed by researchers. ***Knowledge-based computer vision*** is closely related to *artificial intelligence* and formulates the problem of image understanding mostly as a search problem. Image understanding is carried out mainly on a high abstraction level trying to ascertain a mostly complete *symbolic* description of the scene. The visual processing is driven by hypotheses, primarily top-down.

In 1978, the VISIONS project [68] was one of the first projects applying a huge amount of object and domain specific assumptions to interpret scenes from visual input. The knowledge-based paradigm gained increasing popularity in the 1980s. Draper et al. [48] discuss different of these approaches. The ACRONYM system [19] used symbolic reasoning to aid static scene interpretation. It used stored models in form of "object graphs" and applied generalized cylinders to represent the visual appearance of objects. The ERNEST [116] framework uses semantic networks to reason on a symbolic level based on attributes assigned to objects.

But knowledge-based approaches had their limitations mostly caused by their strong believe in the reliable interpretation of perception to symbols [48]. Accordingly, researchers started to concentrate on the *appearance* of a scene as it is perceived by sensors of a vision system. Adelson and Bergen [2] answered the question regarding the "fundamentals of vision" by the *plenoptic function* which describes the scene in dependence of the viewers position and

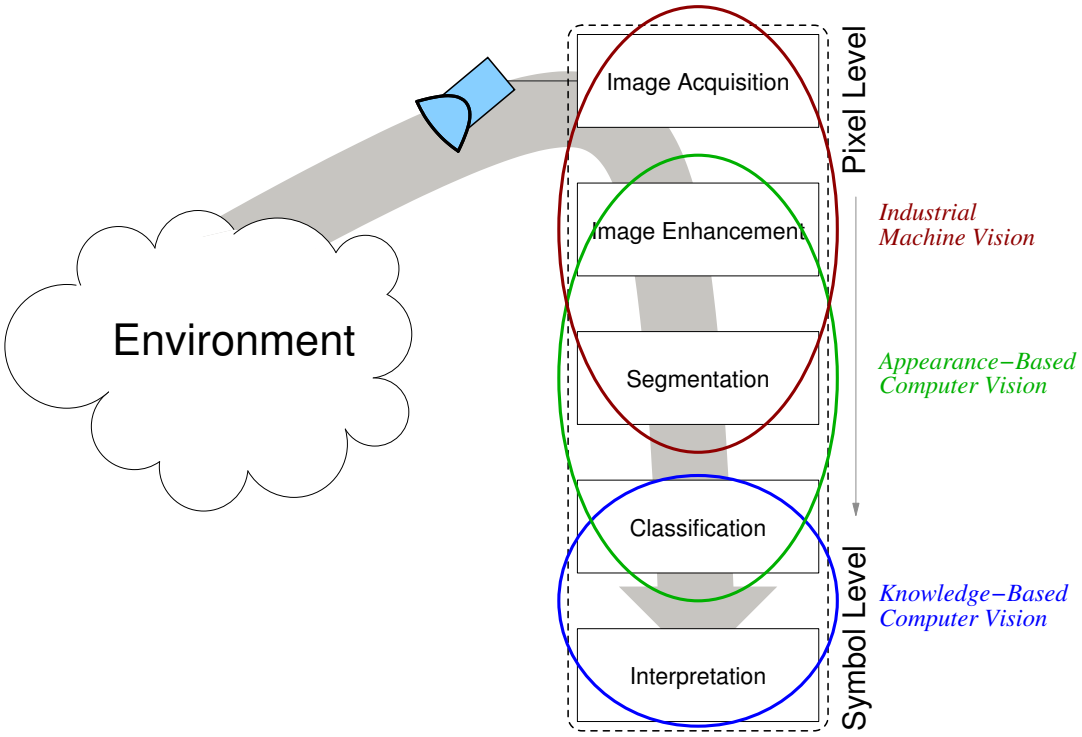


Figure 2.2: Different research emphasis on computer vision systems.

the scene illumination. The idea of ***appearance-based computer vision*** is to model or compensate variations in appearance by an appropriate model using either statistics or adequate features. Appearance-based computer vision is hence closely related to *pattern recognition* and mostly follows a bottom-up processing scheme.

Examples of appearance-based computer vision are manifold. Turk and Pentland [159] applied principal component analysis on face images to perform a classification of individuals. An approach taking into account also some explicit modeling is proposed by Cootes et al. [32] in terms of so-called "active appearance models" that have been applied to a wide domain of appearance-based vision ranging from medical analysis to face identification. Other, more recent approaches [106, 97], use local, mostly invariant features to account for the position-dependent character of the plenoptic function.

Summarizing, Fig. 2.2 provides a taxonomy of research focuses regarding the approaches and its underlying representations outlined so far. Considering computer vision systems as a composition of different components, we can identify *industrial computer vision* to be focused on image acquisition and segmentation mainly, for most reliable and accurate results [50]. *Knowledge-based computer vision* spends most effort on research on high-level reasoning and search strategies for interpretation and classification, while *appearance-based* approaches are most concerned with representation of image on the feature level.

Up to now, computer vision systems have been discussed as *open-loop* systems. As shown in Fig. 2.2, the environment is perceived "as is" and the system has no possibility to interact with it. Although a combination of top-down and bottom-up methods is applicable between different components of the system, the result of the computations has no effect on the environment and the system has no control about, what it visually acquires.

To close the loop is the major foundation of ***active vision systems***. Such a *closed-loop* system is sketched in Fig. 2.3. Contrasted to passive vision, the active approach is not

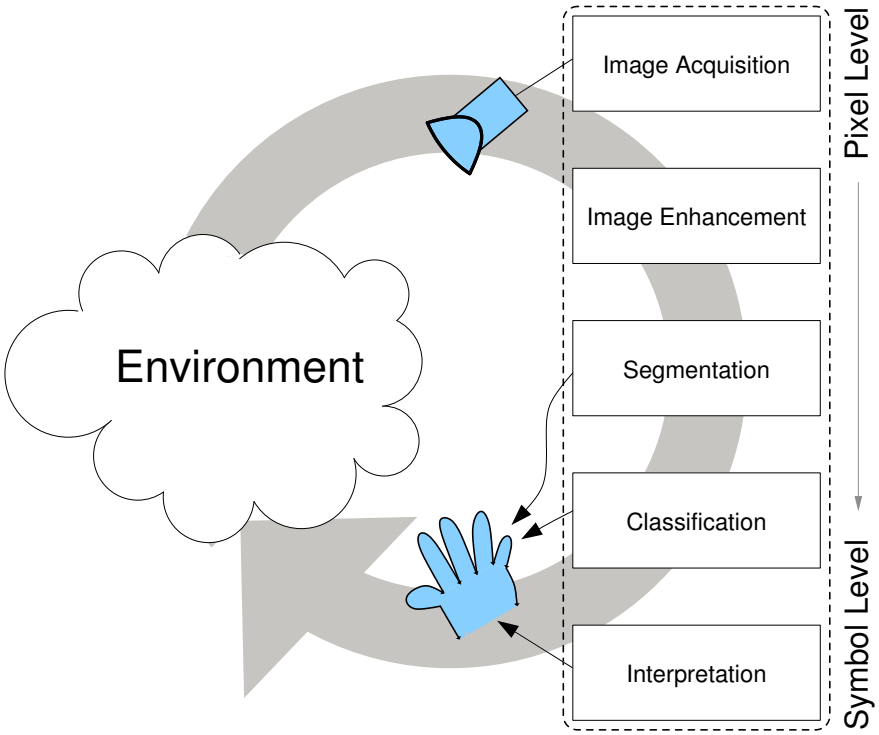


Figure 2.3: Active vision closes the loop.

restricted to the image(s) provided, but can affect the environment and by this means also its own perception. Pahlavan et al. [118] define active vision as follows:

> *An active visual system is a system which is able to manipulate its visual parameters in a controlled manner in order to extract useful data about the scene in time and space.*

Note, that active visual systems must inherently be *online* systems, that are situated in the environment.

According to this definition, interaction between the system and the environment is restricted to modifications of perceptual parameters like cameras positions, illumination, etc. Thus, such systems are active by means of an *active observer* [4] that not necessarily modifies the environment but its own perception. The natural progression of such active vision systems are *embodied* systems as will be described in the next chapter.

Artificial active vision often tries to mimic human capabilities with respect to eye movement and saccades, and aims to find appropriate representations for an actively acquired scene [128]. Applications utilizing active vision concepts range from robot navigation [42] over visual attention [17] in artificial agents to object recognition strategies [46].

2.2 Cognitive Vision Systems

Computer vision research started to "make computers see", almost like humans, but is still far away from this goal. On the one hand signal processing and pattern recognition developed methods to compute abstract representations of the content of images. On the other

hand artificial intelligence investigated on complex deliberative models of human intelligence and behavior. This scission between the pattern recognition and statistic oriented point of view and symbolic paradigms [21] had to be bridged in order to allow for flexible computer vision systems that are able to *see*, *reason*, and *act* in real world environments.

In the 1990s researchers began to take a more systemic view on the subject. Crowley and Christensen [37] considered *vision as process (VAP)* comprised of different components like active camera control, real-time image processing, tracking, perceptual grouping, and object recognition. They defined the general image interpretation task mainly as a problem of control and coordination of different independent and complementary components. VAP started as an robotic project in 1989, but evolved to a more general paradigm for computer vision.

A major foundation of the vision as process model is the role of spatial-temporal context. Vision is not bound to the current image only. VAP applies concepts of *active vision* and explicitly copes with the dynamics of vision in the interpretation process [103]. By this explicit contextual model, VAP borrows some concepts of *knowledge-based computer vision* for goal-driven control but also applies reactive schemes for camera control, for instance. VAP is placed between appearance- and knowledge-based vision systems in Fig. 2.1. It is built on quite strong assumptions making the system be more general applicable by means of its active vision approach. Under the term "Vision as Process" already quite complex vision system studying the interpretation of vision input fusing multiple cues have been developed, e.g. by Kittler et al. [85].

2.2.1 Definition

In the late 1990s a new paradigm of computer vision began to form consolidating the *vision as process* concept on the one hand, and being built on insights of cognitive science on the other: ***cognitive computer vision*** (CCV). While there is a great variety of definitions of *cognition* even within the disciplines of biology, psychology and computer science, all agree that humans are a prototypical cognitive system with an amazing, though effortlessly achieved performance. The goal of CCV is not to explain (human) cognition in detail, but let findings of cognition science, biology and psychology guide the design and development of ***cognitive vision systems (CVS)***.

As a consequence, only a limited notion of cognition is used according to Christensen [25]:

> *Cognition is [...] interpreted as "generation of knowledge on the basis of perception, reasoning, learning and prior models".*

This notion is rather simple compared to models of cognition in psychology or neurobiology [38], but is sufficient to motivate the following central aspects of CVS:

Active Cognition is considered to be an active process [25]. Instead of just monitoring its surroundings, a cognitive system is able to communicate and interact with its environment. It can control its perception in terms of an *active vision* system and explicitly act in the world.

Embodied In order to achieve activeness, a CVS must not be an external observer of the environment, but be a physical part of it and participate in its evolution. A CVS is always defined as a ***closed-loop*** vision system. Embodiment enables a CVS to interact in order to change the state of the environment which subsequently affects

its own perception. Figure 2.3 outlines the design of an embodied vision system. The perception-action cycle is closed by actuators that can manipulate the environment. Embodiment is not only a technical requirement for the construction of CVS, but is also considered as a requirement for the development of cognition as stated by Cruse [38].

Learning Acquired knowledge plays a crucial role in cognition. It allows to recognize things and situations seen before and to act accordingly. Granlund [61] argues, that CVS should inherently by *extendable*, which implies that the system should be able to deal with other situations as exactly the one it has been designed for. The process of acquiring and organizing knowledge about new situations, scenarios, and environments is referred to as *learning*. Learning is directly facilitated by embodiment. Cognitive beings in general learn throughout their whole life which motivates to design CVS with learning capabilities on all levels starting from simple senso-motoric learning up to the learning of new abstract concepts.

Situated The inflexibility of some computer vision systems is among others caused by their lack of *situation-awareness*. For beings with cognitive abilities it is most important to judge their perception with respect to the current situation.

Example 2.1: *Seeing a lion on the cinema screen, for instance, is usually considered much less dangerous than having the same visual perception in the middle of nowhere.*

Such contextual information together with acquired knowledge provide the necessary flexibility to adapt to new situations and to react adequately.

Contrasted to the VAP concept, the major enhancements of cognitive vision systems lie in their envisioned learning capabilities and demand for embodiment, in order to make cognitive systems more adaptive and flexible in their application.

Processing in a cognitive vision system does not follow a fixed path from images to symbols or vice versa. As in the VAP approach, different components are more flexibly combined. This is especially important, since CVS combine patterns of reactive and deliberative behaviors according to Granlund [61].

Hence, the mainly linear architecture shown in Fig. 2.3 is no longer appropriate and we have to outline a more general architecture as sketched in Fig. 2.4 that also allows asynchronous processing loops. Here, no explicit definition of the respective building blocks is given, but the functionality is realized by the interplay of the different components.

In order to meet the defined requirements, data and control flow are spanning all levels. As a CVS should be active and adaptive, it must allow higher level processes to reconfigure or trigger lower level modules to account for a situational change, for instance. Furthermore, results of different processing modules have to be fused, integrated and interpreted in order to give satisfactory results. As processing is not linear and the interplay of different modules is crucial for the functionality of CVS, the question of an appropriate underlying architecture gains focus. Accordingly, research on cognitive vision systems addresses the selection and implementation of appropriate algorithms as well as conceptual, cognitively motivated architectures to integrate these into *systems*. Such architectures are therefore itself a field of research. Our approach presented in this thesis based on a memory architecture is the subject of chapter 5.

Reconsidering some of the adjectives *attentive*, *cooperative*, *interactive*, and *adaptive* associated with cognitive assistance in chapter 1, we can identify their relations to CVS here: CVS are described as *active* systems. This activity can be realized by interaction between

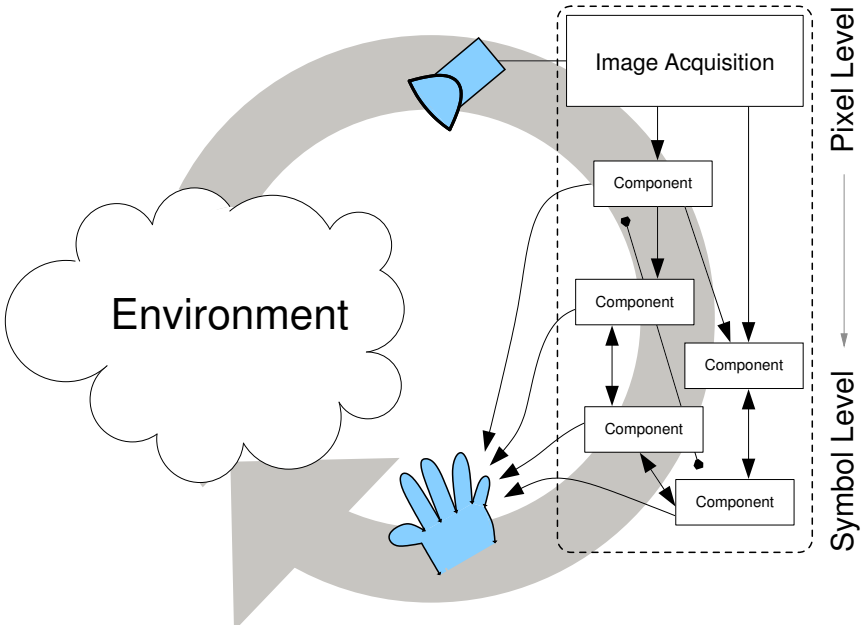


Figure 2.4: Outline of a general cognitive vision system.

the system and the user as outlined in chapter 1. Furthermore, to be *situated* is a prerequisite for *attentive* behaviors. And the ability to *learn* about new situations is crucial to allow *adaptivity*. Hence, the paradigm of Cognitive Vision is in particular appropriate for realizing a system for cognitive assistance.

Summarized, cognitive vision follows a rather pragmatic path in order to build computer vision systems by combining insights gained in different disciplines. The construction of *systems* regarding human principles as a model that influences the design gained focus by means of cognitive vision. Flexible data flows, designated continuous learning strategies, and effective acquisition and application of knowledge shall lead to more general applicable computer vision systems.

2.2.2 Related Work

The above outline of cognitive vision systems draws an ambitious picture. CVS research is still far away from developing a "general vision understanding system" that can compete with human abilities and it is still unclear how cognitive systems will finally proceed to that goal. But under the roof of cognitive vision, research on integrated computer vision systems gained increasing interest in the last years, which is reflected by related publications and also specifically targeted research projects considering cognitive systems as a central paradigm.

The term "*cognitive vision system*" is especially popular in the European scientific community. In other cultures similar concepts are summarized under different term, or even not considered as an own integral concept. Accordingly, Granlund [61] states that it is hard to determine what exactly constitutes cognitive vision systems, and especially where they start and where they end. In agreement with his statement, I try to have a closer look at some work in the following, that claim the term "cognitive vision" for itself, in terms of compliance to the attributes of CVS and the targeted scenarios.

As *embodiment* and being *situated* have been identified as crucial attributes of CVS, many

projects in CVS research are related to *robotics*. These projects use one or several robots as situated and embodied agents that interact with their environment and claim to be build on cognitive architectures.

One of the first projects, that published its insights under the umbrella of cognitive vision systems is CogVis. This project aimed to "*construct truly cognitive vision systems [...] in the context of an embodied agent*" [26]. Many of the considerations regarding cognitive vision systems have been an outcome of research conducted in this project, and in turn also provided inspiration for this work.

A successor project of CogVis aiming also at *cognitive assistance* is CoSy[27]. Different to the system envisioned in my work, the CoSy project aims to construct service robots to assist humans. CoSy nevertheless considers related cognitive paradigms as outlined here in terms of learning, combination of reactive and deliberative behaviors, and a focus on integration of sub-functions.

In another project, Granlund [60] presents an architecture to learn object properties using a combination of neural networks and symbolic processing in a robotic domain. His work can be considered as a cognitive vision system accounting for almost all attributes defined before. The architectural approach he outlines is rather integral. An explicit perception-action cycle for reactive behaviors is interfaced with a symbolic representation and interpretation for more complex behaviors.

A project focusing on a cognitive architecture is RobotCup [140]. This project studies questions of cognition using a humanoid robot. In contrary to the definition of cognitive vision given in this thesis which utilizes human cognition to establish design principles for computer vision systems, RobotCup also aims to gain insights on human cognition from the robotic architecture [105].

A major research area that considers principles of cognition for their conceptualization is *human-robot-interaction (HRI)*. Haasch et al. [64] present an architecture for a service robot that is designed for non-expert interaction in a most intuitive way. An overview of robotic systems that are capable of interaction considering mainly social cognition is given by Fong et al. [54], although they do not relate the implementation of the approaches to principles of cognition systems in particular.

But there are also other projects that claim the term "cognitive vision" for their work which do not fulfill all the definitions given above, or at least only provide a partial implementation of the basic assumptions of CVS. Most often the aspect of *embodiment* is only considered in a much weaker notion as originally proposed by cognitive vision research. By means of this weaker notion, systems for automatic surveillance, as for instance traffic analysis [6], and other more general image understanding systems are proposed as cognitive vision systems, even if their interaction with the environment is quite restricted. Maillot et al. [100], for instance, regard their three layer hierarchical knowledge-based image understanding approach as a cognitive vision system even though the attribute 'embodiment' is hard to assign convincingly. But nevertheless, their approach combines hybrid methods following cognitive principles that allows to subsume their work under the umbrella of cognitive vision.

The broader interest in cognitive vision is also reflected by increased and special funding on the subject. By the end of the year 2005 solely the European Community funds eight projects under the common term "cognitive systems". Some have been already touched by the above descriptions. The three largest ones set out to understand the cognitive development in humans (RobotCup) or focused on architectural issues (CoSy [27], JAST [76])

of either interacting (CoSy) or collaborating (JAST) robots. Besides these, some smaller projects focus on architectures for autonomous robot arms (COSPAL [36]), and mobile robots (MACS [99]).

Cognitive systems have also drawn attention worldwide but usually are termed differently. In Japan, the humanoid robotics projects dominate the development and adopt several aspects of cognitive systems especially in terms of embodiment and situated learning. The most notables works propose cognition as a joint development tool to build humanoid robots that cooperative more closely with humans. Emphasis of this research is on the design principle to develop more flexible and adaptive control structures in robotics. Embodiment is strongly considered here as a major principle to develop cognitive abilities and learning is advocated as best means to cope with the complexity of open environments.

In the USA ubiquitous computing constitutes a major topic for application of principles of cognitive vision systems. The most promising applications considering cognitive principles are smart rooms (e.g., Pentland [122]) on the one hand, which try to understand human activities. On the other hand imitation learning approaches (e.g., Breazeal et al. [18]) that are utilized to build artificial creatures, people can physically interact with and that even implement aspects of social embodiment, apply many principles of cognitive vision, too.

2.3 The Human in the Loop

We have seen so far that cognitive vision systems interact with their environment and several of its paradigms have been identified as being beneficial for assistance systems in particular.

A key attribute, taken a closer look at now, is *embodiment*, which has been introduced as a requirement for CVS. However, there are different possibilities to realize embodiment. Most often, embodiment of artificial systems is achieved by means of robots with manipulators and sensors that can interact with their environment [49]. However, even though there is considerable progress in the fields of mechatronics and robotics, machines that independently explore their environments are still in their infancy.

Ziemke [174] identifies different levels of embodiment in computer systems. Some of his definitions shall be introduced briefly:

Embodiment as structural coupling defines a very broad notion of embodiment that does not necessarily require a body, but only some coupling between the system and the environment, which holds also for software agents.

Physical embodiment explicitly demands for some physical instantiation. In its original definition this demand also holds for objects like tables and chairs which are undoubtable instances that lack any cognitive abilities. Ziemke [174] proposes to narrow the definition to the requirement of sensors and actuators.

Organismoid embodiment stands for physical bodies which at least to some degree have the same or similar form and sensorimotor capacities as living bodies.

Following these definitions, we propose to go even one step further and utilize the human body also as instrument for the embodiment of the artificial system itself. Certainly, the system is only indirectly embodied by this means. Hence, I introduce an intermediate level

of ***mediated embodiment*** between physical and organismoid embodiment as a foundation of the concepts presented in this work.

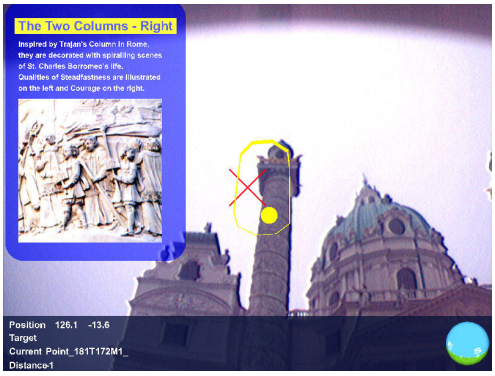
Reconsidering the outline of a cognitive vision system sketched in Fig. 2.4 the user of the system steps in for the manipulators and – partly – also for the perception of the system. We hence propose to integrate the ***human into the loop*** of the integrated system. As will be discussed later on, this idea fits well with requirements of cognitive assistance systems. But let us first discuss, how the user can be integrated in the processing loop effectively.

2.3.1 Augmented Reality and Shared Perception

From the perspective of the artificial cognitive system it is generally a drawback not to have direct control of the manipulators and the perception. When no direct control of the actuators and perception capabilities is possible due to the mediated embodiment of the system, *interaction* has to compensate for it. Accordingly, the system needs effective ways to communicate its knowledge and demands to the user, who presumably acts appropriate. Additionally, the system must be able to cope with the perception that is controlled by the user.

In this work, *vision* is considered as primary cue for perception and also for interaction. Thus, ***augmented reality (AR)*** provides a comfortable way to integrate the user in the processing loop in terms of visual perception and interaction. The idea of augmented reality is to enrich the user's visual perception of a real scene by generated visual information. Augmented reality techniques are widely applied to overlay scenes with additional information. This augmentation can range from simple textual annotation to a complex mixture of virtual and real world, including rendered virtual sub-scenes that perfectly fit in the real view of the user. Usually AR is used for either entertainment or to support users that carry out visually demanding tasks. As an example for the first application, an artificial tourist guide [130] based on AR is shown in Fig. 2.5. A typical example of virtual objects rendered

(a) A person wearing an augmented reality device.



(b) View through the video see-through AR device with augmentations.

Figure 2.5: An augmented reality system for navigation and tourist guidance. (© Vienna University of Technology [130])

in a natural scene are applications in architectural design [166] as can exemplarily be seen in Fig. 2.6(a). In this application scenarios, AR already left its infancy and a lot of research is conducted to embed virtual sub-scenes in the real world most realistically.

(a) An augmented reality system for architectural design (© ARTHUR-System, Fraunhofer Institute).

(b) Modern augmented reality *see-through* devices.

Figure 2.6: Augmented Reality.

Concentrating on the possibility to display information in the field of view of the user, AR provides a very efficient communication channel to articulate information to the user. AR requires special devices – usually kind of glasses – to augment the field of view of the user. Figure 2.6(b) shows some typical examples of AR devices. Generally, two different approaches can be distinguished. AR can be realized based on a *see-through* device. Such glasses are transparent and allow a minimal intrusive realization of augmented reality. AR devices following this approach are also termed "*optical*" approaches. Alternatively, AR can be implemented on a *video see-through* basis. In such systems the scene is captured by head-mounted cameras and redisplayed on small monitors mounted in front of the user's eyes. Using two cameras also allows stereoscopic visualization in the head-mounted displays. The latter has certain restrictions regarding the quality of the view on the scene but it has also a major advantage: The camera used for the video see-through can also be used as visual perception of the scene. Thus, using video see-through AR techniques easily implements a shared perception of the user and the system, especially enabling the required close interaction and knowledge exchange. The perception is under direct control of the user in this case which is in conformance to the idea that the user realizes the embodiment of the complete system. Thus, the human is really part of the system's processing loop.

Azuma [7] exhaustively discusses other pros and cons of the two different approaches. The major advantages of optical approaches are a better resolution and quality of the user's view, and a natural eye offset and field of view. In video approaches the eyes of the user are virtually moved to the position of the cameras which might not be at exactly the same position as the eyes naturally are.

As a consequence of the very tight coupling between the system and the user the video see-through realization has another benefit: The temporal and spatial synchronization between the system's and the user's perception is implicitly given without requiring an extra effort on calibration. To highlight a specific object in the field of view, for instance, the current perception captured by the cameras can be directly augmented and displayed in perfect temporal and positional alignment.

2.3.2 Benefits of the Human in the Loop

Considering the above mentioned concepts of the human in the loop paradigm, certain benefits for cognitive systems in general and assistance systems in particular can be identified. Attributes of assistance systems and cognitive systems have already been associated with each other in section 2.2. Here, I will outline how the human in the loop paradigm affects the implementation of some of these attributes.

situated & attentive It has been stated, that attentive behavior is coupled to situation-awareness of a cognitive system. As the system has the same visual perspective as the user, a model of ***shared attention*** is easy to realize. Although I agree with Kaplan and Haffner [81] who state that shared attention is not the same as simultaneous looking to a far extend, simultaneous looking is nevertheless considered as an important cue for (visual) shared attention between a machine and a human [113]. Thus, besides providing feedback in terms of information display, the deployed AR devices facilitate a shared perception of the system and the user. As seeing is the most prominent perceptual cue in the cognitive processing of humans, users of an AR assistance system usually look to where they act. They unconsciously focus objects they are interested in and that they are going to manipulate in order to complete a task. By means of this, the spatial and visual attention of the user is easily accessible to the system. Using video see-through techniques to realize augmented reality furthermore eases the determination of the visual attention of the user as she or he sees exactly the same as the system does.

But shared attention is not a one way. The system is not only able to recognize the user's attention quite easily, but can also attract her or his attention. Reconsidering the augmentation capabilities, the system can directly affect the user's view. This allows to direct the user's attention to certain places or positions in the scene. For instance, a flashing box around an object attracts attention very much. As a natural consequence a user would usually look at this object. Thus, the system gains indirect influence on its own perception. Recapitulating, the system is inherently situated in the scene together with its user. User and system can mutually influence their perception. In chapter 3.3.3 the role of a shared perception to determine situational contexts is further discussed.

cooperative & (inter-)active A cognitive assistant has to be most cooperative. As outlined before cooperation demands for the exchange of knowledge and information, and thus for *communication*. By communication we do not solely mean verbal communication using speech, but – even more – non-verbal cues like gesturing, gazing, and others more. By enabling the system to present visual information in the field of view of the user, we realize a very powerful communication interface for information flow from the system to the user. But displaying information does not only allow to present information, the cognitive system can furthermore use this interface to ask for information in order to facilitate learning capabilities.

The human in the loop paradigm does not only provide an efficient interface for communication; it also defines an implicit *context* for it. Reconsidering what has been said about attention and situation-awareness in the last paragraph, it becomes clear that a very specific shared context is available by means of shared perception. This inherently makes communication about the scene much more efficient. The *'What?'* and *'Where'* of communication contexts is much easier to determine from a shared perception than by an external observer.

adaptive & learning The role of knowledge exchange and learning capabilities has been stressed for cognitive assistance and cognitive vision systems as well. But how can shared perception and the human in the loop paradigm help to enhance learning capabilities? Generally, *supervised learning* and *unsupervised learning* have to be distinguished. While the latter does not require extensive interaction with the user, the first has usually a high demand for interaction and supervision. In supervised learning, an expert – in the given case, the interactive user – provides the expected results to any given input in order to teach a system. If the human is integrated into the loop by means of AR, she or he can interactively select appropriate perceptual prototypes and label it to teach the system with. Thus, besides joint attention and interaction capabilities shared perception has a direct benefit for supervised learning approaches in cognitive assistance systems.

2.4 Ego-Vision Systems

We so far have seen how the recent approach of cognitive systems opens up new vistas for the construction of flexible computer vision systems. And it has been shown which benefits the idea to integrate the human into the loop for collaboration and interaction has in general and especially for assistance systems in particular. For this combination of *cognitive vision systems* and shared visual perception by means of *mediated embodiment* I propose the term "***Ego-Vision Systems (EVS)***".

Let us dissect this novel term into its parts: The term "**ego**" accounts for the specific *perspective* these systems adopt by means of a shared perception. From a user's point of view, the system basically adopts her or his perspective in terms of perception and interpretation. A far-reaching equivalence between the system's and the user's perception is established, which facilitates close collaboration and shared attention. An ego-vision system perceives the scene in almost the same way as its user does. Ego-vision not only defines an extension in perception and interaction capabilities, but also imposes a major requirement. The system's perception shall be based exclusively on the user's perspective, which allows such-like systems to be basically wearable and not to be bound to a specific place. With computing hardware getting smaller and more powerful, this allows assistance systems to become really mobile. This fundamentally increases the value of such systems and paves the way for novel applications, even in outdoor scenarios and everyday usage.

The prominent role of "**vision**" has been stressed before. In EVS, visual cues play the major role, although multi-modal interaction is briefly touched on, too. The perception of the environment is almost exclusively bound to visual input. Therefore, head-mounted cameras constitute the main perceptual sensors in EVS. Nevertheless, they can be accompanied by additional, but also wearable sensors. But vision is not only used as main perceptual cue, but also to close the loop with the user as outlined before.

While several aspects of shared attention, shared perception, and knowledge exchange using visual cues can indeed be studied individually, the goal of EVS research is to let these findings converge into a *purposive* "**system**". Ego-vision is hence not an end in itself, but should help to develop such systems that especially allow to assist humans, and therefore make benefit of the special ego-vision perspective they take. It shall facilitate to develop systems that develop cognitive abilities. A conceptual architecture is needed as a basis for flexible interpretation and collaboration strategies. As the human is integrated into the loop, reactive behavior of the system is most important and also imposes demands on the conceptual architecture of EVS.

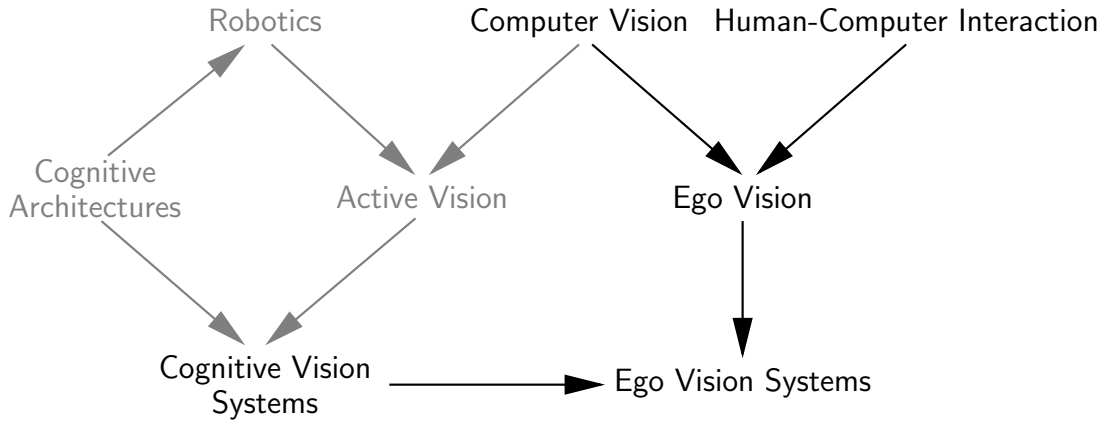


Figure 2.7: Taxonomy of Ego-Vision Systems.

Reconsidering the central ideas of ego-vision systems allows to identify a small taxonomy showing the relation of EVS to other fields in computer science sketched Fig. 2.7. Ego-vision generally picks up concepts of *computer vision* and of *human-computer-interaction*, reflecting the perceptual and articulatory capabilities. It can be seen on the same level as *active vision* approaches, as it follows similar goals. Vision can generally be regarded as being indirectly *active* by means of the *mediated embodiment*. By adopting principles of cognitive vision systems, *ego-vision systems* are defined. There is also a relation to cognitive robotics by means of active vision and cognitive architectures in general.

2.4.1 Related Work

The first researcher, who put the human in the loop using wearable cameras and augmented reality is probably Steve Mann, who's earliest works date back in the 1990s [101, 153]. He started to use wearable cameras in everyday life situations and augmented the scene on user's request, in order to display additional information about objects in the field of view. AR systems that present a virtually augmented real world to the user are wide-spread. Reitmayr and Schmalstieg [130] used augmented reality to realize an information system for tourist guidance in cities. Also in assistance, AR has been used before in order to support a user in pre-defined tasks [88]. Research is furthermore conducted on intuitive interfaces in augmented reality scenarios. Starner et al. [148] played "patrol", an interactive game, using a head mounted camera that recognizes gestures. Kölsch and Turk [89] presented an alternative technique to track hands accurately for interface based interaction in augmented reality environments.

I already stressed the role of attention in ego-vision system and how shared perception can facilitates its articulation to the system. Pilu [126] provides a very comprehensive overview on different aspects of attention for wearable cameras. Nagai et al. [113] did several works on joint attention between artificial agents and humans based on visual cues, but used external cameras that observe the human in front of a robot. An external perspective is also taken by Stiefelhagen et al. [149], who proposed a multimodal model for attention combining sound and gaze. Works on attention recognition using wearable camera is rather rare. Nakamura et al. [114] realized visual attention recognition on the basis of motion patterns in the image sequences captured from a head-mounted camera. They focus their work on the determination of what they call either *active*, which means actively gazing at some part of the scene, or *passive* attention, that covers a longer period in time. Nakamura

et al. [114] use their method to structure recorded video sequences to allow fast access to relevant parts.

The idea to capture the visual input of a user and make it accessible in a structured way also drives the work conducted in the DyPERS project [142]. Their developed wearable system is used to acquired and retrieve so-called "media-memories". The combination of augmented reality and interactive computer vision makes the project most related to the one outlined in this thesis. However, Schiele et al. [142] focus on low-level representations; the information presentation provided by the system is restricted to replay of recorded video-audio episodes.

Another interesting application of wearable augment reality assistance is published by Jebara et al. [77]. They applied geometrical analysis of balls on a billiards table in order to assist a player in planning and aiming for pool billiard games. The scenario is very specific and a wider field of application is not targeted.

In many systems, the role of *context* is stressed in order to provide appropriate information at a given time. Although the term "context" is widely and diversely used in computer science and linguistics, Dey [45] defines context as "any information that can be used to characterize *situations*" and thus relates it to *situation-awareness*. Chen and Kotz [23] give a survey on different projects coping with context-aware computing of mostly wearable devices. Spatial context, for instance, is considered in the work of Wagner and Klinker [164]. Lieberman and Selker [94] give a more general survey of context-sensitive processing in computer systems.

2.4.2 Challenges

As we have focused on the benefits of integrating the human in the loop by means of ego-vision systems so far, we also have to cope with some challenges, too. Most of these challenges arise from the fact, that EVS abstain from any external sensors.

- The visual perception of the scene is realized exclusively using head mounted cameras. Every movement the user performs has a direct impact on the system's input and this movement is basically not restricted. A user can turn and move the head in any arbitrary fashion. Thus, an ego-vision system has to face blurred images, fast movements in the image, and unpredictable changes in the field of view.

- The EVS has no direct control on the cameras, and in turn of its visual perception. The human can look arbitrarily around. Due to the limitations to head-mounted cameras, some events occurring outside the current field of view might thus be overlooked by the system. Hence, an ego-vision system must be able to cope also with changes in the environment that it have not directly been observed.

- In contrast to humans, who can make use of foveal and peripheral seeing, a trade-off for the perception of the ego-vision system has to be found. The narrower the field of view, the better resolution is available, but the less contextual information is accessible in the current field of view. Especially in conjunction with *video see-through*, this trade-off also directly affects the user's perceptions. Note, that a more focused field of view increases other challenges like motion artefacts and fast movements in the video stream.

- In contrast to fixed cameras, the problem of registration and localization arises in

ego-vision setups. Even more, the problem has to be solved anew for every frame, since the motion is generally arbitrary.

In conjunction with the restricted field of view of an ego-vision system, the registration problem becomes even tremendous. If the system is not able to remember the position of an object in the scene, the discrimination of different objects that are not always together in the current field of view is difficult. It is therefore especially demanding to determine the view context of current perception.

All these challenges have a strong implication on the selection of appropriate computer vision algorithms in order to construct a cognitive assistance system based on the concepts of ego-vision. Subsuming, the central challenges arise from ego-vision perception, and the strong coupling between the system and the user. Accordingly, the following chapters will focus on algorithms that are suitable for perception (chapter 3) and interaction (4).

2.5 Purposive Vision and Scenario-driven Research

In contrast to the definition of *general vision* given in the introduction of this chapter, Aloimonos [4] proposed a more application oriented perspective on computer vision termed ***purposive vision***. Instead of targeting at the whole reconstruction and visual interpretation of a scene, which anyway is impossible, the purposive paradigm calls for a partial and opportunistic visual reconstruction considering a specific scenario.

It thus paves the way to ***scenario-driven research*** in computer vision systems. In order to study complex vision systems, one or several tasks or scenarios need to be specified, that provide the test-bed for the developed system. As general vision is still considered as almost impossible and also impossible to be evaluated, well-defined scenarios provide the means to assess the quality of different approaches of integrated system.

Furthermore, ego-vision systems inherently involve the user as part of the system and thus have an emphasis on human-computer-interaction. In turn, evaluation of integrated systems must be conducted in terms of user studies, considering also usability aspects. These also demand for well-defined scenarios to give meaningful results. In this work, this view is adopted to assess the developed ego-vision system with respect to defined scenarios and case studies.

2.6 The VAMPIRE Assistance System

Developing and investigating cognitive assistance systems or cognitive vision systems is not a task that can be accomplished by only one researcher. As the word “system” already suggests, and as has also been stated in the previous sections, different modules have to play together in order to function as a complete, purposeful system. Most of these different functionalities are scientific challenges of their own. In consequence, the contributions presented in this work have to be view at in the context of a larger project. Only by cooperation with other researchers, it became possible to setup an integrated system as it is presented in the following chapters.

The project, this work contributed to, is VAMPIRE. VAMPIRE is a research project on cognitive computer vision funded by the European Union. According to the project's objectives [154] the goal is to develop cognitive vision systems that are able to understand

what they see based on what they have previously memorized. In this sense, research focused on **V**isual **A**ctive **M**emory **P**rocesses. Another important goal is to develop advanced techniques for **I**nteractive **RE**trieval and interactive learning. One of the two main scenarios in VAMPIRE is concerned with cognitive assistance functionalities. A wearable augmented reality system is realized to provide the user with cognitive assistance functions while she or he is acting in an indoor environment. In short, the aim of VAMPIRE is to proceed towards memory prosthetic devices or some kind of *memory spectacles* that memorize objects, events and actions on different representational levels. This term subsumes very well the two major contributions to the assistance scenario in the project. The whole system is to be built based on a *visual active memory*, that can acquire and store knowledge about scenes, task, and events. This memory is on the one hand a conceptual architecture in terms of representation and data fusion and on the other hand a system integration concept that facilitates component coordination and has been published as the *Active Memory Infrastructure (AMI)* by Wrede et al. [170]. The VAMPIRE system is realized as an ego-vision system that takes the user's perspective. Hence, to realize this system, a hardware setup is necessary that allows to integrate the user into the loop. In VAMPIRE such a hardware device has been developed by a project partner as a prototype [145]. I will describe this so-called AR gear in Sec. 4.1. For the prototype of the assistance system, we focused on the memorization of objects and manipulative actions in defined scenarios.

VAMPIRE
Visual Active Memory Processes and Interactive REtrieval

Figure 2.8: The VAMPIRE logo.

The VAMPIRE project was not only concerned with this assistance scenario, but furthermore studied the idea of visual active memories on broader scale. A rather opposite scenario building on similar paradigms as the one outlined in this work tackles the challenge of automatic video annotation in sports videos [104, 28]. This application is however not in the focus of this thesis, although general ideas of an visual active memory are picked-up in both scenarios.

2.7 Summary and Contribution

This chapter has introduced fundamentals of *Cognitive (Vision) Systems* and their extension towards *Ego-Vision Systems (EVS)*. Cognitive vision has been introduced as a quite recently developed paradigm bringing together different disciplines in computer science to construct vision systems that are embodied and situated, and that develop learning and adaptation capabilities. After a short survey on the history of computer vision in general, I gave an overview on the major concepts of cognitive vision systems and discussed how these fit to the envisioned goals of assistance systems.

I defined ego-vision systems as systems that integrate the human in the processing loop and that implement *mediated embodiment* by means of a shared perception and close interaction using augmented reality techniques. The relations between assistance systems and the ideas of ego-vision systems have been discussed. The major benefits and challenges of EVS have been outlined and I discussed the resulting implications on perception and interaction capabilities in such systems. Furthermore, I presented the VAMPIRE project as the context in which the concepts and solutions presented in this thesis have been developed and evaluated. By means of scenario-driven research, the approaches discussed in the following will pick up the paradigms introduced in this chapter and converge into a cognitive ego-vision system for interactive assistance in dedicated scenarios.

3 Perception in Ego-Vision Systems

What is madness? To have erroneous perceptions and to reason correctly from them.
[Voltaire, 1694-1778]

As laid out in the previous chapters, perception is fundamental for *cognitive assistance systems (CAS)*. Having a "blind" system, will neither allow it to be *attentive*, nor to develop *cooperative*, *interactive*, or *adaptive* capabilities as postulated in chapter 1. In order to actively supervise a user and to provide situation-aware assistance, perceptual abilities play a major role. But also for ego-vision systems in general, the question of adequate perception has been identified as a challenging task. Accordingly, this chapter is devoted to solutions targeting at the most pressing problems regarding perception from an ego-vision perspective.

In EVS, perception is basically local in terms of time and space. The current visual perception of the system is a two-dimensional image at a given moment in time. But the goal is to establish a most complete spatial model of the scene, and also to perceive dynamic activities and events occurring in the environment in order to be able to understand what is going on and to act accordingly. Put in one statement it can be said that ego-vision systems have a *local* (two-dimensional) perception that should be considered in a *global* (four-dimensional) spatial-temporal context. The perception and its contextual embedding in space and time is subject to this chapter.

But what should at all be perceived in an ego-vision system that is suited to provide assistance? We focus on assistance applications that deal with different *objects* and their manipulation or application in order to accomplish certain tasks. But in contrast to offline scene or image understanding systems, ego-vision systems are applied in dynamic and interactive scenarios, so perception must likewise not be static. Objects are manipulated, used, and moved around. Thus, the environment continuously undergoes modifications. *Events* and *actions* occurring in the environment are crucial for assistance systems in particular, as usually some kind of manipulation of the environment is the goal of a task that a user is assisted in.

As we are discussing *online* systems, reactivity is supplementary to the other challenges of EVS introduced in the previous chapter. An ego-vision system integrates the user in the processing loop, and thus must be able to perform in compatible reactivity with a human. Accordingly, all algorithms discussed in the following operate in (soft) real-time[1] allowing reactive processing cycles.

This chapter will present solutions accounted for the outlined challenges of EVS. In particular these are

[1] With (soft) real-time I denote a demand for reactiveness, rather than a formal definition of real-time.

Figure 3.1: Example view in an office environment.

- a compensation for the limited field of view by means of mosaicing techniques that implement a pictorial memory,
- acquisition of 3D coordinates and definition of spatial contexts using a real-time pose tracking algorithm,
- an integrated fast learning object recognition approach to achieve a certain level of domain independence,
- and a visual tracking application to account for moving objects and moving cameras.

Note, that most of the presented solutions utilize a combination of recent algorithms that have been especially adopted to account for the needs of ego-vision systems, in particular. Whenever the underlying algorithms are applied in original means, they are only briefly described in this chapter, and the reader is referred to relevant literature to gather more detailed information. Consequently, I will focus on approaches, I have contributed to significantly. For others, I discuss their appropriateness for ego-vision perception. The focus of this thesis in general and this chapter in particular is therefore, to present a convincing selection and combination of computer vision algorithms that are suited for the application in ego-vision systems.

3.1 Mosaics for a Compact Pictorial Scene Representation

The limited field of view has been identified as a challenge for ego-vision perception and has been discussed as a subject of *local* perception in the introduction of this chapter. Only a small portion of the complete scene is visible at a specific moment in time. But it can be observed, that the user performs mostly arbitrary movement, during which she or he scans almost the complete scene, or at least the relevant parts in the course of time. Hence, more and more information about the scene can be gathered, and the system can obtain a *global* pictorial representation of the scene, or at least of all the parts the user has looked at.

Fig. 3.1 shows a view of an office scenario from an ego-vision perspective, which covers a part of the desk. This image has been taken from a sequence in which a user looks around in

the whole office. The most natural representation of all acquired pictorial data is probably a video, that contains all frames captured by the camera(s) while scanning the environment. But utilizing a video as a representation of the scene has several drawbacks, since

- ▷ it consumes huge amounts of memory capacity, although video compression algorithms can be applied to account for this problem at least to some extend,
- ▷ the representation is very redundant, as most parts of the scene do not change from one frame to another,
- ▷ there is no spatial relation between the different views or frames, respectively, and
- ▷ large parts of the scene (e.g., larger objects) are never represented as a whole, making their recognition or analysis a very complicated task.

By means of *mosaicing*, a solution is proposed that accounts for the challenge of the restricted field of view on a pictorial level, avoiding the disadvantages of a video representation. In the following, it is outlined how general mosaicing is adopted for ego-vision systems. Further details regarding the implementation of the approach can be found in [57, 58].

3.1.1 Mosaicing Basics

When capturing images at a rather fast rate while moving a camera, successive images within such a sequence usually overlap by a large amount depending on frame rate and camera dynamics. In order to implement a compact representation of the scene, these redundancies should be avoided. But exactly these redundancies can be utilized in order to construct *mosaics*. In essence, mosaicing describes the process of collecting images, compute the (spatial) transformations between them, and integrated them into one image. By choosing a reference image (usually the first one of a sequence), all other images can be warped to this reference coordinate system using the computed transformations and merged together to one mosaic image. That is why an area of the scene which appears in different frames is only represented once in the mosaic. The elimination of redundancy provides a significant reduction in the total amount of data needed to represent different views of a scene.

The general two-stage process of mosaicing is shown in Fig. 3.2. First, the transformation $\boldsymbol{T}$ of each image $\boldsymbol{I_t}$ to the reference image $\boldsymbol{I_{ref}}$ coordinate system is computed, and second, the different pixels are integrated into the current mosaic $\boldsymbol{M_t}$.

Transformation Different images are generally taken from different view points regarding camera orientations and positions. Thus, successive images of video sequences are captured with respect to different coordinate systems. That is why for each image of the sequence the transformation to a reference coordinate system has to be recovered. In fact, the computation of the warping function is the major problem when creating mosaics and different methods can be applied here. A discussion regarding different transformation schemes is provided by Gorges [57].

Pixel Integration After recovering the parameters of the transformation, the respective image is warped to the reference coordinate system. As a consequence, all images have a common coordinate system and the question arises how to integrate the corresponding

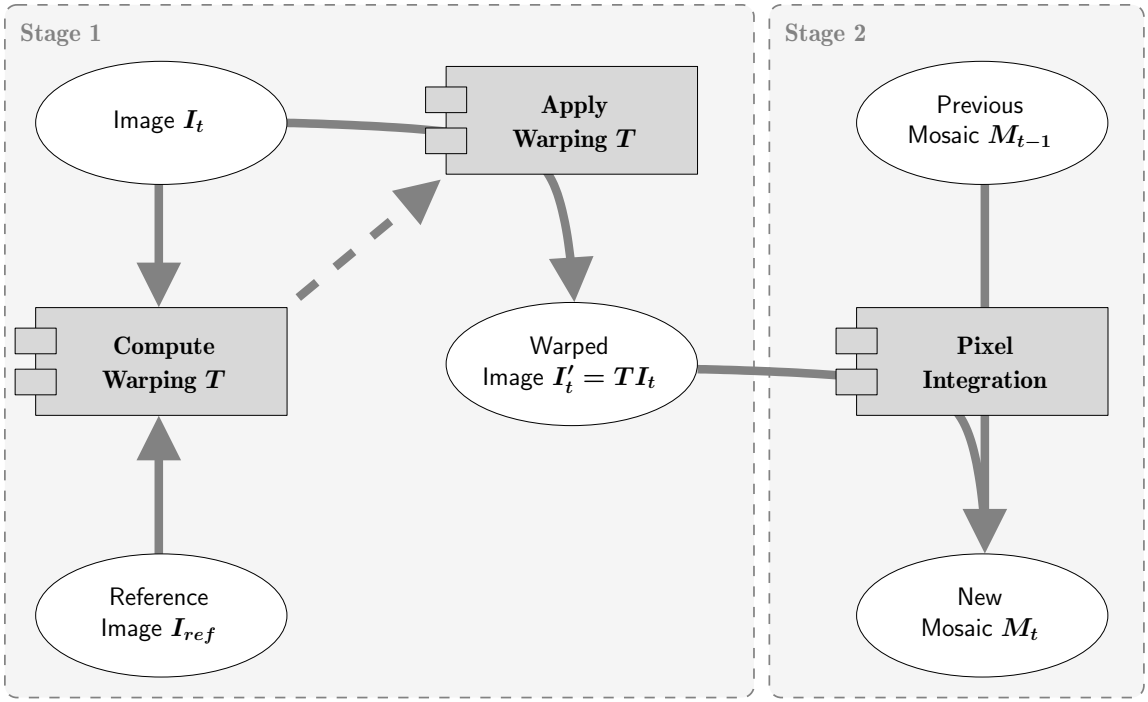


Figure 3.2: Mosaicing as a two-stage process

pixels into one mosaic. Here, different schemes are applicable: It is possible to either take the most recent value of each pixel, overwriting previously captured views, or to perform some kind of averaging to compensate errors and noise. Finally, the firstly assigned pixel value can be retained all the time without considering any newly assigned pixels. The concrete selection of the integration functions depends on the task, and on the available computational power. An averaging scheme usually gives most convincing results, but for the costs of higher computational costs.

Mosaics for Ego-Vision Perception

Mosaicing techniques are nowadays already used in various different applications, even though the common basis is always to represent a sequence of images of a given scene in one image. Thus, mosaicing provides a compact, non-redundant representation of visual information, accumulating spatial different views of the same scene. Besides the compression benefits from avoiding redundancy in mosaics, the larger field of view of the integrated mosaic image serves as a better representation of the scene than the single image data as it covers a larger spatial context.

But almost all recent mosaicing techniques have a restriction which make them inapplicable in ego-vision systems: They apply a restriction regarding the movement of the camera or the scene. This problem is due to the dimension reduction that results from the mapping of the three-dimensional world into one two-dimensional image. The general transformation

(a) (b)

Figure 3.3: Two images taken from the COIL-20 database [115] illustrating two different views of the same three-dimensional object.

of a three-dimensional point $\vec{\boldsymbol{X}} = (\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})^{\boldsymbol{T}}$ in the scene to a two-dimensional point $\vec{\boldsymbol{x}}$

$$\begin{aligned} \vec{\boldsymbol{x}} &= \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{\lambda} \end{pmatrix} = \boldsymbol{P} \begin{pmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \boldsymbol{Z} \\ \boldsymbol{1} \end{pmatrix}, \quad \boldsymbol{P} = \boldsymbol{C}[\boldsymbol{R}|\vec{\boldsymbol{t}}] \\ \boldsymbol{R} &= \begin{pmatrix} \boldsymbol{R_{XX}} & \boldsymbol{R_{XY}} & \boldsymbol{R_{XZ}} \\ \boldsymbol{R_{YX}} & \boldsymbol{R_{YY}} & \boldsymbol{R_{YZ}} \\ \boldsymbol{R_{ZX}} & \boldsymbol{R_{ZY}} & \boldsymbol{R_{ZZ}} \end{pmatrix} \\ \vec{\boldsymbol{t}} &= \begin{pmatrix} \boldsymbol{t_X} \\ \boldsymbol{t_Y} \\ \boldsymbol{t_Z} \end{pmatrix} \end{aligned} \tag{3.1}$$

in an image can be described by a 3x4 projection matrix $\boldsymbol{P}$ in homogeneous coordinates. Projection in homogeneous coordinates is well-defined up to scaling parameter $\boldsymbol{\lambda}$. W.l.o.g[2] $\vec{\boldsymbol{x}}$ is normalized with respect to $\boldsymbol{\lambda}$ so that the third coordinate of $\vec{\boldsymbol{x}}$ equals $\boldsymbol{1}$.

$\boldsymbol{C}$ denotes the camera matrix comprised of intrinsic camera parameters like focal length, skew, and distortions. In the following discussion, $\boldsymbol{C}$ is not relevant and the described method needs no explicit calibration to obtain these intrinsic parameters. $\boldsymbol{R}$ and $\vec{\boldsymbol{t}}$ denote the rotational and translational part of the transformation, respectively. Note, that all following equations assume homogeneous coordinates in order to describe projective transformations. Details regarding projective transformations in computer vision and camera models can be found in [51] in general, and a more detailed discussion in relation to mosaicing is provided by Gorges [57]. But what directly can been seen from Eq. 3.1 is, that the transformation is not bijective since $\boldsymbol{P}$ is a $\boldsymbol{3 \times 4}$ matrix, and thus the 2D representation of a scene is not adequate to represent a general 3D scene consistently as a whole. An example illustrates the problem:

Example 3.1: *If you look at the piggy bank from one side it might look as shown in Fig. 3.3(a). But it looks rather different from the opposite side (3.3(b)). Parts of the object, that have been disguised in the first view are now visible and vice versa. It is not possible to integrate these two different views caused by the three-dimensional body of the object into one two-dimensional geometrically correct image in euclidean space.*

[2] without loss of generality

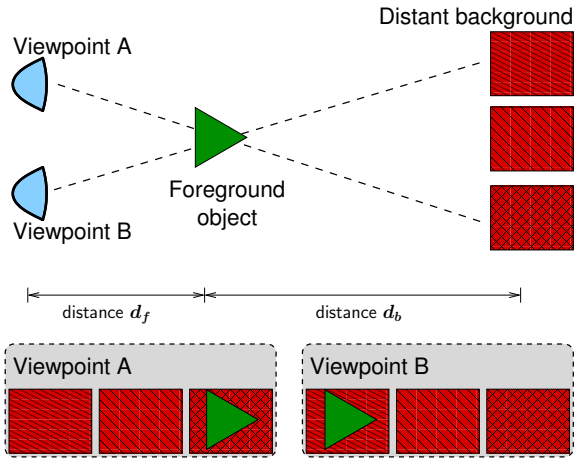


Figure 3.4: Parallax effect: The foreground object appears to be in front of different backgrounds depending on the position of the observer.

Another quite similar problem is the occurrence of parallax effects as soon as the camera is moving arbitrarily. Parallax describes the relative displacement of an object with respect to its background as seen from different point of views as sketched in Fig. 3.4. When translating a camera sidewise in a three-dimensional scene, each part of the scene will move in a different relative speed with respect to each other depending on the distance to the camera and causes overlaps as soon as the camera center is moved. Therefore, the construction of only a single mosaic of the scene will not succeed.

But there are constellations where the projection of 3D points can be simplified. If the problem can be reduced to 2D to 2D subspace mapping, a bijective solution is possible. The simplest solution is applied rather often in mosaicing research. If the motion of the camera is controlled and restricted to rotation about the optical center and to zooming, the translation $\vec{t}$ can be disregarded, leaving the transformation $\boldsymbol{P}$ as a simple $\mathbf{3} \times \mathbf{3}$ transformation

$$\begin{aligned}\begin{pmatrix} x \\ y \\ \lambda \end{pmatrix} &= C \begin{pmatrix} R_{XX} & R_{XY} & R_{XZ} & 0 \\ R_{YX} & R_{YY} & R_{YZ} & 0 \\ R_{ZX} & R_{ZY} & R_{ZZ} & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \\ &= C \begin{pmatrix} R_{XX} & R_{XY} & R_{XZ} \\ R_{YX} & R_{YY} & R_{YZ} \\ R_{ZX} & R_{ZY} & R_{ZZ} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = M\vec{X} \,. \end{aligned} \tag{3.2}$$

In this case the transformation between two images is well-defined. Consider a point $\vec{X}$ in the scene and two different projections M, M'. For each pixel $\vec{x}$ in the first image the corresponding pixel $\vec{x}'$ in the second is exactly determined by

$$\begin{aligned} \vec{x} &= C[R|0]\vec{X} = M\vec{X} \\ \vec{x}' &= C'[R'|0]\vec{X} = M'\vec{X} \\ \vec{x}' &= M'M^{-1}\vec{x} \end{aligned} \tag{3.3}$$

Suchlike mosaics are known as panoramic mosaics and are quite famous in digital photography nowadays.

But this restriction certainly does not meet with the requirement of ego-vision perception. It is a major assumption in EVS, that the camera movement cannot be restricted in any way, as the user directly controls the position and orientation of the camera. Therefore, another solution has to be found to utilize mosaics as a compact pictorial scene representation in ego-vision systems.

Taking a closer look at the geometric foundations of mosaicing unveils another solution to the problem. Avoiding translation of the camera is not the only possibility to achieve a dimension reduction in the projective transformation. It is also possible to create mosaics of planar sub-scenes. If the observed scene is planar, we can w.l.o.g. set $\boldsymbol{Z = 0}$ for each scene point $\boldsymbol{(X, Y, Z)}$ lying on the particular plane $\boldsymbol{\pi}$. Then, the projection $\boldsymbol{P}$ in equation 3.1 reduces to

$$\begin{aligned}\begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{\lambda} \end{pmatrix} &= \boldsymbol{C} \begin{pmatrix} \boldsymbol{R_{XX}} & \boldsymbol{R_{XY}} & \boldsymbol{R_{XZ}} & \boldsymbol{t_X} \\ \boldsymbol{R_{YX}} & \boldsymbol{R_{YY}} & \boldsymbol{R_{YZ}} & \boldsymbol{t_Y} \\ \boldsymbol{R_{ZX}} & \boldsymbol{R_{ZY}} & \boldsymbol{R_{ZZ}} & \boldsymbol{t_Z} \end{pmatrix} \begin{pmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \boldsymbol{0} \\ \boldsymbol{1} \end{pmatrix} \\ &= \boldsymbol{C} \begin{pmatrix} \boldsymbol{R_{XX}} & \boldsymbol{R_{XY}} & \boldsymbol{t_X} \\ \boldsymbol{R_{YX}} & \boldsymbol{R_{YY}} & \boldsymbol{t_Y} \\ \boldsymbol{R_{ZX}} & \boldsymbol{R_{ZY}} & \boldsymbol{t_Z} \end{pmatrix} \begin{pmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \boldsymbol{1} \end{pmatrix} = \boldsymbol{M\vec{X}_\pi} \end{aligned} \tag{3.4}$$

for a $\boldsymbol{3 \times 3}$ matrix $\boldsymbol{M}$ and a point $\boldsymbol{\vec{X}_\pi}$ on a plane $\boldsymbol{\pi}$. Hence, the similar simplification as for the motion restriction of the camera is achieved by the limitation to planar sub-scenes. The transformation between two images taken of the same planar sub-scene is well-defined again by

$$\begin{aligned} \boldsymbol{\vec{x}} &= \boldsymbol{M\vec{X}_\pi} \\ \boldsymbol{\vec{x}'} &= \boldsymbol{M'\vec{X}_\pi} \\ \boldsymbol{\vec{x}'} &= \boldsymbol{M'M^{-1}\vec{x} = H\vec{x}} \; . \end{aligned} \tag{3.5}$$

It should be noticed here that M must be invertible, which is not given, if the camera is itself positioned on the plane. But this is only a theoretical issue, as the plane would not be detected as plane anymore. This is due to the fact, that planes will be constructed and tracked in a data-driven way from detected points lying on the respective plane, as will be shown later.

According to the considerations outlined before, we can indeed generate individual, independent mosaics for planar sub-scenes, instead of restricting the movement of the camera. The transformation that describes the relation between two images of the same scene is called the ***homography H***. Such homographies play a major role in mosaicing, since a homography $\boldsymbol{H}$ determines already the warping transformation $\boldsymbol{T}$ between two (successive) images in a mosaic as shown in Fig. 3.2.

Related Work

Before we step into the description of the realization of the system, a brief review of recent mosaicing related literature shall be given. A lot of research has been done on applications of Mosaicing [74, 73] and improving their performance [176, 120]. Möller et al. [107] especially focus on the distinction between dynamic and static parts of the scene, which is an interesting extension to the work presented here. But all these approaches mainly focused on the conventional mosaicing method rather than on the restrictions. They provide no solution to the occurrence of parallax effects and apply restrictions on the camera movement.

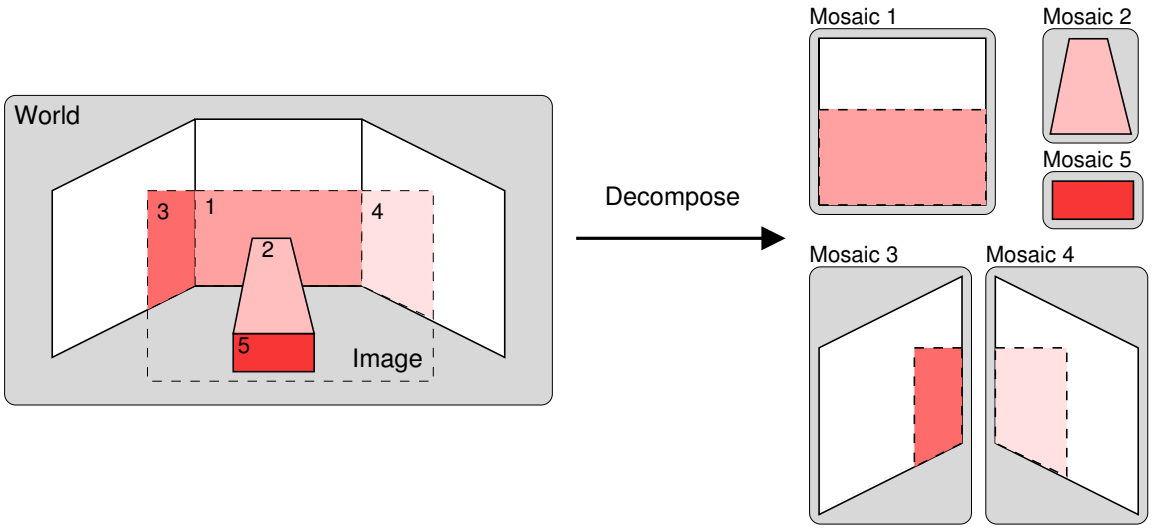


Figure 3.5: Scene decomposition into planar sub-scenes.

In order to overcome the restrictions for mosaicing, mosaics with parallax and layers with parallax were introduced by Kumar et al. [91] for aerial images. In this case, additional information about the 3D structure is stored to take account for parallax and to make the construction of mosaic images more robust. Another approach [121] tries to present mosaicing as a progress of collecting strips to overcome most restrictions. The strip collection copes with the effects of parallax by generating dense intermediate views, but is still restricted to controlled translational parts in the motion of the camera. These approaches still only work for their specific applications and do not overcome the restriction in general.

Baker et al.[10] describe an approach to represent a scene as a collection of planar layers calculated from depth maps. But in contrast to the algorithm presented in this thesis, the focus is mainly on approximating the 3D structure of the scene than on actually generating mosaics.

3.1.2 Mosaics of Planar Sub-Scenes

As a consequence of the above examinations it can be concluded, that it is obviously not possible to generate one consistent mosaic from the visual input in an ego-vision system. But although the motion of the camera cannot be restricted in such setups, partial mosaics of planar sub-scenes can be computed as proved by Eq. 3.4. Thus, the basic idea to make mosaicing techniques applicable in ego-vision systems is to compute individual mosaics for planar sub-scenes as illustrated in Fig. 3.5.

The central assumption for this idea to be convincing is, that the scene must be comprised of planar sub-scenes. This assumption appears very strong at a first glance since many objects in the real world lack of any planar faces, like balls, bowls, and many others more. But in fact, planarity is dependent on the distance a plane has to the observer, or the camera, respectively. Parallax effects are mostly negligible, if the distance between the observer and the foreground object $\boldsymbol{d_f}$ in Fig. 3.4 is much larger than the distance $\boldsymbol{d_b}$ between the foreground object and the background. The same holds for curvatures as long as they are relatively small with respect to the viewing distance.

Fortunately, the assumption of planar sub-scenes holds rather well especially in man-made

environments like houses, cities, and so on.

Example 3.2: *Many of the assistance scenarios, one can think of, have to deal with objects lying on a table. A table is a planar object in general. And furthermore objects like books, tools, etc. lying on the table appear almost planar considered separately, presumed the point of view is not changed completely, e.g., the user is walking around the table to view the scene from the opposite side. The front of a book lying on a table can visually be considered as part of the table surface, since the distance* $\boldsymbol{d_b}$ *is usually much smaller than the distance* $\boldsymbol{d_f}$ *if viewed at from a common point of view.*

Thus, it is assumed that large parts of the scene can be considered almost planar, even if exact planarity is unlikely to be found in real scenes. Of course, some parallax errors have to be accepted for only partial scenes. Nevertheless, the results in section 8.2.1 indicate, that the errors are only small in real scenarios. The amount of acceptable parallax errors can be controlled by a planarity constraint applied during the scene decomposition as will be shown in the following.

3.1.3 Realization

The considerations outlined above lead to the proposal of a mosaicing approach appropriate for ego-vision perception:

(i) **Scene Decomposition:** First, a decomposition of the scene into planar sub-scenes has to be computed. In the outlined approach, stereoscopic pictorial information is used to detect these planes. By means of this a scene is decomposed as sketched in Fig. 3.5.

(ii) **Motion Recovery:** Second, the planes have to be tracked individually and online during a sequence. The motion of each plane needs to be recovered in order to calculate warping functions.

(iii) **Pixel Integration:** Finally, for each of the detected planes separate mosaics are created by registering them to their respective reference frame. Existing mosaics are expanded by integrating the current warped planar regions.

Hence, not only one global mosaic is built, but one mosaic for each planar sub-scene. The set of a all plane mosaics is assumed to provide a most complete and compact pictorial representation of the static parts of the scene.

Fig. 3.6 gives an overview of this concept and the computational modules introduced here.

Scene Decomposition

For the scene decomposition a stereoscopic approach is utilized. Therefore, it is assumed that the scene is perceived using two head-mounted cameras as shown in Fig. 4.2. As no assumption regarding the calibration of the camera is made, the approach is applicable to any stereo setup and not restricted to the hardware setup described in Sec. 4.1. For the decomposition a hybrid approach combining feature-based decomposition and image-based propagation is utilized. A feature point generation and tracking approach proposed by Zinßer et al. [175] is applied as a basis for the feature-based decomposition and as well for the motion recovery of the individual planes.

In order to decompose the current view into planar sub-scenes, the following four steps are applied:

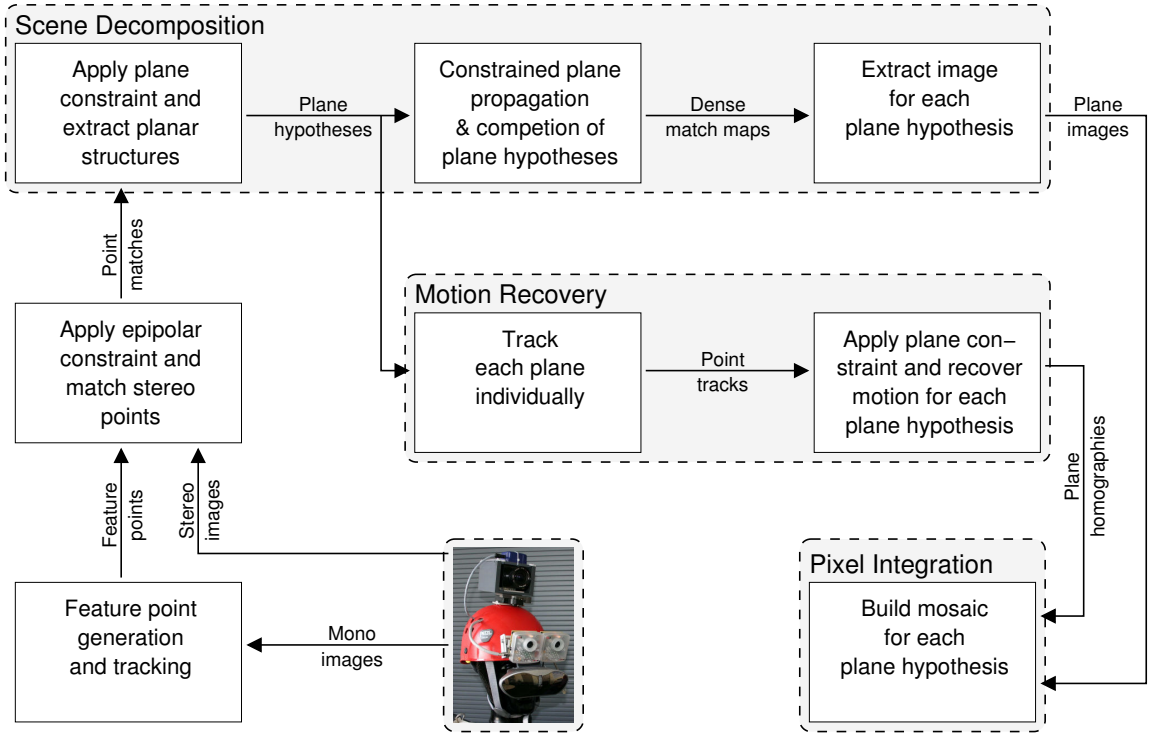


Figure 3.6: Mosaicing system outline: A three-step procedure is implemented for mosaicing in the ego-vision system.

(i) **Local Coplanar Grouping:** Starting with extracted key points from a pair of images provided by the feature point generation and tracking module, their pair-wise correspondences are computed using epipolar geometry and normalized cross correlation [51]. A plane hypothesis is represented by a *local* group of point matches forming a planar patch. The representation of a plane in this state of the processing is purely *feature-based*.

(ii) **Coplanar Grouping - Extension of local patch:** Point matches outside the local patch are added to the plane if they satisfy the plane model.

(iii) **Constrained Plane Propagation:** From a set of point matches, the plane is now extended to pixel regions which satisfy the plane model. The result is a dense match map of a plane which displays textured regions of the plane. By means of this, the representation of a plane is developed in an image-based fashion.

(iv) **Second plane propagation - A map of the plane:** Finally regions with less texture are assigned to the next neighboring textured region. The result is a boolean map which tells whether a pixel is part of the plane or not. Conjuncting this map with the current image of the scene, yields a pixel representation of the plane which is suitable for mosaicing.

Fig. 3.7 illustrates this four steps.

In order to compute plane hypotheses, stereo matching of feature points is applied. Fig 3.8 shows an example of computed stereo matches. It has been shown that stereo images of the same plane $\boldsymbol{\pi}$ are related by a 2D projective transformation (homography) $\boldsymbol{H}^{stereo}$ and

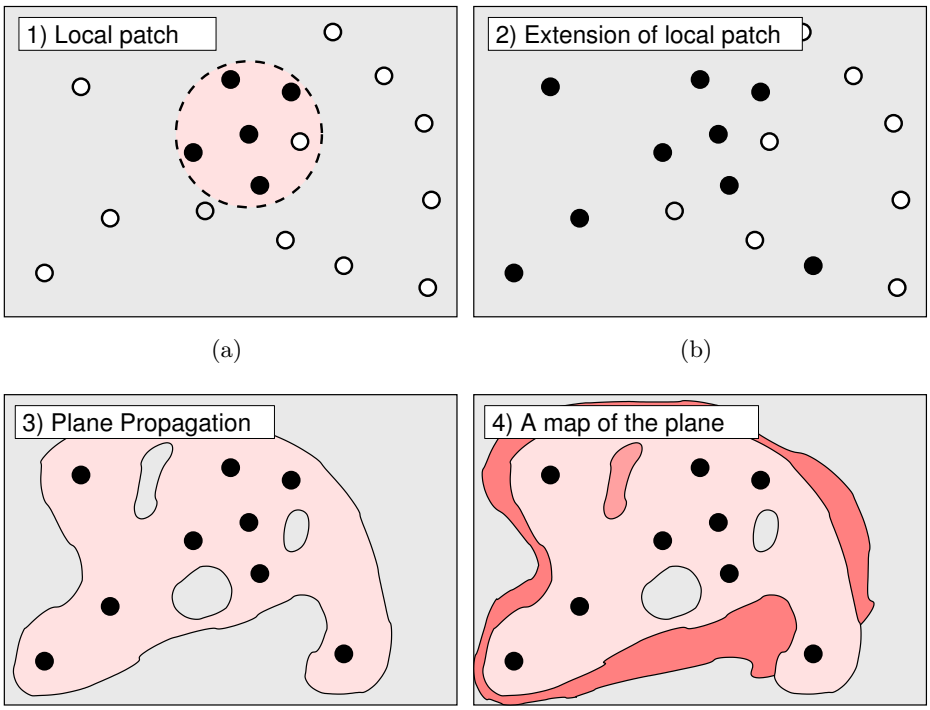


Figure 3.7: Hybrid evolution of a plane: The figure shows the evolution of planes within the scene decomposition procedure illustrated by the sequential expansion of a single plane. In the first and second step, the plane is represented by a set of point matches. A black point indicates an inlier point while a white point represents an outlier point. In the final steps, an image-based representation of the plane is obtained.

Figure 3.8: Stereo matching.

that a homography is uniquely defined by four point matches (cf. [70]). This condition is termed the *plane constraint* as $\boldsymbol{x}, \boldsymbol{x'}$ are pixels that have to satisfy $\boldsymbol{x'} = \boldsymbol{H}^{\boldsymbol{stereo}}\boldsymbol{x}$ and $\boldsymbol{d}(\boldsymbol{H}^{\boldsymbol{stereo}}, \boldsymbol{x}, \boldsymbol{x'}) < \boldsymbol{\delta}$ where $\boldsymbol{d}$ is the distance measure of a point match with respect to a homography.

Feature-based Coplanar Grouping However, after extracting key points from stereo images, any four matched points will define a plane [70] and in turn define a homography which has 8 degrees of freedom. If more points are given, the solution for a homography is over-determined and a model fitting has to be applied. In this approach, RANSAC[3] [53] is applied to fit a homography to matched feature points. RANSAC is an algorithm that is applied to estimate parameters of a statistical model from a set of observed data. It is well-known to be much more robust than, for instance, least square solutions, and especially allows to identify outliers in the observed data.

How can planes be computed from point matches? A *plane hypothesis* is defined as a pair $(\boldsymbol{M_i}, \boldsymbol{H_i^{stereo}})$ where $\boldsymbol{M_i}$ is a set of point matches and $\boldsymbol{H_i^{stereo}}$ a corresponding homography representing the plane model. The set of all point matches is denoted as $\boldsymbol{M}$. The *dominant plane* $\boldsymbol{\pi_{dominant}}$ of a scene is defined as the plane hypothesis which incorporates the largest amount of point correspondences. The basic assumption is that an initial set of point correspondences is given which involves the dominant plane of the scene. But the set contains points which are not compatible to the plane model of the dominant plane. By detecting and removing these outlier points the dominant plane can be recovered. Assuming that there exists a plane model which consists of most of the feature points, these points are termed "inliers" , while points not fitting to that model are "outliers". In order to identify and reject the outlier point, RANSAC is used.

By means of RANSAC, plane candidates $\boldsymbol{\pi_i}$ can be established. By choosing the current dominant plane hypothesis $\boldsymbol{\pi_{dominant}}$ and removing its point matches from $\boldsymbol{M}$, the next dominant plane of the scene is found similarly until no new planes can be found or a maximum number of allowed planes is reached. The result of this procedure is a rough decomposition of the scene into a set of plane hypotheses represented by point matches $\boldsymbol{M_i}$ and respective homographies $\boldsymbol{H_i}$.

Since every four point matches define a plane in general, it is an important issue to distinguish *virtual planes* from physical ones. In order to avoid the extraction of virtual planes, we apply a *local planarity constraint*. By restricting the choice of the four points to random local image areas and neighboring points, and fitting plane hypotheses to these patches, it is probable that extracted planes are at least locally planar.

In order to make the method more robust and to really recognize physical planes only, point matching is extended to patch matching. Given an estimated homography, the neighborhood of feature points in the left image defines a patch that can be transformed to the corresponding right image. By computing the normalized cross correlation between the transformed and the corresponding patch, inappropriate point matches can be identified and discarded from the plane hypothesis. The process of the feature-based coplanar grouping is illustrated in Fig. 3.9.

Image-based Plane Propagation Up to now, an algebraic representation based on feature points has been used for the plane hypotheses. But a plane has a defined extension in the real scene, the plane hypotheses have to be grounded on a pixel level. Since planar surfaces

[3] abbreviation for "RANdom SAmple Consensus"

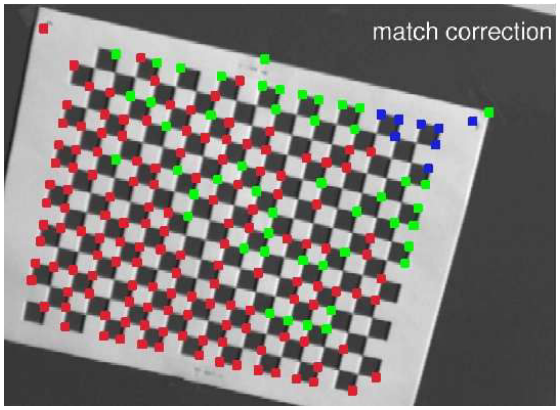
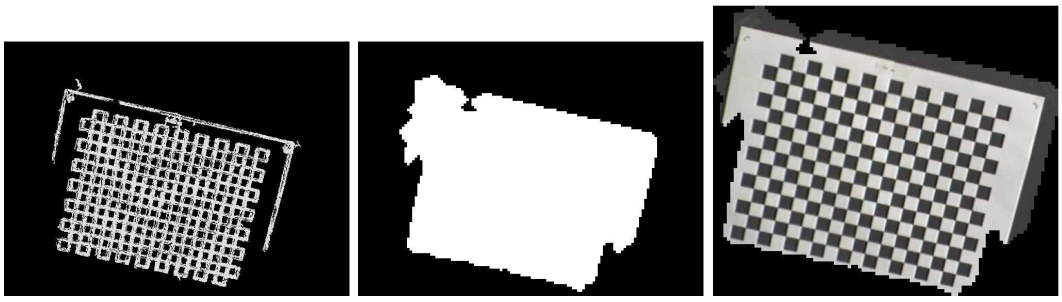


Figure 3.9: Extension of the local patch: Shown is the evolution of a plane hypothesis. Starting from a local patch (blue points), points matches outside the patch are added (green points) if they satisfy the homography of the local patch. The planar homography is updated considering also the new points. This transformation can be used to correct mismatches and add new points (red points) to the plane.

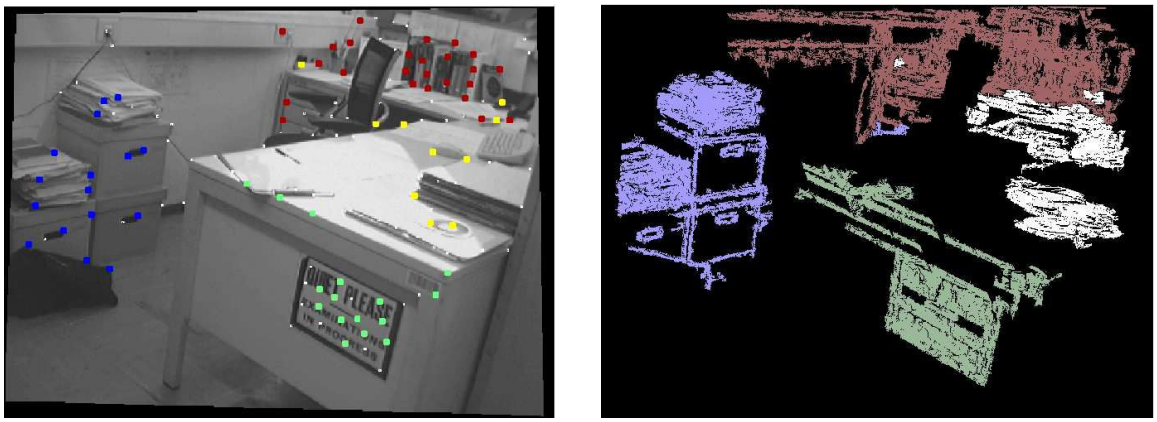
in a scene may contain holes and as there might be regions in the scene for which not enough information is available to assign them to a plane, a pixel-based plane growing method is applied. Based on the algorithm described in [93], an image-based propagation process which densifies the plane hypotheses is utilized. This resembles classical region growing methods for image segmentation. Instead of a homogeneity criterion, normalized cross correlation between point matches is again used for region expansion. Starting from a set of matches with high textureness, the algorithm densifies the matches to regions with less textureness. Expansion stops in regions which diverge from the reference homography or have no texture. This restricts the propagation to regions which can be approximated by the plane hypothesis.



(a) dense map (b) binary map (c) image map

Figure 3.10: Image-based propagation example.

A map for each plane $\boldsymbol{\pi_i}$ from the segmented image is extracted which shows only textured regions of the plane (e.g. in Fig. 3.10(a)). For building a mosaic of the plane we also have to embed the texture-less regions. Thus, another growing procedure is invoked which takes the densed map as seed and extends the map into texture-less regions. The assumption made here is that a discontinuity in the plane also is reflected by a discontinuity in the image. Since this assumption is rather strong, the map growing is limited to a maximal



(a) Detected point matches (b) Plane propagation

Figure 3.11: An example of plane propagation and competition.

distance a texture-less patch is allowed to have from an ensured plane patch. A binary map as shown in Fig. 3.10(b) is computed as the result of this processing step. This map can be applied as a mask on the original image to obtain an image map as shown in Fig. 3.10(c).

Competing Plane Hypotheses So far, only the propagation of a single plane has been considered. Given a set of plane hypotheses, the idea is to start a competition between these planes. Therefore, each plane hypothesis is also associated with the best correlation score among all its point matches. Then, only the plane $\boldsymbol{\pi_i}$ with the best point match $\boldsymbol{m_{best}(a,b)}$ is allowed to start a single propagation step. Thus, the neighborhood $\boldsymbol{N(a,b)}$ of point match $\boldsymbol{m_{best}}$ is densified and new point matches are added to the plane. The chosen plane provides its next best point match and the next iteration begins. The propagation stops if none of the planes has a point match left to be processed. The result of the plane competition and the image-based propagation step is shown in Fig. 3.11.

As a result of the scene decomposition, each plane is represented by its assigned point matches $\boldsymbol{M_i}$, its stereo homography $\boldsymbol{H_i^{stereo}}$ and a pixel-based plane map $\boldsymbol{I_i}$. To create a mosaic, the map of the plane is used to determine the region of the current image that has to be warped and integrating into the resulting mosaic. But how this image has to be warped is not answered yet and will be discussed in the following.

Plane Motion Recovery and Planar Mosaic Construction

We will now turn back to the general approach of mosaicing illustrated in Fig. 3.2. Since the scene is decomposed into planar sub-scenes, mosaicing can be implemented on the basis of the generated planar image maps. But what is needed to integrate new frames into an existing mosaic is the warping transformation, which can be described as a homography $\boldsymbol{H^{motion}}$ between two successive frames of one camera.

Fig. 3.12 illustrates the two different types of homographies applied in the outlined approach. $\boldsymbol{H^{stereo}}$ represents the stereo matching that is computed from the point matches as explained above. We will now take a closer look at the computation of $\boldsymbol{H^{motion}}$. Therefore, we built upon the fact that the motion of a plane is determined by the motion of its points. Fortunately, points on the planes are known by means of the matched feature

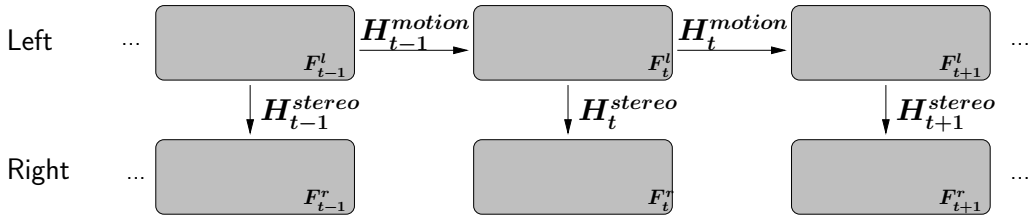


Figure 3.12: Homographies between the stereo frame sequence of a plane.

points, and the applied feature point tracking proposed by Zinßer et al. [175] allows to track these features robustly in real-time of a longer sequence of image frames. Thus, no computationally expensive image matching and registration needs to be applied to estimate the homography $\boldsymbol{H^{motion}}$ between two successive frames. Note, that the computation of $\boldsymbol{H^{motion}}$ is based only on one image of each stereo set. No stereoscopic processing is used to track the motion of the planes. Accordingly, tracking of feature point only has to be performed on mono images.

The motion recovery performed for each plane can be divided into two steps:

(i) **Tracking of plane points:** Given a set of points on a plane, each point is tracked independently. The set of points assigned to a plane is updated by the plane decomposition. Furthermore, the tracked plane has to be updated in terms of integrating new points and removing the ones gone out of sight. Therefore the homography $\boldsymbol{H^{stereo}_{t}}$ is recomputed and new points are added if they fulfill the planarity constraint.

(ii) **Recovering plane motion:** The resulting point tracks $\boldsymbol{T_t = (p^i_{t-1}, p^i_t)}$ are supposed to lie on the same plane. For each two views of a plane, there exists a homography $\boldsymbol{H^{motion}_{t-1}}$ (see Fig. 3.12) which relates $\boldsymbol{p^i_{t-1}}$ to $\boldsymbol{p^i_t}$. Again, RANSAC is used for a robust estimation of this homography.

Based on the inter-frame homographies $\boldsymbol{H^{motion}_{t}}$ all plane images are warped to the reference frame $\boldsymbol{F^l_1}$ of the mosaic. The integration computes the pixel-wise median of the warped frames to determine the value of the resulting mosaic pixel.

Implementation

As ego-vision systems are online systems, some consideration have to be made in terms of computational effort and time consumption. The goal of mosaicing in ego-vision perception is to compensate the limited field of view and serve as a *pictorial memory* (refer to chapter 5) that can be utilized by other components of an EVS. Thus, the goal of the presented mosaicing approach is to compute a *complete* view of planar sub-scenes. Once detected planes should be tracked and extended with new views. The approach needs to keep track of existing planes and relate new views to these existing ones.

This tracking approach requires that processing is done almost in real-time. Unfortunately, computing the scene decomposition is as well expensive as the construction of mosaics in terms of computational costs. Both steps are too resource consuming for recent computing hardware to be applied on every frame of an image sequence, assuming a frame rate of at least 15Hz.

Computing a new scene decomposition every $\boldsymbol{n}$ frames is not sufficient to be able to map the current planes to planes detected in the last processing loop, since the view usually changes too much, due to the arbitrary motion of the camera in ego-vision systems.

Thus, processing is splitted up into three main modules. For the implementation of the outlined approach, ICEWING, a graphical plugin shell that allows vision algorithms to be composed of several plugins [95], is utilized. In conjunction with the communication framework XCF [169], which also serves as a basis for distributed processing in the *visual active memory* approach outlined in chapter 5, ICEWING allows parallel processing. The imple-

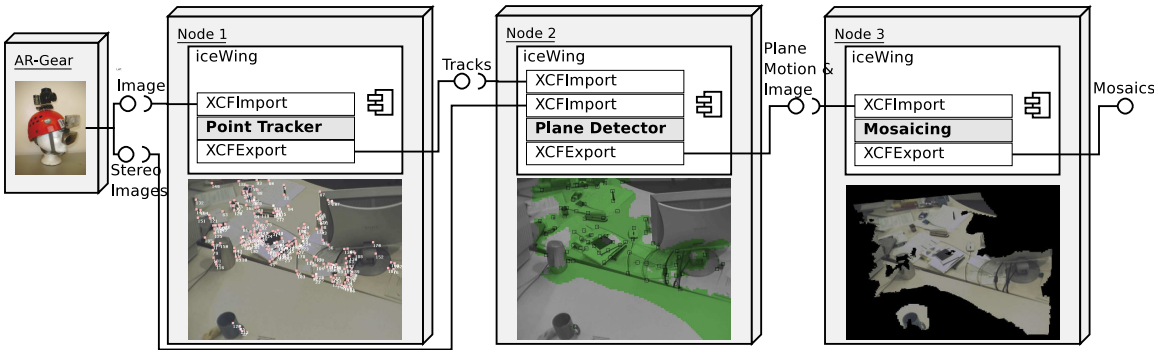


Figure 3.13: Implementation of the Mosaicing using ICEWING

mentation of the distributed processing scheme is illustrated in Fig. 3.13. Tracking and scene decomposition are done in parallel and almost asynchronously.

The two head-mounted cameras (see upper left of figure 3.13) capture stereo images. The ICEWING-Plugin *point tracker* [175] detects feature points in the left image and tracks these at about 15Hz frame rate. The second plugin *plane detector* computes stereo matches from the tracked feature points and the corresponding right images. As this correspondence matching cannot be done at frame rate, parallel processing of tracking and plane detection is essential. In order not to slow down the tracking, the scene decomposition is executed on a different computational node than the feature point tracker. So-called *XCF framework plugins* (denoted `XCFImport` and `XCFExport` in the figure) facilitate a transparent distribution of the processing scheme. Once planes are detected by stereo matching, tracking the feature points indeed also allows to track the individual planes and to compute the respective homography $\boldsymbol{H}^{\boldsymbol{motion}}$ as outlined above. Finally, individual *mosaics* are asynchronously computed whenever new plane information becomes available. By means of the outlined architecture it is on the one hand possible to track planes in real time allowing the approach to be applied in an online setting, and on the other hand provides a reliable decomposition of the scene into planar sub-scenes.

Note, that although the major goal of the approach is to provide a compact pictorial representation of the scene for ego-vision systems, the scene decomposition itself performs a structural analysis of the environment that can be useful for further processing. The knowledge of planes existing in the scene provides contextual information that can be useful, for instance to relate the position of detected objects to another. The role of such spatial contexts is subject to Sec. 3.3.3.

3.2 Object Recognition

With the mosaics approach presented in the last section, the level of a pictorial representation has not been left. The result of the processing is still an image, but to understand and interpret visual perception, we need to go one step further. Symbols describing what can be seen in the scene have to be established. And as a scene is usually comprised of several objects, we will now turn towards object recognition in ego-vision systems. In the notion of this chapter, object recognition, as presented here, is *local* perception, because it is only concerned with the current image that is captured by cameras in an EVS.

Object recognition and categorization is a major issue in computer vision in general and also relevant for assistance systems in particular. As introduced in chapter 1 assistance is most concerned with the handling, memorization, and manipulation of different kinds of objects. Hence, such systems have to be able to detect and classify objects in order to supervise and assist a user.

From the ego-vision perspective, recognizing objects is a special challenge due to the facts outlined in Sec. 2.4.2. Main challenges arise from the variable camera positions, and the requirement of online processing. As a consequence, some requirements have to be considered regarding the selection of an appropriate method to detect and classify objects in EVS.

3.2.1 Object Recognition in Ego-Vision Systems

As already stated, object recognition in ego-vision systems has to be applied online, at least if its result affects the processing loop with the user directly. It must facilitate a reactive behavior of the whole system. Is this sense "online" reflects that, given perception as an image, the object recognition must almost at real-time decide, where an object is in this image and which object it is.

As for all perception functionalities, the object recognition also has to cope with the restrictions of using mobile sensory only. Besides the already mentioned requirements of online applicability, this induces also that only very weak assumption regarding the view point, the illumination, and the position of respective objects can be made by a recognizer in advance.

As a consequence of the claim for adaptability and flexibility of a cognitive system, the number of objects and their appearance is generally not fixed. Thus, object recognition must either be very generic and based on categories, or needs to exhibit learning capabilities to account for specific situations, scenarios, and objects. Accepting that generic object recognition is still very much in its infancies, learning and adaptation become most relevant. Also to account for different illumination and appearances of objects in different scenes, learning seems to provide a loophole out of this dilemma. But in order to be useful in an ego-vision assistance system, object learning must not be an expert task, but something a regular user can accomplish. Whenever a system is applied in a modified scenario, it should neither require a complete rebuild of the object recognition nor of any other part of the assistance system. Accordingly, not only the recognition performance decides about the appropriateness of an approach, but also its (re-)training capabilities.

As object recognition is being studied for decades, many different techniques have been developed. To give a broad survey of the different methods would go beyond the scope of this thesis. But generally two paradigms can be distinguished in object recognition: In the first paradigm, detection and classification of objects are considered as one procedural step in which both problems are solved at once. This solution is also referred to as *object detection* or as segmentation-free recognition. In the following, the most prominent approach in this

category proposed by Viola and Jones [160] will be discussed with respect to its applicability in EVS. An alternative technique using tensor-based discriminant analysis for object detection has recently been published [11]. But also the well-known method of normalized cross-correlation is a representative of this kind of approaches [51].

But object recognition can also be implemented as a two step procedure, comprised of *segmentation* and *classification*. This class of object recognizers has been well studied in research since decades and follows a strict bottom-up processing scheme as already sketched in Fig. 2.2. Common to all approaches in this family is that the first step is applied to determine the boundaries of an object – or parts of it – while successive steps are considered with computation of features and the classification itself. The problem of image segmentation has been studied quite a while. Reviews of different data-driven segmentation techniques are provided in [56, 24]. Most approaches are based on (color) regions, or salient features like edges and are purely data-driven. Successive classification can either be based on the image data itself or on prior computed features.

Related to this class of object recognition methods are so-called *part-based* approaches which do not try to segment the object from the background as a whole, but first try to identify parts of an object independent of each other. The whole object is recognized by analyzing the spatial composition of the local features or patches. Hence, these techniques are also multi-stage bottom-up processes but generally more flexible and less prone too small errors. One of the most famous approaches applying this idea have been proposed by Lowe [97]. He applies scale invariant features and their spatial composition to detect even partly occluded objects in cluttered environments. Ahmadyfard and Kittler [3] combine regions of objects using relaxation algorithms on attributed relational graphs in order to implement the recognition of 3D objects.

In the following, two opposed approaches for object recognition are presented. Each follows a different paradigm in the notion introduced before and both have their advantages and drawbacks with respect to their applicability in ego-vision systems. In this thesis, both approaches are discussed from an application point of view and not on the algorithmic level, since developing new object recognition approaches has not been in the focus. Rather, their respective appropriateness for EVS is investigated and discussed.

3.2.2 A Two Stage Object Recognition Approach: The VPL system

The object recognition method proposed by Bekel et al. [13] is based on the two stage procedure – segmentation and successive classification – as introduced before. In their approach, the segmentation step is based on the integration of different saliency measures such as local entropy, symmetry and Harris' edge-corner-detection [69] in an *attention map*. Based on this map, objects are discerned and segmented. Each segment is normalized in orientation according to its axis of largest extension and then processed by the succeeding classification step.

VPL classification consists of three steps which are outlined in Fig. 3.14. First, vector quantization is applied (V-step) to obtain a raw partitioning of the input data. After assigning an incoming image patch to one of the clusters, PCA is locally applied to extract suitable features for classification (P-step). This simple realization of local PCA enables fast training and avoids manual tuning of training parameters. Finally, several local linear maps (L-step) project the extracted features onto a probability vector. Each component of this vector corresponds to an object class known to the system. The final classification result is the object with maximum probability.

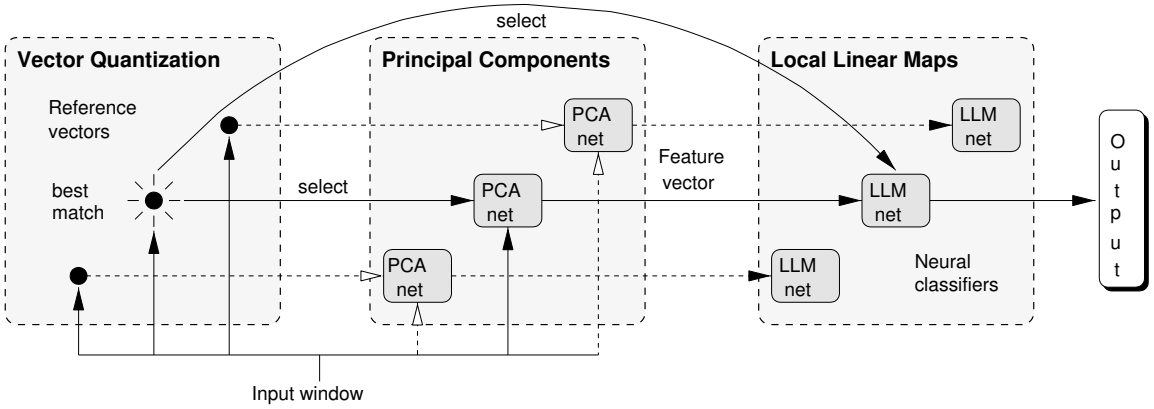


Figure 3.14: Schema of the VPL classifier as proposed by Heidemann et al. [71].

The approach has some benefits for ego-vision systems that shall be outlined in the following:

- ▷ Object segmentation and classification work in (soft) real-time at frame rate on recent computing hardware.
- ▷ The system is trainable with only few (about five) views of an object from different directions to provide reasonable results. The respective number of required views in the training set is dependent on the level of generality that needs to be achieved in recognition and the given task.
- ▷ The separation between segmentation and classification allows to apply the segmentation step alone. A cognitive vision system can use the information that there is an object, defining a region of interest, at a given place, although the class is not (yet) known. This facilitates interactive learning in a mixed initiative style, since the system is able to indicate or reference an object and ask the user about its name.

But the outlined method also has a major drawback that should not be left unconsidered here. The segmentation step of the VPL approach is purely data-driven in a bottom-up manner, and not applying any knowledge about objects and the background. It is therefore rather prone to errors and mis-segmentations. Especially cluttered backgrounds make the segmentation inapplicable, since object and background are not distinguishable in that case. The scenarios, in which this approach can be applied are therefore restricted, especially because no rejection class is implemented.

3.2.3 A Boosted Cascade of Classifiers for Object Detection

In order to encounter the problems arising from purely data-driven image segmentation inducing unrecoverable errors in object recognition, object detection approaches, that evade explicit segmentation or perform segmentation and classification at once, gained focus. The method proposed by Viola and Jones [160] belongs to the class of object detection algorithms. Their framework has originally been used to detect faces in images very rapidly and very reliably. We adopted and slightly modified the original approach and applied it for object detection in ego-vision systems.

Viola and Jones [160] proposed to use rather simple block features, that can rapidly be computed on grey-scale images as the basis to detect textured structures in images. Such

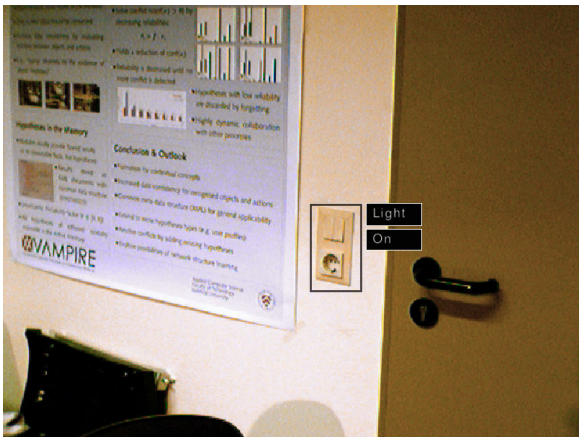


Figure 3.15: Object detection result: A light switch has been detected.

features are computed as the differences of summed up pixel values between two distinct regions in a defined search area in the image. The classification is a thresholding operation regarding this difference. The power of the approach lies in the very fast evaluation of these features for a very high number of candidate regions in the image.

In the training phase, boosting [141] is applied in order to establish the optimal combination of features and parameters to solve a two class problem which decides whether a respective region contains the object in question or not. An example of a detection result is presented in Fig. 3.15 for an object detector that has been trained on light switches.

In order to implement a multiple object recognition system using this approach, multiple detectors have to be instantiated, one for each object. The approach has some benefits regarding the requirements outlined at the beginning of this section:

- The object detection is very fast and performs in real-time.
- In contrast to the previous approach, object detection works very reliably in cluttered environments. It avoids the separate segmentation step in preference for a direct search for respective objects.
- The recognition is based on grey-scale images only, which makes it more robust against illumination changes.

But again, some major drawbacks have to be considered, too:

- The outlined approach requires a huge amount of training data. The number varies depending on the generality that needs to be achieved, but can easily reach a magnitude of 1000 samples per object. To make the approach nevertheless applicable for interactive online training in ego-vision, we make profit of the human in the loop paradigm of ego-vision systems. Visual object tracking [63] is applied to crop views of the object that is to be learned from the continuous image stream. By means of this, the user only has to move the head while keeping the object of interest in focus to acquire the necessary number of training images in about a minute, assuming an approx. frame rate of 15Hz.
- In addition to the high number of training samples required by the outlined approach, the training itself is very time consuming. Boosting in general is a rather “brute-force”

approach systematically appraising many different features causing the long training times. We developed a modification of the original approach of Viola and Jones [160] to speed-up this process by selecting only a smaller set of features randomly in each boosting cycle. In order to compensate the reduced recognition performance caused by this random selection *gentle AdaBoost* is applied instead of *discrete AdaBoost* as in the original approach. Details regarding this enhancement can be found in the work of Peters [123]. With several other optimizations of the approach in terms of extension of the feature set and an additional optimization of the weak classifiers, the training time could be reduced from several days to a few hours in magnitude. Nevertheless, training still takes too long to train an object detector in an online fashion.

- ▷ The object detector scales linearly with the number of objects to be detected, since an individual detector needs to be applied for each object class. This property reduces the number of distinct objects that can be recognized in real-time.

Conclusion Two rather different approaches for object recognition have been outlined above, and a clear decision regarding their appropriateness for ego-vision assistance systems is hard to be drawn. It is desired to have an object recognition that is able to work reliably even in cluttered and complex environment. But the price paid by the boosted cascade approach in terms of training time and amount of training samples is quite high. For the tasks of ego-vision assistance systems, the VPL approach presented first is more convenient, as it allows rapid training of new objects in an online matter, although other approaches in recent research report better discrimination capabilities. But interactive learning is one of the goals in assistance systems in particular and also in cognitive vision systems in general. Therefore, the VPL approach has been developed by a project partner with the emphasis on the kind of envisioned interactive ego-vision systems. As a consequence, this object segmentation and recognition technique is chosen for the integrated system that undergoes user studies presented in chapter 7.

3.3 Spatiality in Ego-Vision: From 2D to 3D

In the last section object recognition has been presented that takes the current perceived image and either detects or classifies objects in this image. As a consequence, the position of an object is given with respect to the coordinate system of the current image, which has been discussed as *local* perception in the introduction of this chapter. But a scene is basically three-dimensional. The goal is to relate the two-dimensional local perception to the three-dimensional spatial context to provide a most complete *scene model*.

Such a scene model is especially important to facilitate assistance in terms of a spatial memory.

Example 3.3: *Reconsider the famous example of the lost keys introduced at the very beginning (Ex. 1.2). In order to tell Zoe where she left her keys, the system must have a scene model in which the position of these keys are stored.*

The human in the loop paradigm of ego-vision systems facilitates joint attention between the user and the system in a very direct way as outlined already in the definition of EVS in chapter 2. Thus, in ego-vision the spatial context provides a very rich cue to determine the user's attention by means of the currently focused part of a scene. In order to use this model of visual attention to realize situation-aware assistance, the viewing direction and position of the user, and correspondingly the system, needs to be determined.

In this section a closer look is taken on how the spatial viewing context can be determined and how it can be used to implement a three-dimensional scene model. Again, specific properties of EVS make these tasks challenging. The sensory for perception should be "on-board" and the position and orientation of the camera is neither restricted nor controlled by the system. Hence, the algorithms have to be carefully chosen to account for these demands.

3.3.1 Visual Self-Pose Recognition and Tracking

The perception of the system is directly affected by the viewing direction and position of the user and, in consequence, the camera. In order to provide a global spatial context, this parameters need to be recognized from a global point of view. This means, that the pose $\boldsymbol{T_{pose}}$, which consists of three parameters $\vec{\boldsymbol{r}}$ for the rotation, and of three parameters $\vec{\boldsymbol{t}}$ for the translation, of the camera should be given in a global coordinate system.

There are several different solutions known to the problem of pose tracking. A good survey is provided by Rolland et al. [136], who discuss as well so-called inside-out and outside-in tracking approaches. The difference between both classes of tracking is the positioning of the tracking sensors. In outside-in tracking approaches the sensor is fixed with the reference coordinate system in the environment and tracks features on the moving target. Hence, they are not applicable in ego-vision system, since no external sensory should be used here. The alternative is inside-out tracking in which the sensory is moving together with the object to track. The features that are tracked to compute the ego-motion are fixed in the environment and define the global coordinate system.

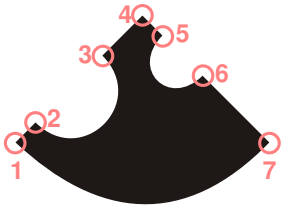


Figure 3.16: Cheese target.

Chandraker et al. [22] propose a vision-based inside-out tracking solution that determines the pose of the camera with respect to artificial landmarks, which is utilized here. By means of a fully calibrated, head-mounted camera, landmarks can robustly be detected, and are tracked to provide a continuously updated estimation of the pose according to the global coordinate system defined by the targets. To pose tracking at high frame rates and more robustly, a highspeed CMOS camera [109] is applied here, although the approaches basically works also with standard head-mounted cameras. To utilize their approach in indoor environment, so-called "cheese target"[4] landmarks shown in Fig. 3.16 are attached to the walls as illustrated in Fig. 3.17. Several of these targets can be used in order to cover a larger area of the environment or to be more robust, but one target is already sufficient to determine the six parameters of the camera pose using a geometric analysis of the seven corners as shown in Fig. 3.16. The relative pose $\boldsymbol{T_{pose}}$ of the camera with respect to this target is determined by four points already, given that the three-dimensional world coordinates of these four points are known [22]. Additionally detected points improve the accuracy and robustness.

By means of this approach, an ego-vision system can perceive highly accurate pose parameters for all six degrees of freedom at a high rate of more than $\mathbf{15}\boldsymbol{Hz}$. The approach requires some explicit calibration steps, which is at least to some extend contrary to the idea of cognitive systems of being flexible regarding the scenarios. However, inside-out tracking approaches require much less effort compared to outside-in approaches. They still provide the most flexible means to implement a global reference coordinate system. An extension of the approach proposed by Stock and Pinz [151] also incorporates natural landmarks to provide an avenue to an even more flexible pose estimation in future systems.

[4]The term "cheese target" is devoted to the shape of the artificial landmarks shown in Fig. 3.16.

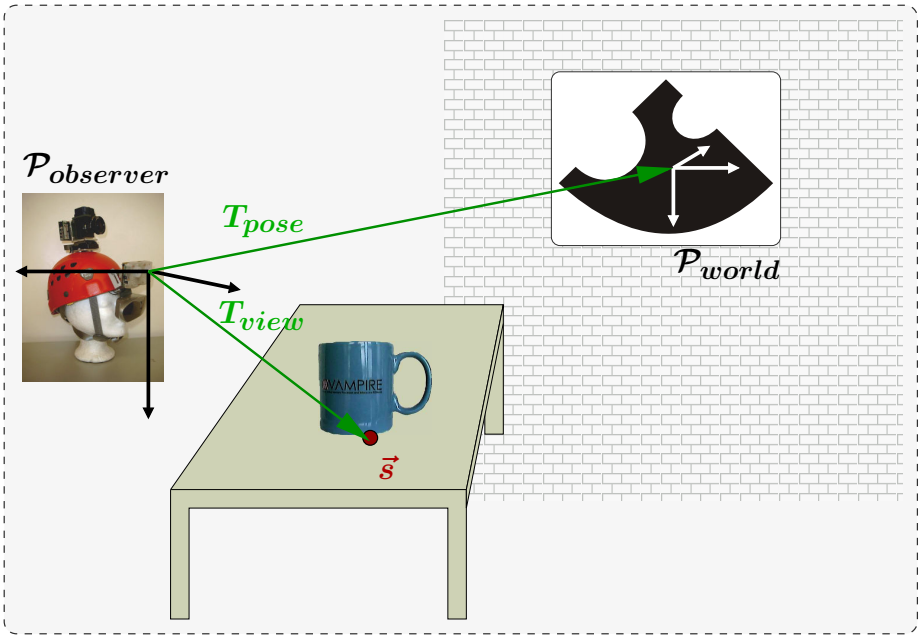


Figure 3.17: Self-pose tracking and object localization: The global position of an object is determined by the pose of the observing camera.

3.3.2 3D Object Positions

In section 3.2 approaches for the local perception of objects resulting in two-dimensional object position in the current image have been presented. With the self pose computation at hand, it is now possible to put these local perceptions in the global spatial context. To be precise, the goal is to compute three-dimensional object position with respect to a global coordinate system.

The required transformations are already sketched in Fig. 3.17. Assuming a world coordinate system $\mathcal{P}_{world}$ a transformation $T_{pose} = (R, \vec{t})$ to the coordinate system of the observer $\mathcal{P}_{observer}$, which equals the perspective of the system in our case, is given as the result of the self pose computation outlined before. Now, in order to compute the 3D position $\vec{s}$ of the object with respect to the world coordinate system, the transformation T_{view} needs to be estimated. This problem here is, that images are only two-dimensional and do directly allow to infer the required three-dimensional transformation. Information about the distance between the camera and the object is not directly available.

But several solutions are known to solve this problem. Besides using special sensors that directly measure the distance to the object, e.g., with laser range sensor, stereoscopic image processing techniques are often applied. [70] gives an overview of the different approaches and also background information on how to utilize stereo information to compute the depth of a scene. Generally, stereoscopic approaches are usually motivated by human vision capabilities and make use of the *disparity* between two images of the same scene if viewed at from different positions. In this work, stereoscopic image processing has been applied to detect planes in the scene as outlined in Sec. 3.1. To compute the distance of an object to the camera, one can either compute a depth map on a pixel level from local stereo disparities, or the disparity of object positions from two different view points, as for instance proposed by [146].

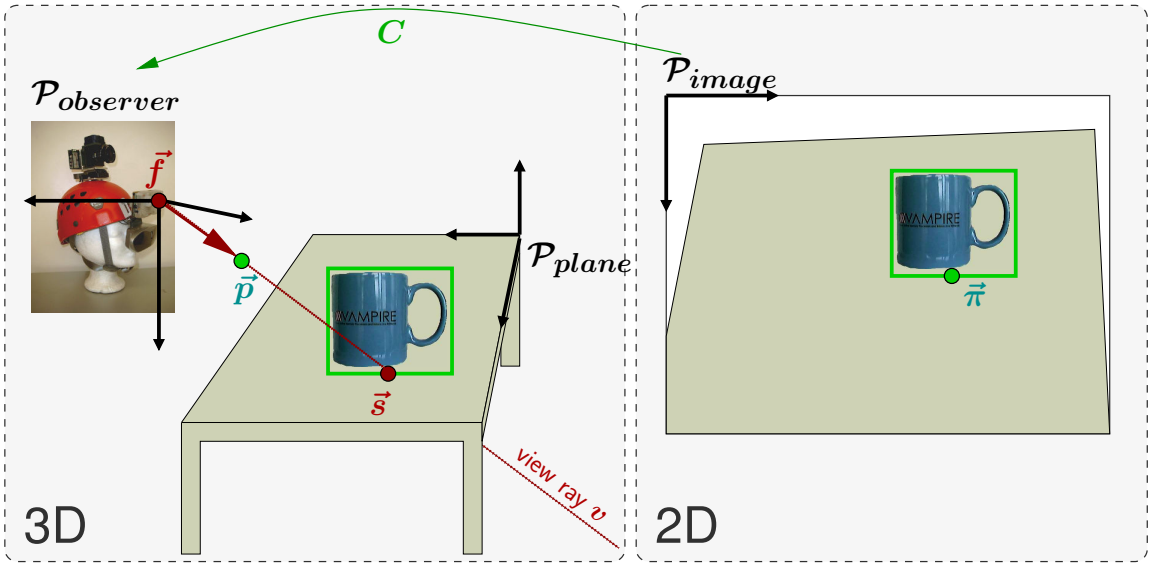


Figure 3.18: 3D position of an object.

But alternatively, contextual knowledge about the scene can be applied in order to compute the 3D positions of objects. Assuming that an object is always positioned on-top of some other object and not floating in "mid-air", its position can be computed if the correct position of the supporting object is know. In Fig. 3.17 this idea is sketched in terms of a cup, that is placed on a table surface. If the three-dimensional position of this table is known, the position of the object can be easily computed from a monocular image, as will be shown in the following.

In order to simplify the problem, it is assumed that the image of objects is captured up-right, so that the lowest pixel in the object bounding box in the image approximately corresponds to the real world lower boundary of the object. In consequence, not a complete three dimensional description of the object's boundaries is computed, but only its lowest center point, termed $\vec{s}$ in Fig. 3.18, is identified. But for most applications in assistance scenarios this coarse description is sufficient, since it allows to re-localize the object accurately enough. Accordingly, it will now be outlined how the three-dimensional position of this support point $\vec{s}$ can be computed from the central pixel $\vec{\pi}$ of the lower bounding box of the particular object as shown on the right of Fig. 3.18. As explained by equation 3.1 on page 31 already, a projection is defined by

$$\vec{x} = \begin{pmatrix} x \\ y \\ \lambda \end{pmatrix} = P \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \quad P = C[R|\vec{t}] \; . \tag{3.6}$$

C is computed using camera calibration and represents the transformation from the image $\mathcal{P}_{image}$ to the camera coordinate system $\mathcal{P}_{observer}$ as illustrated in Fig. 3.18. R and $\vec{t}$ denote the rotation and transformation of the camera coordinate system to the world coordinate system $\mathcal{P}_{world}$, respectively. They are obtained using self position recognition as outlined before. Note, that a transformation between the head-mounted CMOS camera on top of the helmet in the left picture of Fig. 3.18, that actually is used for the pose recognition, and the front-mounted cameras is neglected in the following discussion. This transformation is fixed in the setup and needs only to be calibrated once. It can be seen

as part of the pose recognition and tracking algorithm, and is therefore incorporated in $\boldsymbol{R}$ and $\vec{\boldsymbol{t}}$.

Without loss of generality, it can be assumed that the optical axis of the camera is pointing out of the $\boldsymbol{X}$-$\boldsymbol{Y}$-plane with respect to the camera coordinate system intersecting the image plane at $\boldsymbol{(0,0,1)^T}$. The focal point is thus defined as $\vec{\boldsymbol{f}}_{\boldsymbol{cam}} = \boldsymbol{(0,0,0)^T}$ in the $\boldsymbol{\mathcal{P}_{observer}}$ coordinate system.

A pixel $\vec{\boldsymbol{\pi}}$ with respect to $\boldsymbol{\mathcal{P}_{image}}$ can now be transformed into the camera coordinate system $\boldsymbol{\mathcal{P}_{observer}}$ by

$$\vec{\boldsymbol{p}}_{\boldsymbol{cam}} \;=\; \boldsymbol{C^{-1}} \begin{pmatrix} \vec{\boldsymbol{\pi}} \\ \boldsymbol{1} \end{pmatrix} . \tag{3.7}$$

By means of this object point $\vec{\boldsymbol{p}}_{\boldsymbol{cam}}$ a *view ray* $\boldsymbol{v_{cam}}$ passing the focal point $\vec{\boldsymbol{f}}_{\boldsymbol{cam}}$ is defined. Although the distance to the object is still not known, its position is on the straight line defined by the view ray. Afterwards, this view ray has to be transformed to the world coordinate system. Therefore, $\vec{\boldsymbol{p}}_{\boldsymbol{cam}}$ is transformed to $\boldsymbol{\mathcal{P}_{world}}$ resulting in

$$\vec{\boldsymbol{p}} \;=\; \boldsymbol{R^{-1}} \left(\vec{\boldsymbol{p}}_{\boldsymbol{cam}} - \vec{\boldsymbol{t}}\right) \tag{3.8}$$

as illustrated in Fig. 3.18. The second point that defines the view ray is the focal point $\vec{\boldsymbol{f}}_{\boldsymbol{cam}}$ which can also be transformed quite similarly as

$$\begin{aligned} \vec{\boldsymbol{f}} \;&=\; \boldsymbol{R^{-1}} \left(\vec{\boldsymbol{f}}_{\boldsymbol{cam}} - \vec{\boldsymbol{t}}\right) \\ &=\; \boldsymbol{R^{-1}} \left(-\vec{\boldsymbol{t}}\right) \end{aligned} \tag{3.9}$$

since $\vec{\boldsymbol{f}}_{\boldsymbol{cam}} = \boldsymbol{(0,0,0)^T}$. Given $\vec{\boldsymbol{p}}$ and $\vec{\boldsymbol{f}}$, the view ray on which the object point is located in the world coordinate system, is simply defined as

$$\boldsymbol{v}: \;\; \vec{\boldsymbol{s}} = \;\; \vec{\boldsymbol{f}} + \boldsymbol{a} \left(\vec{\boldsymbol{p}} - \vec{\boldsymbol{f}}\right) . \tag{3.10}$$

Now everything is at hand to compute the three-dimensional position of the object in the image. Therefore, the view ray $\boldsymbol{v}$ is intersected with the plane $\boldsymbol{p}$, which pose is assumed to be known. A plane is defined by its normal vector $\vec{\boldsymbol{n}}$ and a point $\vec{\boldsymbol{b}}$ that lies on the plane [5] by the following condition

$$\boldsymbol{p}: \;\; \boldsymbol{0} = \;\; \vec{\boldsymbol{n}} \left(\vec{\boldsymbol{s}} - \vec{\boldsymbol{b}}\right) \tag{3.11}$$

for any point $\vec{\boldsymbol{s}}$ on that plane. The intersection between a given line $\boldsymbol{v}$ and a plane can be computed as

$$\boldsymbol{a} \;=\; \frac{\vec{\boldsymbol{n}}(\vec{\boldsymbol{b}} - \vec{\boldsymbol{f}})}{\vec{\boldsymbol{n}}(\vec{\boldsymbol{p}} - \vec{\boldsymbol{f}})} . \tag{3.12}$$

Substitution of $\boldsymbol{a}$ in Eq. 3.10 defines the position $\vec{\boldsymbol{s}}$ of the object in the world coordinate system as illustrated in Fig. 3.18 by

$$\vec{\boldsymbol{s}} \;=\; \vec{\boldsymbol{f}} + \frac{\vec{\boldsymbol{n}}(\vec{\boldsymbol{b}} - \vec{\boldsymbol{f}})}{\vec{\boldsymbol{n}}(\vec{\boldsymbol{p}} - \vec{\boldsymbol{f}})} \left(\vec{\boldsymbol{p}} - \vec{\boldsymbol{f}}\right) . \tag{3.13}$$

In order to compute the three-dimensional position of objects using this approach, planes in the scene need to be identified and parameterized. There are two possibilities, how this

can be achieved. First, the planes identified by the scene decomposition, that has been part of the creation of mosaics of planar sub-scenes as presented in Sec. 3.1, can generally also be applied to localize objects in 3D. A practical problem is that each plane in these scene decomposition is only defined by a homography, and therefore does not contain complete three-dimensional information as required here. A solution to this problem is simple: One can attach cheese targets to some relevant plane in order to obtain valid 3D information by applying the outlined pose recognition. The extend of such a plane can be determined by the scene composition approach in the regular way. Furthermore, the information about the correct 3D pose of a given plane that contains a cheese targets and has also been detected by the scene decomposition, basically also allows to calibrate the stereo setup presented in Sec. 3.1. In consequence, the three-dimensional pose of other planes can be determined and object positions on that planes can as well be computed.

Alternatively, the relevant plane can be pre-defined together with the setup of cheese targets in the room. Given a defined indoor setup as sketched in Fig. 3.17 with objects placed on a table, one can manually define the transformation to this table plane $\boldsymbol{\mathcal{P}_{plane}}$ with respect to the cheese targets defining the world coordinate system attached to the wall. Thereby, an efficient and very robust solution is provided to compute the three-dimensional position of objects from monocular two-dimensional recognition results.

3.3.3 Planes Defining View Contexts

The idea of a decomposition of the environment into different planes provides an avenue to define specific *view contexts*. Such a view context can be defined as a limited spatial region which is looked at by the user. A table is an example of such a view context and it can be useful to associate specific properties, expectations or functions with it.

Example 3.4: *Assume a construction scenario with a table, on which the assembly should be constructed, and a box containing assorted components. Whenever the user looks into the component box, the assistance system tells the name of the respective component looked at. When the user looks at the construction site, the next step of the construction is communicated and supervised.*

In this example, the functionality of the system is directly controlled by the user, simply by the view direction. View contexts can generally be pre-defined, or result from an automatic scene decomposition as outlined in Sec. 3.1. They can be determined from the user's pose in a rather similar way as object positions are computed.

A view context $\boldsymbol{V_i = (L_i, p_i)}$ is defined in space by means of a plane constraint

$$\boldsymbol{p_i}: \quad \mathbf{0} = \quad \vec{\boldsymbol{n}}_{\boldsymbol{i}} \left(\vec{\boldsymbol{x}} - \vec{\boldsymbol{b}}_{\boldsymbol{i}} \right) \tag{3.14}$$

and a boundary $\boldsymbol{L_i}$, given as a closed, regular, and convex polygon on $\boldsymbol{p_i}$. A binary function $\boldsymbol{lookAt(\vec{a}, V_i)}$, that is evaluated as either "true" or "false", can be defined to determine whether a view context is active or not. In order to determine its value, the optical axis $\vec{\boldsymbol{a}}$ of the head-mounted camera, which pose is known by the methods introduced in the previous section, is intersected with the plane $\boldsymbol{p}$ in the similar way as described by Eq. 3.12. Afterwards, the resulting intersection point is checked whether it lies inside the polygon $\boldsymbol{L_i}$.

The definition and utilization of view contexts allows to control system behaviors by simply looking at specific parts of the scene. The view context of a table will play a major role, when the integrated assistance system is presented in chapter 7. By defining view contexts as bounded, planar areas, a flexible means is provided to implement such spatial contexts.

3.4 Action Recognition

In the introduction to this chapter it has been discussed, that ego-vision perception is not only considered with the static composition of a scene but also with its modification and events occurring in it. In the assistance system focused on here, the user has to deal with objects and their manipulation. Therefore, in the notion applied in this section, *actions* carried out in the environment deal with the manipulation of objects performed by the user.

Allowing the ego-vision system to perceive and recognize human actions paves the way to a more context-sensitive assistance and is a foundation in order to supervise the accomplishment of instructions.
Example 3.5: *When John assembles his wardrobe, he requires step-wise assistance (see Ex. 1.1). Although he could explicitly ask for the next instruction or look it up in the manuals, a human assistant, Zoe in that given example, would be more cooperative as she would engage in case of errors and would automatically provide the next instruction if one step has been accomplished.*
In order to realize a like-wise cooperative functionality in artificial assistance systems, actions carried out in the environment need to be correctly recognized.

Although recognition of human actions is a challenging task in principle and a lot of research is conducted on this subject, the ego-vision perspective makes the problem even more complicated. In contrast to the approaches for ego-vision perception presented before, the *temporal* context plays a crucial role in action recognition, because actions inherently have an extension in time. Therefore, the dynamics in the scene need to be taken into account.

Bobick [16] proposed a quite famous taxonomy to distinguish between *movement*, *activity*, and *action*. Movements are atomic primitives that can be segmented and classified individually. They are local and usually view dependent. Bobick [16] introduces activities as sequences of different movements, that are no longer view dependent. Recognizing activities mainly considers the statistics of the composition of this sequence of movements. I will not follow this distinction very strictly, as I will focus on activities that are mainly composed of only one movement. In consequence, activity recognition to some extend equals movement classification in our system.

On top of movements and activities, Bobick proposes *actions* as larger scale entities that require interpretative context. Action recognition accordingly considers not only the mentioned motion patterns in space and time, but also contextual knowledge, causal relationship, and interaction. The context of the activity and its effect play a major role, too. Action recognition can therefore not be seen as an enclosed problem, but is affected by many different functionalities and components of a vision system. The necessary context, that allows to recognize actions as proposed in this thesis, is given in terms of object recognition results and view contexts as outlined before. The focus in this section is to explain, how *activities* can be recognized from the ego-vision perspective. As stated before, the recognition of activities directly correspond to the classification of the objects' movements. Of course, in order to recognize action from these activities successively, the contextual information is incorporated.

The recognition of object manipulations or other human activities from visual cues is a well studied subject in recent research. Generally, holistic and trajectory-based approaches can be distinguished. The first analyze the image sequence as a whole in order to extract motion patterns that are specific for given activities. Techniques to recognize motion between successive images like optical flow as proposed by Cutler and Turk [39] or Nagai [112], for instance, or temporal templates [40], are often applied in order to recognize activities.

Usually such approaches are only applicable for static or controlled cameras and are rather sensitive to environment changes.

Alternatively, trajectory-based approaches do not analyze the complete image, but capture *local* movements, usually of either parts of the body or the manipulated objects itself. Most approaches found in literature analyze the trajectory of the manipulating hand to classify activities. The trajectory is acquired by segmenting and tracking skin colored regions or special markers attached to the hand. These approaches for visual trajectory-based classification either use a static camera for, e.g., surveillance [131] or interaction [119]. Acquiring a trajectory always requires approaches to track an object or body part from one frame to the other in an image sequence. Therefore, tracking is generally closely related to action recognition using trajectory-based approaches.

The work of Yu and Ballard [173] is based on an assumption similar to the one taken in our approach. They presume, that users track their own hand when they perform an action. Yu and Ballard [173] take this assumption to recognize attention changes and segment longer term actions.

An approach that provides a basis for the work presented in the following is proposed by Fritsch et al. [55]. They capture hand trajectories using a static camera and combine these with contextual information about the manipulated objects to allow a robust classification of human actions with a probabilistic framework.

3.4.1 Action Recognition in an Ego-Vision System

In order to recognize manipulation actions in an ego-vision system, an object-centered approach is preferred to an analysis of the hand trajectory as it is originally proposed by Fritsch et al. [55], since the hand is more unlikely to be present in the field of view than the object itself. The proposed approach starts with initially detected and classified objects and analyzes their movement during manipulation. This movement is classified into different activities and used together with the class of the object to recognize the action that is carried out. The fundamental assumption taken there is that the recognizable actions are distinguishable on the basis of two properties: First, the *class* of the involved object, e.g., bottle, cup, screwdriver. And second, the *movement* of the object while it is used or manipulated, respectively. In the notion of Bobick [16], the object's class is part of the context that allows to classify actions. This problem can be considered as being solved by the object recognition approaches outlined before in Sec. 3.2. The coupling between object recognition and action recognition is subject to Sec. 7.3.2 and will be discussed in detail there.

Reconsidering the challenges of ego-vision, again the fact that no external sensors should be used and that the motion of the camera is therefore unrestricted, are determining the choice of appropriate algorithms. Furthermore, the limited field of view has to be considered. The above arguments raise two major questions:

1. How to capture the movement of manipulated objects?
2. How to compute the absolute movement from the ego-vision perspective?

Answers to these question are given in terms of an appropriate combination of different visual tracking approaches in order to capture the object's motion, to compensate the camera's motion, and finally to compute the *absolute* trajectory of manipulated objects. In

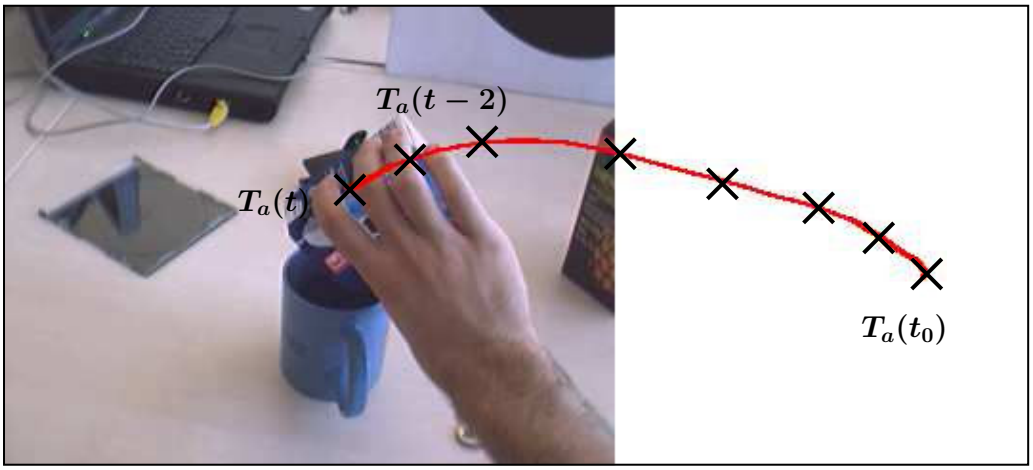


Figure 3.19: Frame $\mathcal{I}_t$ with absolute trajectory.

Fig. 3.19 the absolute trajectory of a manipulated object is shown. Note that the complete trajectory is not visible in the current frame $\mathcal{I}_t$. The movement of the object has started earlier and the user followed the object's movement with the eyes. The start point is therefore out of sight but is still relevant in order to recognize the activity correctly, which is exclusively determined by this movement.

The complete architecture of the action recognition sub-system is sketched in Fig. 3.20. The highlighted modules involved in "Activity Recognition" are subject to a more detailed explanation in the remainder of this section. Activity recognition is split up into four different modules. At first, the movement of the object has to captured, which is implemented by the *region tracking* module. Object recognition is very unlikely to be applicable to capture the movement of the manipulated object as it either does not perform fast enough to be applied on every frame of the image sequence, or is very sensitive to occlusion or cluttered background as discussed in Sec. 3.2. But it can be assumed that the object is recognized and detected until the manipulation action actually begins. Accordingly, it is proposed to initiate real-time region tracking of objects that are likely to be manipulated. Instead of tracking every object that is found in the field of view, a model of *shared attention* is utilized as a consequence of the ego-vision concept of the system. Due to the shared perception it

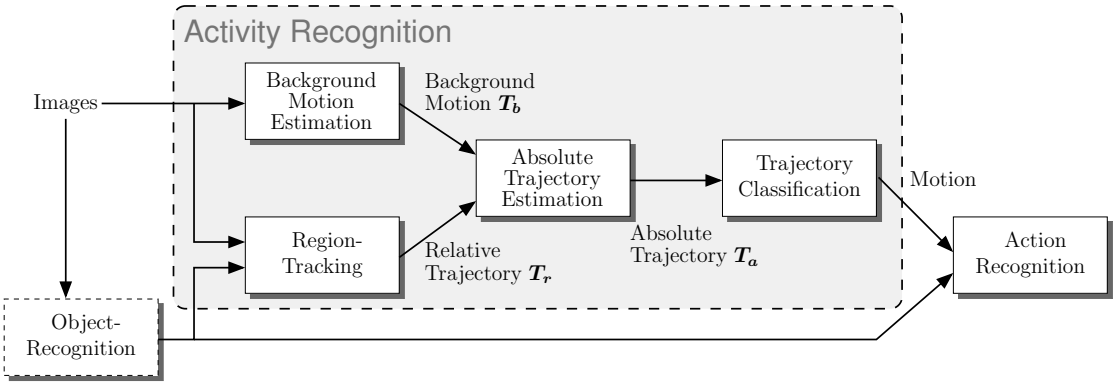
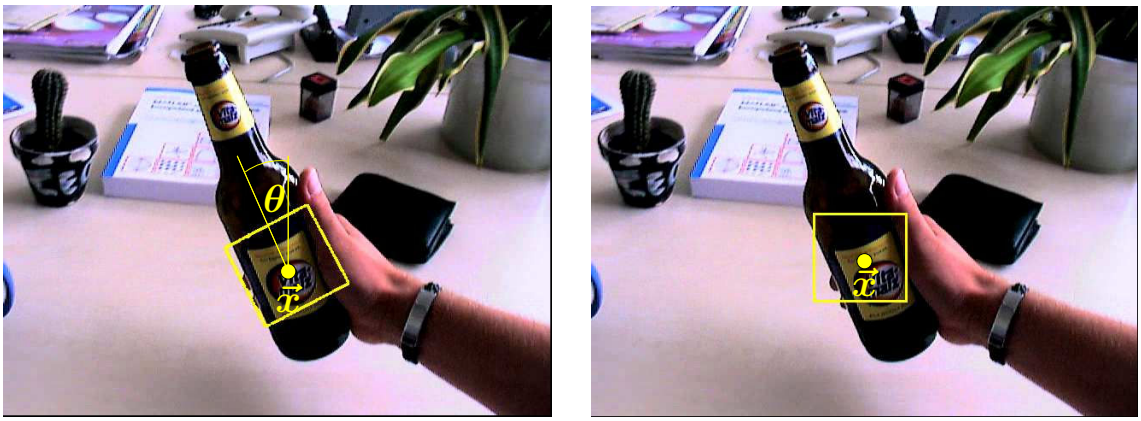


Figure 3.20: Architecture combining visual tracking and condensation-based trajectory classification for activity recognition.



(a) Hyperplane tracker (b) Kernel-Based tracker

Figure 3.21: Object tracking.

is possible to recognize at which object the user looks at a given time. Assuming that users generally look at an object before they start to manipulate it allows to track these objects of interest. Therefore, tracking is initiated on objects that have been reliably detected and are in the center of the user's view. It will be detailed in Sec. 7.3, how the interplay between the different components is implemented in the integrated system.

Any object of interest is tracked, no matter of it is manipulated or not. Consequently, not every movement corresponds to an activity that should be recognized by the system. Due to that reason, it is important that the successive classification method is able to reject unknown movements as non-relevant activities.

Since the scene is perceived from head-mounted cameras exclusively, the *region tracking* module only provides the *relative* motion $\boldsymbol{T_r}$ of the object with respect to the user's view. To obtain the *absolute* motion $\boldsymbol{T_a}$ it is furthermore necessary to estimate the user's or camera's motion $\boldsymbol{T_b}$, respectively. The corresponding module to compute $\boldsymbol{T_b}$ is termed *background motion estimation*. The module *absolute trajectory estimation* computes the absolute motion $\boldsymbol{T_a}$ from $\boldsymbol{T_r}$ by compensating $\boldsymbol{T_b}$.

Outside of the activity recognition block two other modules are sketched in Fig. 3.20. The class and position of the object is determined by the *object recognition* module as presented in Sec. 3.2. The result of the object recognition is incorporated as context to draw the final decision regarding the conducted action in the *action recognition* module.

3.4.2 Region Tracking

Several different approaches for region tracking can be found in literature, each having different benefits and drawbacks [44]. To be applicable in an online system in realistic environments, certain requirements regarding processing speed and robustness needs to be fulfilled by the utilized approach. For the task outlined above, two different algorithms have been integrated and evaluated, that are both able to perform in soft real-time. Both are briefly introduced and are discussed in the following.

Hyperplane Tracking The first realized tracking algorithm utilizing the approach of *hyperplanes* [63] allows tracking according to different motion models. It can be designed to

either track only translational object motions as well as according to more complex motion models like rotational transformations as shown in Fig. 3.21(a). Basically, hyperplane tracking assumes the change of intensities of a set $\mathcal{R}$ of n selected pixels in the tracked region in an intensity image I to be explained by a transformation F

$$I(F(\mathcal{R}, \vec{x_t}), t) \;=\; I(F(\mathcal{R}, \vec{x}_0^*), t_0) \tag{3.15}$$

that is defined by the motion parameters $\vec{x}$ at a given time. $\vec{x}_0^*$ denotes the initial motion parameters of the region to track with respect to the image coordinate system. It is obtained from the image at time t_0. The goal is to determine the motion parameters $\vec{x}$ at a given time t. The idea of hyperplane tracking is to estimate a linear transformation A that basically represents a set of hyperplanes and allows to estimate the change of the parameters $\delta\vec{x} = \vec{x}_t - \vec{x}_{t-1}$ solely from the difference of pixel intensities as

$$\delta\vec{x} \;=\; A\left(I(F(\mathcal{R}, \vec{x}_0^*), t_0) - I(F(\mathcal{R}, \vec{x}_{t-1}), t)\right) \;. \tag{3.16}$$

This linear transformation A is estimated in a training phase at the beginning of the tracking process by applying a number small random transformations of the initial region according to the underlying motion model. Tracking based on this approach is fast and allows to track according to any motion model represented by F. It especially also allows to track rotation, affine, and projective models, but for the prize of higher complexity. The approach requires the tracked object to be textured and is quite sensitive to occlusion, because the motion parameters are estimated from intensity differences exclusively.

Kernel-based Tracking To overcome the limited robustness of the Hyperplane Tracker, another approach which is much more robust to occlusion and requires no time consuming training phase has been integrated: The kernel-based tracking proposed by Comaniciu et al. [31] which as been adopted and implemented by Bajramovic et al. [9]. They use color histograms as features to compute the similarity between the reference model and a candidate model and to estimate translation parameters $\vec{x}$. The term "kernel-based" has been assigned because the tracked region is represented by means of a convex and monotonic kernel function k. Comaniciu et al. [31] propose to use a kernel with Epanechnikov profile here, which is a simple linearly falling profile. The distance between a reference and a candidate model is defined on the basis of N-bin color histograms $Q^* = \{q^*(n)\}_{n=1,\ldots,N}$ and $Q_t = \{q_t(\vec{x}, n)\}_{n=1,\ldots,N}$, respectively. The kernel function is applied to weight pixel far from the center less than pixel close to it. The difference between to histograms is defined as

$$\begin{aligned} D \;&=\; \sqrt{(1 - \rho[Q_t(\vec{x}), Q^*])} \\ \text{with}\quad \rho[Q_t(\vec{x}), Q^*] \;&=\; \sum_{n=1}^{N} \sqrt{q_t(\vec{x}, n) q^*(n)}. \end{aligned} \tag{3.17}$$

The mean shift algorithm [30] is applied for an iterative minimization of this distance by optimizing the motion parameters $\vec{x}$. The approach is admittedly restricted to translational motion models (cf. Fig. 3.21(b)), but provides a very good tradeoff of high accuracy and robustness.

Evaluation for Activity Recognition In order to evaluate the different tracking approaches with respect to accuracy and robustness, synthetic image sequences, containing several patches of different objects that are moved around on a background image at different

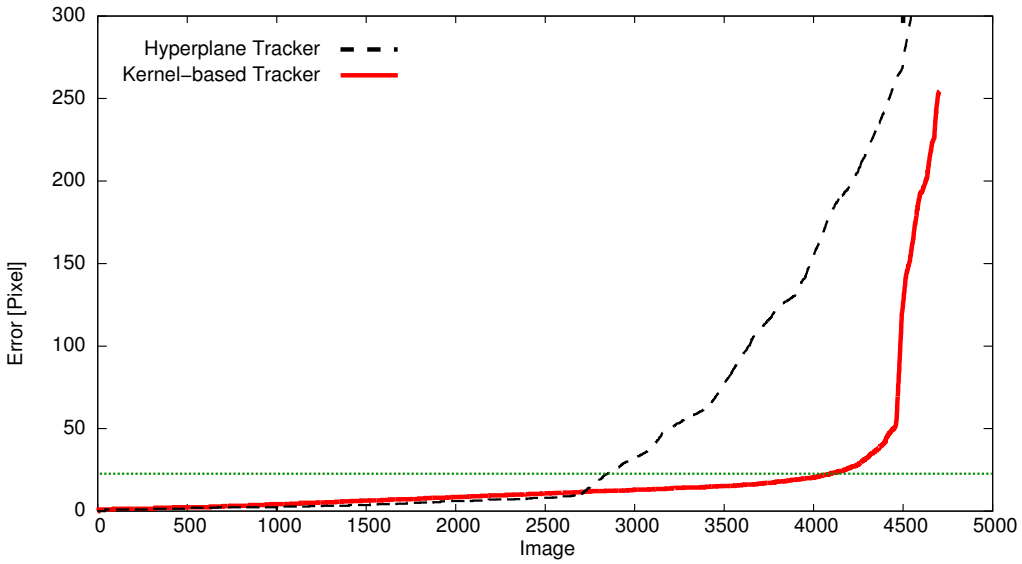


Figure 3.22: Absolute tracking errors.

speed in a controlled manner, are created. Additionally, the background image is shifted randomly to simulate the user's movement. By means of this, ground-truth data is directly available to evaluate the tracking approaches separately. Fig. 3.22 presents results of this quantitative evaluation study comparing the kernel-based tracker with its translational model only and the hyperplane tracker with a motion model accounting for translation and rotation. Tracking errors are measured as the Euclidean distance in pixel deviation for each image of the sequences and displayed in ascending order in the diagram.

In order to discuss robustness and accuracy, tracking errors of more than 20 Pixels are treated as a failed tracking (lack of robustness) and smaller errors as a lack of accuracy as indicated by the horizontal line plotted in Fig. 3.22. This threshold has been chosen with respect to the size of the tracked patch and the size of the background image. It can be seen that both trackers are quite accurate in more than 50% of all cases, but the hyperplane tracker exposes to be less robust as only 58% of all tracked frames are below the defined robustness threshold in contrast to more than 85% when applying the kernel-based tracker. As expected, for allowing a more complex motion model of the hyperplane tracker including rotation, the price of reduced robustness has to be paid.

As a qualitative insight, it can be stated that robustness is outweighing accuracy in the integrated system as the applied classification algorithm, that will be detailed later, can cope quite well with noisy data. Consequently, the kernel-based tracking approach proved to be more appropriate to recognize activities from captured object trajectories according to this study. It furthermore allows to start tracking immediately, because it requires no time-consuming training phase. The computational time effort of the two approaches has been compared by Deutsch et al. [44]. The measurements obtained in their study are shown in table 3.1 and support the selection of the kernel-based tracker for the given task.

Approach	Initialization [ms]	Tracking [ms per frame]
Hyperplane	528	2.33
Kernel-Based	2	1.03

Table 3.1: Computational costs of tracking approaches according to [44].

3.4.3 Background Motion Model

As mentioned before, computing the absolute motion requires the estimation of the user's motion to compensate it from the relative motion obtained by the region tracking module. Various approaches for the estimation of three-dimensional ego motion from image sequences are known in the literature (e.g., [156, 167]). In the given scenario, we have to cope with the fact, that a lot of perturbation is caused by the performed activity itself. Furthermore, no explicit metric reconstruction of the ego-motion is necessary as only the relative motion of the image with regard to the object motion needs to be computed. In the proposed method the image background is tracked according to an affine motion model based on individually tracked patches [143, 175]. Applying a Harris corner detector [69], feature points are selected and tracked from frame to frame as illustrated in Fig. 3.23. The algorithm estimates the affine motion for each image patch defined by the neighborhood of the feature points. Afterwards, *Least Median of Squares (LMedS) regression* [138] is applied to estimate the global background transformation $\boldsymbol{T_b}$. LMedS is known to cope well with outliers and thus allows to compute the global image movement neglecting local deviations caused by the movement of the object itself. This method works well as long as the majority of the tracked points lies on the background, which is a acceptable assumption for quite small objects. To fulfill this assumption it is also possible to remove all feature points, that are inside the tracked object region, from the estimation of the background motion. However, our studies, reported in Sec. 8.2.2, so far unveiled no need for this enhancement.

3.4.4 Classification

Given the relative object motion $\boldsymbol{T_r}$ and the background motion $\boldsymbol{T_b}$ as transformation matrix, the absolute trajectory can be computed as $\boldsymbol{T_a} = \boldsymbol{T_r}\boldsymbol{T_b}^{-1}$[5]. According to the introduction of the section, I consider activity recognition as movement classification. Therefore, the recognition of activities is based on this absolute motion. In order to recognize activities an adapted version of the fast and robust condensation-based trajectory recognition method (CTR) proposed by Black and Jepson [15] is applied. CTR is an extension of the

[5] $\boldsymbol{T_b}$ can always be inverted here, as it describes an affine transformation.

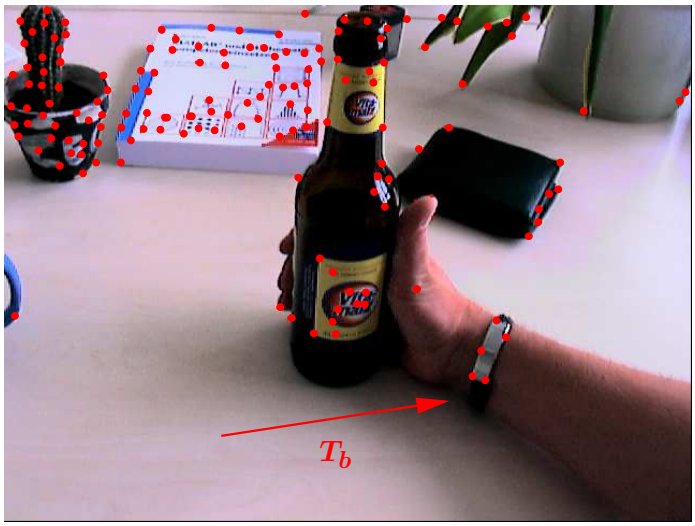
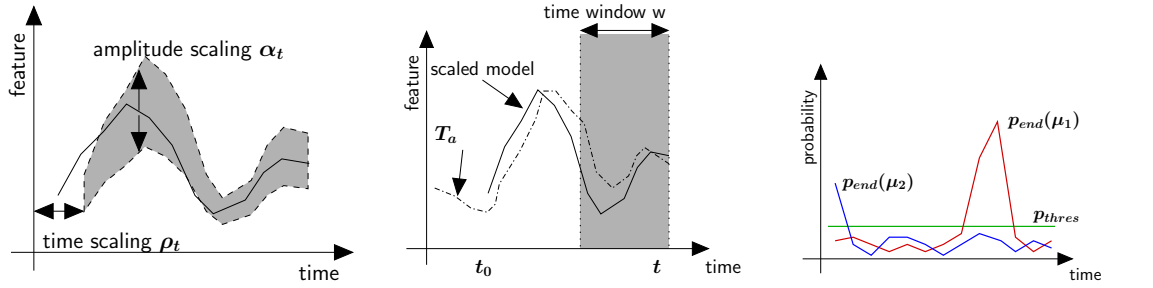


Figure 3.23: Background motion estimation from feature point movement.



(a) Model μ_i with time and amplitude scaling.

(b) Matching scaled model μ_i with input data T_a.

(c) End probabilities $p_{end}(\mu_1)$ and $p_{end}(\mu_2)$ with recognition threshold p_{thres}.

Figure 3.24: Scaling and matching of a model μ.

condensation algorithm developed by Isard and Blake [75]. The classification system itself has been developed by Fritsch et al. [55], to whose work I refer for more details regarding the algorithm.

Generally, the condensation algorithm compares several models μ_i – each representing a dedicated activity – with a currently observed feature vector $T_a(t)$ using a randomly generated sample set of size N. In this notion, $\mu_i(\phi)$ denotes the model sample feature vector at time $\phi = t - t_0$. In CTR, this model is constituted by a model trajectory that can be obtained by computing a mean trajectory from training samples. A model is propagated starting at t_0 and evolves over time as illustrated in Fig. 3.24(b). To cope with variance in the execution of the individual activity the model μ_i is scaled in time ρ_i and amplitude α_i as shown in Fig. 3.24(a) before model samples are generated by the condensation algorithm. The intervals to choose values for α_i and ρ_i from have to be measured from training sets to give most promising results. The quality of the match of such a scaled model – the sample n with its parameter vector $\mathbf{s}^n$ – and the measurement $T_a(t)$ is expressed in a weight $\pi_t^{(n)}$. The temporal characteristics of the activity is included by using the data in a time window w of several previous model steps.

Recognition of activities is performed by calculating the end probability $p_{end}(\mu_i)$ for each model i by summing up the weights $\pi_t^{(n_i)}$ of all samples representing that model with a matching position in the last $d\%$ of the trajectory length; $d = 10$ has been established as a good choice heuristically. A model is considered to be detected if the threshold of the end probability $p_{end}(\mu_i)$ for this model is reached as shown in Fig. 3.24(c). Hereby, an implicit rejection is implemented, because only completed and known activities are recognized.

Depending on the features provided by the tracking algorithm the CTR uses only the velocity Δx and Δy of the object or also its rotation $\Delta\gamma$ as shown in Fig. 3.21. The stochastic classification approach allows robust and fast recognition even on noisy input data, compensating a lack of accuracy of the chosen tracking approach to some extend.

Remarks

By means of the presented approach, action recognition can be realized to discern actions that differ either in the class of the involved object or in the typical motion of the manipulated object. The set of different activities that can be recognized by this approach is indisputably restricted, but provides a means to discern and detect all relevant actions in

scenarios targeted at in this work. In Sec. 8.2.2 results of the action recognition component in an exemplary object manipulation scenario will be presented. Furthermore, the two different tracking approaches will be contrasted when applied in the integrated system.

3.5 Summary and Contribution

In this chapter, I presented several approaches that enable a computer system to perceive its environment from an ego-vision perspective. These approaches all face the challenges of ego-vision perception, e.g., arbitrarily moving cameras, a limited field of view, and varying application contexts.

The chapter presented both, approaches that have been developed by project partners that we have adopted and integrated, and approaches I especially conceptualized for EVS. The first have only been discussed briefly and their particular assets and drawbacks with respect to the challenges have been presented. The major contributions in this chapter are the latter approaches that have especially been developed for the ego-vision assistance system.

The first unique approach presented utilizes mosaicing techniques to overcome the limited field of view, that is a consequence of the ego-vision perspective, in order to establish a compact, non-redundant pictorial representation of the scene. I proposed to compute mosaics of planar sub-scenes in order to cope with the problems of parallax and occlusion effects that have to be faced in case of arbitrarily moving cameras. The three stage architecture first decomposes the scene into approximated planes using stereo information, which in successive steps are tracked and integrated to mosaics individually.

Next, I discussed how objects can be detected and classified in an ego-vision system. One- and two-stage approaches have been distinguished. Methods belonging to the second type first segment an object from the background and classify it successively, while the one-stage approaches perform this functionality as one operational step. In consequence, two distinct approaches implemented by project partners [13] and in cooperation with Peters [123], respectively, are presented and discussed. Although both have their benefits with respect to ego-vision perception, we decided for the two-stage approach for most application scenarios, because it also allows to interactively teach new objects. This feature perfectly fits for the envisioned class of interactive assistance systems.

Afterwards, I presented how a pose tracking approach, which has been developed by Stock and Pinz [151] and other project partners, can be used to compute three-dimensional object positions and recognize view contexts. As for the mosaicing outlined before, a planar decomposition of the scene is applied.

The last perceptual method presented in this chapter focused on the recognition of simple actions from an ego-vision perspective. An action is determined by the class of the manipulated object and its movement. The class is obtained by the object recognition as outlined. Instead of performing a very complicated three-dimensional tracking from an ego-vision perspective to capture the object's movement, a more robust solution is proposed. Object detection results are used to initiate tracking of objects in order to capture their relative movement. As the absolute movement is required to recognize activities, the self-movement of the user's head and the camera, respectively, is computed and compensated. This does not allow a complete three-dimensional reconstruction of the object's movement, but provides a two-dimensional approximation, which proved to be appropriate for activities conducted almost in parallel to the image plane. For the implementation, I built upon visual tracking methods developed by Zinßer et al. [175], Gräßl et al. [63], and Richter [134]

in order to capture the movements of the object and the camera. The resulting trajectory is classified into activities using a condensation-based recognition scheme developed by Fritsch et al. [55]. In conjunction with the information about the class of the manipulated object, actions can be recognized.

By the concepts presented in this chapter, the cornerstone has been laid to facilitate perception in terms of a composition of the scene, as well as recognizing manipulations in it. All approaches follow ego-vision paradigms and can exclusively work with mobile sensory equipment, as envisioned for assistance systems.

4 Closing the Loop: User Interaction

People do little by reason, a lot by passion and most by habit.
[Chinese fortune cookie]

When reconsidering arguments given in Sec. 2.3, it becomes obvious, that communication between the system and the user must not be a one way. Shared attention, cooperation, adaptation, and learning require not only that the system is able to perceive the environment in a rather similar way as the user does. They also request, that the system can directly articulate its view and knowledge; simply, that it can interact with the user. Remember, that in ego-vision systems the leading concept is to integrate the user in the processing loop. This integration is illustrated in Fig. 4.1. Note the strong correspondence of this schematic drawing to Fig. 2.4. To enable interaction, the system does not only need to perceive the environment as explained in the previous chapter, but must also perceive and understand the user's instructions and articulations. To close the loop, an EVS must also be able to present knowledge, articulate its own demands, and, in consequence, render output that is

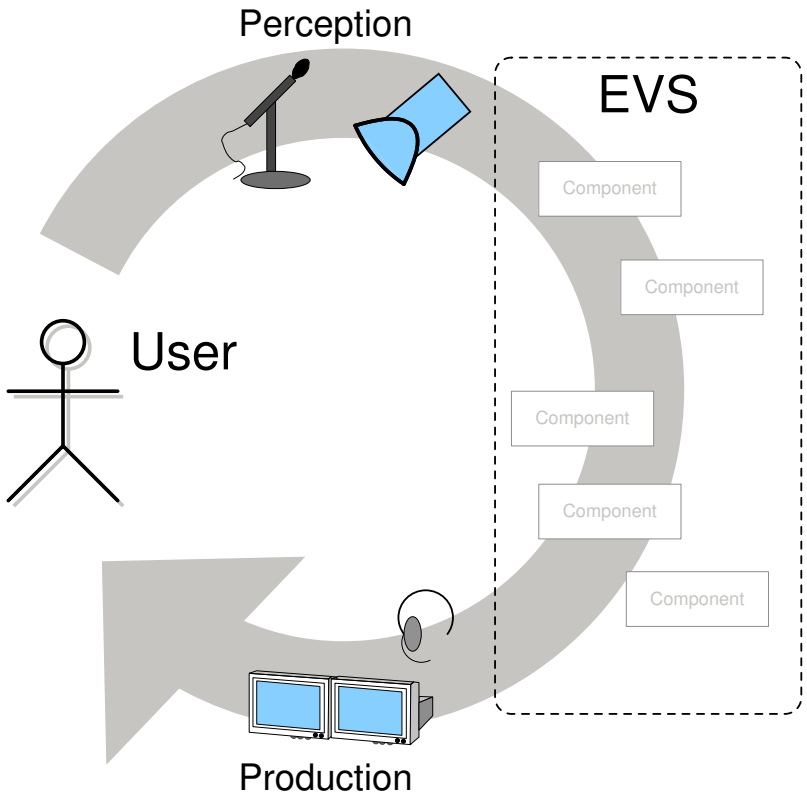


Figure 4.1: Interaction between the ego-vision system and the user.

accessible to the user.

In the presented approaches, vision is the major productive modality utilized by the system. In chapter 2 it has already been stated that augmented reality (AR) provides a good means to visually integrate the user. But the whole interaction is not only restricted to vision. In order to make interaction more efficient and the system more intuitive to use, other input modalities have also been integrated to develop multi-modal interaction capabilities.

What are typical use cases for interaction with the system in the envisioned assistance scenarios? First of all, interaction is required for online learning and tutoring of the system. An example is teaching new objects as outlined in Sec. 3.2. A new object is introduced to the system interactively by capturing different views of it. The views do not only have to be captured interactively, but also have to be labeled by a human tutor to let the system know the correct names. Even more, learning in a cognitive system never ends. It must be possible to correct the system, if it does something wrong in order to continuously adapt to new situations and constantly improve it. Second, to provide assistance, the system must be able to present information it has stored or learned to the user and give appropriate instructions or advices regarding the task. And third, the user of an assistance must be given the possibility to explicitly request information and also to be in control of the overall system behavior.

This chapter discusses these aspects beginning with a brief hardware description of the augmented reality device developed in the VAMPIRE project. This one serves as the system's physical interface and provides the necessary communication channels. Afterwards, I will discuss the role of *feedback*, *visual highlighting*, and *information presentation* for error recovery, cooperation and directing attention. It will be presented how these demands are accounted for by AR visualization techniques and how the different capabilities are implemented. Finalizing this chapter, the system's capabilities to perceive and interpret multi-modal input are presented.

4.1 The VAMPIRE AR Gear

Interaction with the user and integrating her or him in the system's loop requires special hardware setup. From the interaction point of view, this setup must allow to perceive the user's input to take commands, queries and so on, and also to articulate and present information back to the user in order to close to loop. The hardware device termed "*AR gear*" has been developed in the context of the VAMPIRE project [147] as a research prototype. It is utilized to implement an ego-vision system as outlined in this work. Fig. 4.2 gives an impression of the hardware setup. The AR gear comprises different sensors for perception and a head-mounted display (HMD) for visualization attached to a helmet to implement a mobile interaction device. Table 4.1 lists the main different hardware components and their relevant features. All algorithms and approaches presented in this thesis are realized

Component	Type	Specifications
HMD	I-visor 4400VPD	SVGA(stereo), 60/70/75 Hz VESA
WebCams	Fire-i	IEEE1394, 640x480, 15/30 fps
CMOS camera	TU Graz "i:nex"	1024x1024 pixels, 10 bit, up to 2600Hz
Inertial sensor	Xsens MT9	6 degrees of freedom, 100Hz
Wireless Mouse	Gyration GyroMouse	scroll-wheel and 3 buttons
Wireless Microphone	Sennheiser EM 1031-U	

Table 4.1: Components of the AR gear.

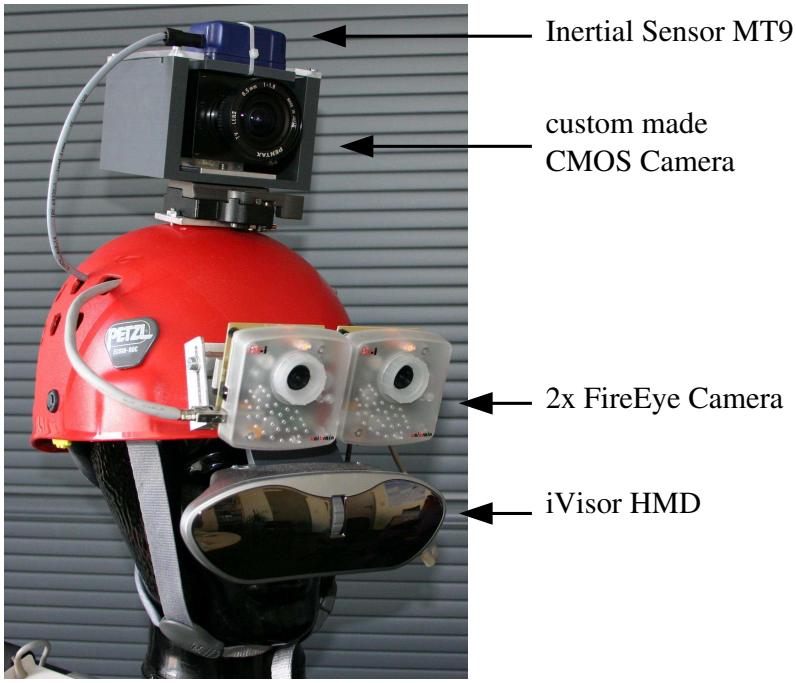


Figure 4.2: The VAMPIRE AR gear.

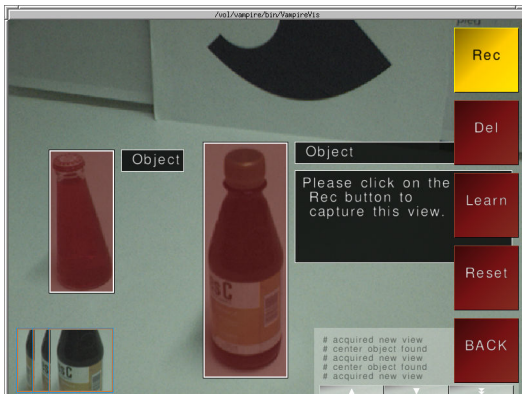
using the hardware incorporated in this AR gear.

The visualization capabilities in the AR gear are implemented by means of a video see-through system as introduced in Sec. 2.3. Stereo images are captured by two Fire-i cameras, processed by a Laptop with an OpenGL graphic chip (nVidia Quadro; hardware supported stereo) that allows to render overlays on the captured images which then are displayed in the head mounted displays (HMD). Video see-through allows to avoid the problem of exact positional registration of the user's view to the system's video capturing. It makes appropriate augmentation much easier. A disadvantage of this approach is the rather unnatural modification of the user's field of view, as she or he has the eyes virtually moved to a different position, namely the position of the cameras at the helmet, which requires at least some habituation. But having direct access to the user perception allows to really integrate the user into the loop and to ensure more easily, that user and system perceive the same.

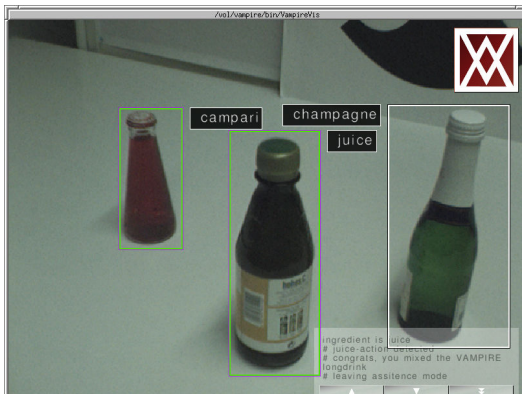
By means of the two Fire-i cameras mounted on the front of the helmet a stereo setup is available, that is also needed to compute the planar decomposition that has been presented in Sec. 3.1, for instance. Note, that these low-cost cameras are not suitable for exact photogrammetric measurements using stereo-vision, but still allow a robust decomposition and tracking as proved by the evaluation of the mosaicing module.

To support the visual self-pose recognition described in Sec. 3.3.1, the AR gear also comprises an inertial sensor that can be used to enhance and stabilize the result of the pose tracking process as proposed by Ribo et al. [132]. But furthermore, this sensor also captures data about the motion of the user's head on its own. In Sec. 4.3.2, it will be briefly outlined how this sensor can be used to recognize head gestures to implement a more natural input modality to interact with the system.

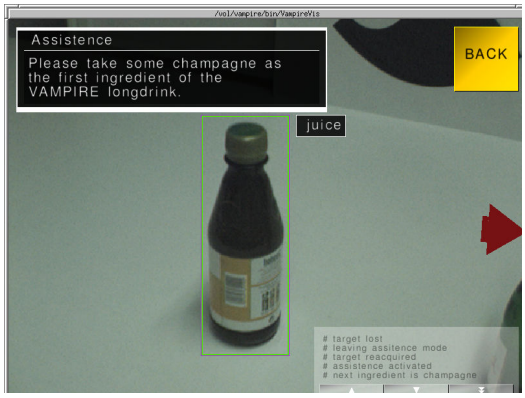
In addition, a wireless microphone (which is displayed in Fig. 4.2) is an integral part of the setup to allow speech input by the user as outlined in Sec. 4.3.1. Furthermore, a wireless



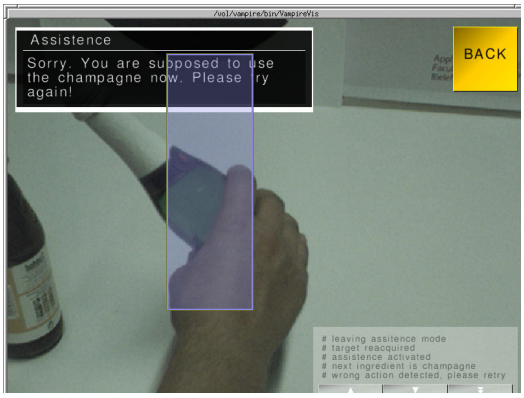
(a) Objects highlighted and captured views displayed.



(b) Displayed object recognition results.



(c) Spatial reference: The user is guided by means of arrows.



(d) Feedback: The blue box indicates the tracked object.

Figure 4.3: Visualization capabilities of the assistance system: Screenshots of the user's augmented view on the scene in some different use cases. These screenshots have been taken from the integrated assistance system described in chapter 7.

mouse is available as input device to control the system. As the goal of research in ego-vision assistance systems are *mobile* systems, only the scroll-wheel and the buttons of the mouse are used as input modalities, since usually no surface to move the mouse is available in such mobile settings.

4.2 Visualization and Feedback

Let us now take a closer look on the interaction capabilites of the envisioned EVS and how AR can be utilized here. As stated before, augmented reality in the presented system is following a video see-through approach. In order not to cause too long delays in the user's perception, the video see-through must be realized as efficient as possible. Therefore, OpenGL techniques are applied for the augmentation of the user's view. The so-called *visual interaction server (VIS)* realizes the video see-through loop and implements basic functionalities to augment the users view by different graphical elements of which some are shown in Fig. 4.3.

But what should actually be visualized, what is relevant to facilitate an effective and efficient

interaction with the user? Reconsidering the benefits of the user in the loop paradigm presented in Sec. 2.3, the following main use cases related to visualization can be identified.

4.2.1 Spatial Attention

In the context of ego-vision systems, the benefit of shared attention has already been discussed. It is a requirement for collaboration and interactive learning. Especially if the user and the system share the same view of a scene, visualization provides a simple means to articulate spatial attention. In our system the user's attention can be attracted to specific parts of the scene by highlighting these as exemplarily shown in Fig. 4.3(a). Here, a red overlay over an object is rendered in the view of the user. Hence, the system has a possibility to easily reference this part of the scene very efficiently and intuitively and attract attention. Furthermore, even parts of the scene that are not currently visible can be referenced by visualization. Arrows displayed at the borders of the field of view intuitively guide the user to places in the environment, the system wants to refer to (see Fig. 4.3(c)). With the current pose and viewing direction of the user being available by techniques presented in Sec. 3.3.1, these arrows can continuously be updated to always point to the correct 3D position.

4.2.2 Process Feedback

It is undoubtable impossible to have a computer system – or even a human – that will not make any errors or produce unpredictable results. Especially in an interactive system as envisioned in this work, every now and then the system has to face yet unknown objects, environments, and so on. If the system would not be able to communicate its interpretation of the perception it has, no correction imposed by the user could be triggered. And, even more perturbing, the user would not even notice, if or why something goes wrong.

Consequently, *feedback* plays a major role to facilitate user triggered learning, error correction and adaptation, as it allows the user to decide if further tutoring is necessary or not. An example of feedback is displayed in Fig. 4.3(b) in which the results of object recognition are displayed to the user by means of annotations indicating the coarse boundary and the computed label of an object. In this presented case, a user can directly ascertain that the system has correctly recognized all objects in the field of view. If this would not be the case, the user can trigger interactive learning of the respective object with the techniques presented in Sec. 3.2.2. Besides the annotation of objects shown in Fig. 4.3(b) for object recognition and in Fig. 4.3(d) for the object tracking described in Sec. 3.4, feedback can also be realized using textual visualization in either dialogs or in a "logging area" (on the lower right of each sub-figure in Fig. 4.3). Summarized, feedback provides a means for the user to look "inside" the system and understand why something might go wrong. It is a foundation to permit her or him to counteract appropriately, which facilitates adaptation and error-recovery.

4.2.3 Information Presentation

A third use case, which is most relevant for assistance scenarios, is the presentation of information or knowledge. Assistance is, as outlined in the introduction, concerned with interaction about the environment and therefore the system needs a communication channel to present its knowledge and advices in an appropriate way to the user. This way of

articulation can differ dependent on the kind of information. If the system wants to express where an object is located in space, the visualization using highlights and arrows, that is also used to draw spatial attention as outlined before, is probably most efficient in this case. Acoustic or textual instructions might be more suitable in other cases. Accordingly, the VIS provides different augmentation elements to articulate information:

- ▷ Arrows and annotated highlights (Fig. 4.3(c) and Fig. 4.3(b)) to refer to spatial positions.
- ▷ Images or parts of images (lower left of Fig. 4.3(a)), e.g. to display captured images, memorized views of objects and illustrative icons.
- ▷ Text displays (Fig. 4.3(c) and 4.3(d)) for the presentation of instructions and all other kinds of information that is not visualizable appropriately by the other elements.

Besides these elements, the visual interaction server is also capable to handle menus (on the right of each screen shot in Fig. 4.3) and dialogs that provide a means to directly control and instruct the system in a more administrative way. This elements are used to implement a simple graphical user interface (GUI) as known from all major computer desktop environments. The interaction with this GUI can be achieved in different ways, which will be tackled in Sec. 4.3.

4.2.4 Selective Visualization

Before turning towards input modalities, let us first take a closer look on what should actually be visualized. This question is relevant as a huge amount of information is generated in a running system. A visualization of "everything" is not only hard to realize, but also undesirable as it would cause an immense mental overload for the user. What is effectively needed, is an adequate selection of information to be actually presented to the user.

Obviously, any information that is directly requested by the user should be provided. A user who asks for the keys, for instance, should receive either a textual hint or guidance by means of arrows to find the desired object. But as stated in the motivation of assistance systems in chapter 1, it is desirable that the system provides the right information in the right moment without being explicitly asked for. In the definition of EVS, this feature has been introduced as *attentive* behavior. It requires *situation-awareness* of the assistance system. A specific situation should directly affect the information presented to the user.

The objects the user pays attention to, the spatial context, and the actions performed, all together define the situational context that can guide the decision what is to be visualized. In the integrated system, a component termed *context-aware visualization* is envisioned to control the visualization of information. One of its purposes is to present object recognition results to the user, only if it fits the current situation. To explain this more in detail reconsider Ex. 3.4 on page 52, that has been introduced to explain the concept of view contexts. In that given example the object annotation has been activated or deactivitated depending on the current view context. The *context-aware visualization* considers the viewcontext and triggers the visual interaction server to actually present the object annotation to the user if necessary.

Despite all the things said in this section, the selection of the correct information to be presented to the user is highly depending on the respective task and situation. Accordingly, it is hard to find a general solution to this problem. But it will be shown in chapters 7 and

refchap:evaluation, how the adequate integration approach facilitates the analysis of the situation. This allows a quite simple and straight-forward implementation of information selection modules as the outlined context-aware visualization component.

4.3 Input Modalities

As illustrated in Fig. 4.1 and hinted in the last sections already, interaction also requires to perceive the user's input in terms of commands, tutoring, and requests. When developing assistance systems, it is important to keep in mind that the hands are usually bound to the accomplishment of the desired task, and that interaction with the system ideally should not occupy them. Consequently, we abstain from using a keyboard as the main input device in the assistance scenario as it has several disadvantages. First, a regular keyboard is rather bulky and so quite inappropriate for a mobile assistance system. Second, handling a keyboard occupies the hands. Consequently, interaction and task accomplishment cannot be done in parallel. Instead another, more natural, way of interaction needs to be implemented. Therefore, it shall be considered that different modalities or forms of communication are appropriate depending on the type of interaction required. It might, e.g., be rather laborious to control a graphical user interface, that is shown in the HMD, by verbal spatial description like *"choose the third button from the top on the left side"*. As known from modern desktop environments, direct spatial navigation, e.g. using a mouse, is much more comfortable in that case. But often free textual input is the most natural way of communication, for instance, to pose a request or question to the system.

Consequently, different input modalities are realized to allow efficient interaction with the system and facilitate its control. The major input modalities are outlined in the following, whereas some are just an integration of existing works that are mostly taken as is and hence only presented very briefly. The interaction scheme using head gestures however is a novel approach discussed a little more in detail.

4.3.1 Speech

Speech can be considered as one of the most natural and also comprehensive communication modality of humans. Accordingly, natural speech provides a powerful means also to facilitate efficient human-machine-interaction, and is hence integrated as an input modality in assistance systems. In the implemented assistance prototype fast and robust speech processing is achieved by combining speech recognition on the basis of hidden Markov models (HMM) with a task-related semantic-based grammar, which is used to extract instructions and corresponding information from the speech input. The ESMERALDA framework [52] is utilized for the speech recognition and understanding solution. It allows to combine HMM to model the acoustic charateristics of natural speech with a grammar to robustify and interpret the classification results.

The speech recognition and understanding component allows to ask questions to the system, answer simple yes/no questions, tutor the system by assigning labels to objects and scene that are seen, and ultimately allow to talk about the visually perceived scene. The current implementation of speech recognition and understanding is admittedly rather inflexible and needs to be redesigned depending on the task the system is applied in. Although, this thesis focuses on visual perception, speech recognition is inherently necessary to implement an appropriate interface for the user in order to evaluate the system as a whole (cf. chap 8).

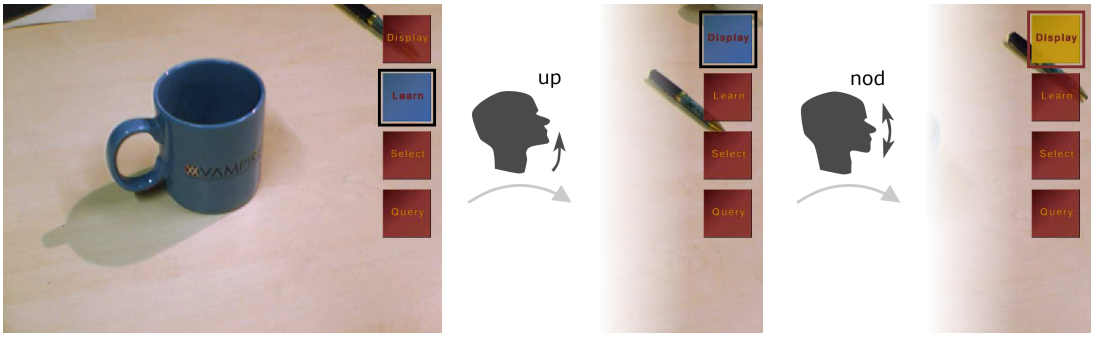


Figure 4.4: Screenshots showing the control of the GUI menu using head gestures. By means of gestures 'up' or 'down' the user switches between menu items. Selection is done by performing a 'nod' gesture.

4.3.2 Head Gestures for Interaction

Another approach to implement a meaningful input modality that allows to keep the hands free is to use head gestures for a limited set of semantics. Since head gestures provide a wide range of conversational functions, they have been studied in human-machine interaction research quite a while. [41, 82, 87, 150]. All these contributions present interfaces where head movements are recovered from externally captured video data. Using computer vision, human users are monitored and their head pose is tracked in oder to detect and discern head gestures. Though it mimics the way a human communication partner would perceive head gestures, this approach has obvious limitations. The required external sensing (i.e. the static camera) restricts applications relying on head gestures to prepared environments, for instance smart rooms, and stands against the paradigms of the ego-vision approach. Finally, automatic visual head gesture recognition is not a trivial task; it requires considerable computational effort, yet robust pose estimation cannot be guaranteed.

As an alternative, the inertial capturing device, that is anyway part of the AR gear (cf. Sec. 4.1), can be utilized as data source to allow interaction by head gestures. This digital motion tracker registers 3D rate-of-turn and acceleration at frequencies of 100 Hz and with a noise level of less than 1° in orientation. It therefore provides a convenient means for accurately measuring head motions in real time. And can be used to discern a wide range of different head gestures useful for human-machine interaction as outlined in [66]. In that work, different types of head gestures are discussed, while for the interaction in the assistance system a limited set of gestures is sufficient.

We apply head gestures to control the menu of the GUI and express affirmation or denial semantics. Accordingly, head gestures to express *up* and *down* to navigate in the vertical menus, and *nod* and *shake* to express affirmation or denial, respectively, need to be discerned as illustrated in Fig. 4.4. As the user will not only perform these gestures but move the head also in any arbitrary way while using the system, robust rejection of other head motions is necessary.

The approach utilized to classify different head gestures is based on semi-continuous linear hidden Markov models (HMM) with Gaussian mixtures to parameterize observations or measurements. HMMs are especially very well suited for the classification of time series that vary in their duration. They are nowadays quite famous in speech recognition also, as stated above. For the recognition of head gestures, the ESMERALDA software package already utilized for speech recognition written by Fink [52] is applied again with some

adaptations done by López Romero [96] and Hanheide et al. [66].

The raw data available from the employed inertial sensor supplies six dimensional samples of two attributes (each three dimensional) at a rate of 100 Hz: "rate of turn" $[\frac{deg}{s}]$ and "acceleration" $[\frac{m}{s^2}]$. These features constitute the raw input data $\boldsymbol{s}$ for our classification approach. Although these samples could be applied directly to the HMM-based classification algorithms, previous experiments have shown that features describing the characteristics of the dynamics allow for much better recognition performance. Here, *Fourier descriptors* $\boldsymbol{F_n(t)}$

$$F_n(t) = \frac{1}{K}\left|\sum_{k=0}^{w-1} s(t+k)e^{-2\pi ink}\right|$$

are applied as features computed from sliding, overlapping windows of the original six dimensional inertial data $\boldsymbol{s(t)}$. In the above equation the window size is denoted as $\boldsymbol{w}$. Best results could be achieved with a window size of $\boldsymbol{w = 10}$ $(\boldsymbol{100ms})$ overlapping by $\boldsymbol{\delta = 6}$ $(\boldsymbol{60ms})$ to capture the dynamics of head movements. This allows to compute a maximum number of $\boldsymbol{w}$ features, but it is expected that descriptors with higher $\boldsymbol{n}$ are bound to jitter, since they are only bound to high frequencies in head movement. Accordingly, we choose a smaller number of only $\boldsymbol{n = \{1 \ldots 4\}}$ descriptors. The feature extraction with the above values results in a **24**-dimensional feature vector. To estimate the parameters of the Gaussian mixture models, $\boldsymbol{k}$-means vector quantization [98] is carried out on a set of training features. Afterwards HMM parameters are trained from manually annotated data using the Baum-Welch-algorithm resulting in a set of HMMs; one for each trained head gesture. All trained HMM are evaluated on the input observation data in parallel using the Viterbi-algorithm. The HMM with the highest probability is taken as classification result. Rejection is implemented by comparing the probability of the HMM encoding for head gestures with dedicated rejection models, that have also been trained and evaluated similarly.

4.3.3 Wireless Mouse

The above introduced input modalities, speech and head gestures, both utilize pattern recognition rechniques in the common sense. They both suffer from errors in the recognition process, which can hamper interaction in a real scenario. Even if both are generally efficient ways of communicating with the system, it might occur that, in a given situation, they fail to understand correctly what the user wants to express. In order to let the system still be usable, a wireless mouse is integrated as a "fallback" control device that allows the user to access major system functions. As stated before, only the scrollwheel and the buttons of the mouse are used to navigate through menus, e.g. substituting for head gestures in Fig. 4.4, and select options in dialogs. Instead of saying a command again and again because the speech recognition is not able to understand it correctly, the mouse control often provides a shortcut to reach an interaction goal, although one hand must interrupt its current task. In the user studies presented in Sec. 8.4 mouse and speech input are compared with respect to their interaction appropriateness.

4.4 Summary and Contribution

This chapter shed some light on the interaction capabilities of an assistance system following ego-vision paradigms and applying augmented reality techniques. I presented the basic

visualization capabilities of the system that are necessary to (i) direct attention, (ii) provide feedback of internal processes, and (iii) to present information to the user. Furthermore, the problem of deciding *which* information should be visualized has been touched. The presented visual interaction server has been developed jointly with Siegl and Pinz [144] and Wrede [168] in the VAMPIRE project, as well as the outlined component for context-aware visualization.

As interaction is essentially a two-way communication, different input modalities that allow the user to articulate commands, information, and requests to the system have been presented. I especially developed a method on the basis of hidden Markov models to use head gestures as an input modality on the basis of the AR gear. Other input modalities like speech [52] and a wireless mouse have been integrated in the system requiring only small adaptations.

Together with the perception capabilities described in chapter 3, the interaction capabilities presented here allow to close the loop with the user.

5 The Visual Active Memory

The true art of memory is the art of attention.
[Samuel Johnson, 1709–1784]

In the previous chapters different issues of ego-vision with respect to perception and interaction have been tackled. Now, the question, how these different functionalities can be composed to an integrated system, shall be answered. Several different issues, which have exhaustively been discussed under the term *cognitive vision system* in Sec. 2.2 already, have to be faced. Accounting for the different attributes of CVS like *learning* and *situation-awareness* and the demanded flexible processing paths make the design of such systems a challenging task in itself. Granlund [61] has identified some of these challenges. As a first one, he identified the huge amount of data resulting from vision as primary perceptual cue and the necessity to interpret this data to achieve compact representations. A second challenge constitutes from the requirement to realize and combine different inference mechanisms; either goal-driven (top-down), or event-driven (bottom-up). And finally, a cognitive vision system is fragmented into different components implementing different functionalities. Accordingly, flexible strategies and appropriate representations need to be developed and must converge into a conceptual architecture for such cognitive vision systems.

In the following, a ***Visual Active Memory (VAM)*** [162] will be motivated. It is motivated from insights of cognitive science and demands arising from computer vision, assistance systems, and ego-vision. This memory adopts concepts of a central memory for system integration as first proposed by Wrede et al. [170]. An ***Active Memory Infrastructure (AMI)*** [171] provides the basis for the realisation of this VAM. This chapter will not focus on the functional and technical benefits of the AMI for software integration in general, but will briefly describe some of its concepts as far as necessary to understand the approach of the visual active memory. It will instead focus on the *representation* of information in such a memory in order to facilitate the interplay of different components, learning capabilities, adaptive behaviors, and to cope with uncertainty on a generic level.

5.1 A Memory for Cognitive Vision Systems

The term "*memory*" already appeared several times in this work so far and it has been said that a memory is utilized as the basis for the integration of the envisioned systems. But why a central memory can be useful and how it should be structured is still an open question. In the following a closer look at the requirements and the conceptual design of such a memory from the perspective of cognitive vision systems shall be given.

5.1.1 Why a Memory?

Let us start by asking the obvious question regarding arguments voting for some kind of memory in cognitive systems. Besides the fact that one would generally agree that it is important not only to consider the current moment in time, a more specific, threefold view on the subject is given in the following to answer this question.

Ego-Vision In Sec. 2.4 *ego-vision systems* have been introduced as computer vision systems that forgo any external or stationary sensory. An EVS perceives its environment exclusively using wearable sensors like head-mounted cameras. In order to gain a more complete model of the environment some kind of memory that accumulates the perception obtained from different perspectives is almost indispensable. A Memory can compensate for the restricted field of view of such a system. Note, that this accumulation can basically take place on different levels. On the one hand, considering images for instance, methods have already been presented that memorize on a pictorial level by means of the *mosaicing* approach presented in Sec.3.1. On the other hand, the accumulation can also take place on higher levels based on interpreted perceptions. As an example, the perception of a cup seen at a specific position is memorized and can be compared to a new perception. Based on this memorized information about the cup, the ego-vision system can decide whether to assign this perception of a cup to the already known one, or to memorize it as another cup at a new position. A solution to this general data fusion problem is discussed later on in Sec. 6.4 under the term "hypotheses anchoring". Note, that the position of a cup can be recalled from the memory, even if it is not in the current field of view of the system, allowing to virtually enlarge the awareness space of the system. A memory is thus a prerequisite for EVS to build up a global model of the environment and to compensate for the limited perception capabilities.

Assistance Systems Also from the view point of assistance systems in general, there is demand for a memory. The major task of any assistance system as defined in chapter 1 is to articulate knowledge that is either pre-modeled or interactively obtained. Accordingly, an assistance system demands for memorized knowledge. Knowledge in the sense of cognitive assistance is manifold. Consider the knowledge about the constitution of a specific task like, e.g., the sequence of different actions to be conducted to prepare a coffee. An assistance system needs to know the sequence of these actions like pouring water in the machine, filling coffee beans, pressing the start button in order to provide assistance in such a scenario. All this knowledge must be available in an adequate representation and has to be articulated in an appropriate way. For the added value of an interactive assistance system, it must also be able to compare that knowledge to the user's actions in order to understand what the user is doing as discussed in Sec. 3.4. The higher level concepts of tasks need a representation that allows to compare and recognize them on the basis of perceptions and acquired information. But the memory is also required to store not only models of actions but also knowledge about objects, user preferences, and others more. For assistance systems, it is most important that knowledge can be acquired and memorized, but furthermore also to relate this knowledge to perceptions and informations of a current situation.

Cognitive Systems The definition of *cognitive systems* generally also implies a demand for a memory. It can generally be claimed that memorization capabilities are key attributes of any cognitive being. It is a prerequisite for any form of learning, adaptation, and flexibility

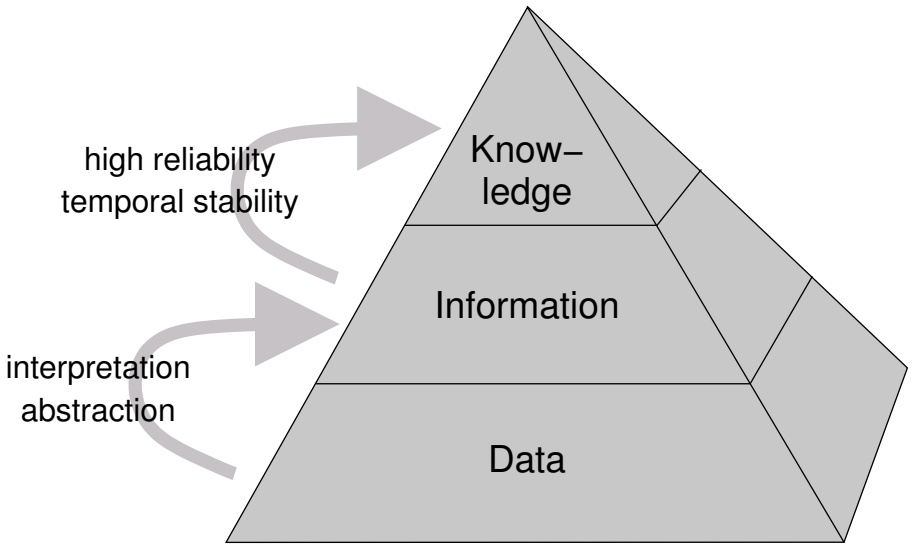


Figure 5.1: From data to knowledge.

as stated by many psychologists, e.g. [8]. In the sense of cognition, a memory is the foundation for any further cognitive processing. The ability to recall previously obtained knowledge and to relate perceptions to experiences is fundamental for cognitive beings. But even in short reactive cycles, when information is processed almost immediately, a short-term memory is required to hold the interim results of the processing queue. As a consequence, a system that should at least partially develop cognitive abilities needs a memory. Is has become apparent that cognitive systems demand for close coupling between representation and processing [59] to facilitate learning and extensibility. In consequence, cognitive systems require an *active* memory that directly affect processing by all means. Therefore, any memory content should fundamentally be available for all the different processes involved in a cognitive system. In consequence, the memory itself is a central cognitive function in such systems.

5.1.2 Data, Information, and Knowledge

The relevance of a memory and its content has already stressed. Now, a closer look at what is actually stored and exchanged in a memory shall be given. Inspired by a definition of Aamodt and Nygard [1], the definitions of *knowledge*, *information*, and *data* in cognitive processing are discussed with respect to the representation of content in a memory. Aamodt and Nygard [1] define the following distinctive features:

Data are syntactic entities. Data patterns carry no meaning are usually processed on a very low abstraction level; associated with a very specific time and state.

Information is interpreted data with meaning and can undergo further semantic interpretation. Information is often created as output from data interpretation.

Knowledge is information involved in reasoning and interpretation processes; it is often the output of a learning process. Knowledge is usually valid for a longer period and not only bound to a specific moment in time and space.

Figure 5.1 sketches these three levels in a pyramid to emphasize their mutual relation. To make this three levels and their dependencies more explicit, the path from the lowest level of the pyramid to its peak is traced in the following.

From Data to Information

Let us first look at the lower levels of the pyramid in Fig. 5.1. In a vision system, there are generally many different sources of *data*. Basically every sensor provides data of a specific domain that differs in dimensionality and amount, complexity, etc. Data is bound to a specific time and carries no meaning in itself. But data can of course be exchanged, stored and – most important – interpreted. By interpreting data, *information* is generated. This interpretation process is a challenge usually tackled by *pattern recognition* research.
Example 5.1: *The personal assistance ego-vision system captures an image of Zoe's keys when she is looking at them. This image of her keys is considered to be data. When she now wants to train the system this keys, the image data serves as input to train an appearance-based object recognition algorithm. After the object recognition has been trained it can interpret the image data and in consequence generates information in terms of a symbol "keys".*
Information has *semantics* and imposes already some level of abstraction and simplification. When a vision system perceives the world, information is generated from captured images by applying certain models that allow an interpretation of the image data. Note, that the interpretation does not have to consider only a specific instant, but can consolidate perceptual data over time. Although information in this sense is bound to a specific situation, it can be memorized for a longer period of time in order to recall specific events that occurred in the past. This emphasizes a strong correspondence to the *episodic memory* [157] known from cognitive neuroscience, that is also often termed the *event memory*.

Information can form entities that are exchanged between processing component in a computer system or stored in a kind of memory. It is a feature of cognitive systems that these entities are often still associated with the underlying data to facilitate reconsideration and hybrid processing.

From Information to Knowledge

In the notion of Aamodt and Nygard [1] information is bound to a current situation and mainly a semantic interpretation of the current perception. When there is no support for information in any sense, information becomes invalid and useless. Thus, information is considered to be very *specific*. The life time of information is thus relatively short and bound to perceptions at a given time.
Example 5.2 (continuation of Ex. 5.1): *If the system has perceived the keys on the table, using for instance the object recognition approaches outlined in Sec.3.2, this is considered to be information. But the appearance model of the object that allows to classify keys is a result of a training process that generates knowledge.*
In the definition of Waltz [165], the above example describes the application of knowledge to interpret the data in a deductive reasoning process. In this notion, the interpretation of image data as an object (e.g., a cup) generates information, while the model of how a cup looks like constitutes knowledge.

But how can knowledge be created or established? Of course, knowledge can be pre-defined in some way externally by a human expert. But Waltz [165] also discusses the emergence of

knowledge as a discovery process of previously unrecognized patterns. Learning and data mining processes are thus applied to generate knowledge from information and data. This definition subsumes the acquisition of specific knowledge for interpretation to, e.g., recognize objects, as well as more abstract knowledge relating different kinds of information.
Example 5.3 (continuation of Ex. 5.1): *By frequently observing the user putting the keys on the table, the system can establish a relation between the two concepts "table" and "keys". So when the user is searching for the keys, the system can presume that they are on the table, even if they are not visible.*
This knowledge about the keys being usually on the table emerges from continuous observation, re-occurrence of certain patterns and the detection of significant correlations. Knowledge is the set of concepts about the appearance, relations, dependencies, and roles of entities in the world.

As a consequence, and in contrast to information, knowledge does not need constant support to stay valid once it has been established. It is either build upon learned or pre-given models that are much more *general* and stable than information. Knowledge in general has a much longer life time, but it should nevertheless not be treated as an irrevocable fact. It might however become necessary to revise even knowledge, if the assumptions made to establish it become invalid.

5.1.3 Conceptual Design of the Memory

The differentiation into "data", "information", and "knowledge" has been introduced to motivate a structured representation of memory contents. But it is a conceptual assumption of cognitive vision systems in general, that these borderlines between data, information, and knowledge are more porous than indicated by the perspective outlined in the last section. The crossovers between the different levels in the pyramid in Fig. 5.1 are especially in the focus of such systems. The amount of predefined knowledge should be reduced to a minimum in favor for learned and dynamically acquired knowledge, to enable adaption and flexibility. Furthermore, cognitive systems shall *actively* apply the knowledge they have and gather new knowledge by means of interaction. Thus, a cognitive vision system must not consider the levels independently, but must comprise a concept to integrate them under one roof. In the following, the *visual active memory* is introduced as an all-embracing conceptual architecture allowing to represent and process "data", "information", and "knowledge" in an integrated approach.

Although the borderlines are quite porous, the insights obtained in the previous section still motivate some hierarchical structuring of a memory, in general. Neuro-physical studies support this hypothesis of a hierarchy. It is assumed that memorization is generally a time-dependent process that can be coarsely divided into a short-term and a long-term memory. Based on content, the memory is structured as a hierarchy where the content of the higher system is at least partially grounded in lower systems as indicated by Tulving [158]. He proposed a simple model termed *SPI* for the organization of cognitive memories. According to this model, information is encoded *serially (S)* into this hierarchy in the notion of Tulving. Output from one system provides the input to another. But however, information is stored in *parallel (P)* on the different layers. For instance, the holistic visual impression of a scene is as well stored as its interpretation into distinct objects. Finally, content can be retrieved from the memory *independently (I)* by all systems.

Following these ideas, a memory design is proposed accounting as well for the hierarchical organization in memories, as also for the specific requirements of cognitive ego-vision sys-

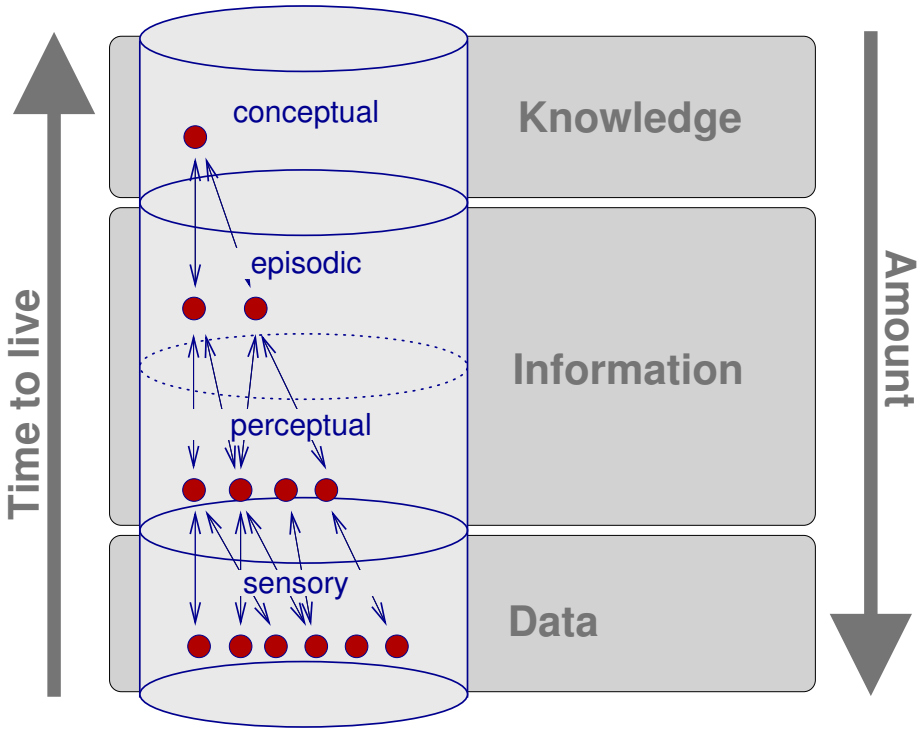


Figure 5.2: Conceptual layers in the memory.

tems. Hence, the distinction between data, information, and knowledge is subsumed and generalized by means of a four-layer design of an integrated memory shown in Fig. 5.2. As the concepts of a hierarchical memory are developed along the arguments of the previous section, the rough correspondences between the three levels, "data", "information", and "knowledge", and the four layers in the memory concept are indicated by boxes in the background of the figure.

All entities stored in the memory are schematically drawn as circles in the figure and termed "***memory elements (ME)***" in the following. Between these elements, arrows are drawn to indicate mutual links. Such links can result from inference processes, dependency analysis, or just to express cause-effect relations and grounding.

Memory Layers Starting from the bottom of the hierarchical structure, the *sensory memory* is introduced. In an ego-vision system, it is necessary to memorize raw or pre-processed pictorial data to compensate the restricted field of view. But furthermore, in order to facilitate, e.g. object learning as introduced in Sec. 3.2, the memory must be able to store image patches to utilize these as training material. This sensory memory basically corresponds to the level of *data*. As a special case, this sensory memory corresponds to a *pictorial memory* in a vision system.

Above, the layer of *perception* is introduced. In analogy to the step from data to information described in the previous section, data is interpreted to result in perceptual memory elements. Often links between the interpreted data and the data itself are useful in order to reflect the interpretation at a later time. In a computer vision system, the perceptual layer contains interpretation of visual input. Examples for this layer are results from object or action recognition as they are generated by the respective approaches discussed in chapter 3.

In the previous section "Data, Information, and Knowledge", it has been stated that information can consolidate perceptions over time and is in correspondence to the concept of a *episodic memory* also known from cognitive neuroscience. Accordingly, another layer is introduced which conceptually also belongs to the abstraction level "information". The major difference between the perceptual and the episodic memory lies in its duration. While the content in the first is only valid at one specific moment in time, the latter constitutes information about a longer episode and can subsume several percepts. It can be seen as a collection of events that are represented with respect to time and place.

On the top of the memory a layer termed "conceptual memory" is designed that directly corresponds to "knowledge" in the notion of the previous section. It contains concepts that result either from learning processes based on episodes and interaction, or are pre-modeled. The conceptual widely corresponds to the *semantical memory* known from cognitive science.

Cognitive Foundations It is a basic cognitive foundation of memories, that they have a limited capacity [8]. This it not only a natural restriction in cognitive beings, but also a major requirement for cognitive systems to work. In order to allow efficient access to information, it is necessary to focus on the relevant pieces of information. As the quote at the very beginning of this chapter suggested, the art of memory is thus also an art of attention. Accordingly, *forgetting* is an active process in memories, that discards stored information regarding its reliability and relevance. Forgetting is a cognitive function for its own that is dedicated a more detailed discussion in Sec. 6.6.

Time is a discriminative feature of memory elements in the different layers that also affect forgetting. Memory elements have a mean *time to live* which for the lower layers usually is relatively short, due to their short validity time as sketched in Fig. 5.2. Only when linked to some elements in higher layers, these might still be useful and memorized for a longer period of time. In contrary, concepts usually have no expiration time and are kept in memory almost always.

Opposed to the time to live, the amount of memory elements in the respective layers can be considered. As elements on the data layer are only kept for a quite short time, a high fluctuation can take place on this layer and many of these elements can be stored. On the contrary, the fluctuation and amount of memory elements is much smaller in the episodic and conceptual memories.

This four-layered memory concept provides the foundation for the development of the visual active memory concept architecture. The differentiation of "knowledge", "information" and "data" inspired by Aamodt and Nygard [1] is substituted in favor of the "VAM perspective" with its four-layered architecture as described above. Note especially, that "knowledge", "information" and "data" are rather general terms, and will accordingly be utilized in a more general meaning in the remainder of this thesis, and will no longer be restricted to the notion of Aamodt and Nygard [1].

5.1.4 Requirements

In order to account for the different viewpoints on a memory outlined in Sec. 5.1.1, and considering the layers of abstraction and the conceptual architecture introduced before, some general requirements can be identified for the conceptual architecture of a visual active memory.

Two requirements that are relevant with respect to the VAM architecture are already ful-

filled by the underlying AMI framework [168].

Flexibility If a central memory is envisioned that is able to store entities of the different memory layers, a significant level of flexibility is required. The memory must thus be as *general* as possible, but still be able to represent the differences in semantics between this levels of abstraction. The representation must respect specific features of stored entities like their size, complexity, and structure. The AMI therefore proposes XML [14] as the representational basis of memory elements as a solution to this issue.

Availability As stated several times, processing path are not fixed in a cognitive vision system. In general, any component can develop a demand for specific pieces of information. Although not all memory elements are needed by every component all the time, a general availability of information for all the components is necessary to be taken into account by the integration approach. Processes must be able to query for specific information if they require at any given time. The AMI directly considers this requirements by its proposition of the memory as a central component, that keeps data available for processes whenever they need it.

The framework AMI will be briefly presented in Sec. 5.2. But there are still some requirements left, that have to be considered for representation of the memory and the coordination of process flows in the system. These are not directly reflected by the integration infrastructure, but need explicitly to be respected when developing an integration concept on the basis of an active memory.

Reliability When facing real-world perception and interactive systems, also uncertainty and errors have to be considered. Basically, the memory should not contain nonsense and erroneous information. But one cannot be generally sure, whether some information is correct or not. There might be some content in a memory, that is very reliable, but also some other that needs to be revised. It is however not an easy task to identify memory content as erroneous and to decide what should be kept in the memory and what should by discarded. Especially accounting for the limited capacity of a memory it is important to ensure a certain level of reliability in the stored information. A mechanism to identify unreliable and irrelevant content and to remove it is necessary. A mechanism responsible for removing it has been introduced as *forgetting* before, but it still has to be discussed how reliability and uncertainty can be expressed in the memory, and how the system should cope with it.

Compactness In order to allow scalability and to keep the system as efficient as possible, redundancy should be avoided in the memory. This means, that content should ideally not be doubled in the memory, which demands for a mechanism to compare new information to the content already stored in the memory. In conjunction with the reliability requirement outlined right before, compactness also regards the limited capacity of the memory.

Maintainability As a key feature a memory is able to store entities over an extend of time. This holds especially for the episodic and conceptual memories. In conjunction with the capability to provide a compact content it is furthermore important to account for the chronological process of the acquisition of memory content. The *history* of a memory element is relevant and should be preserved. New perceptions are related to existing ones and can substitute information in the memory with more recent information. But uncertain new input should not automatically trigger to forget

older but probably more confident information. Accordingly, it is important to allow memory elements to be updated even partly and have a structured way to trace the history of a memory element. However, updating a memory element is always preferred to the generation of a new one. Thus, whenever additional information regarding a memory element is available, it should be added to the element.

These fundamental attributes of the proposed memory will be reconsidered several times in the remainder of this work. It will be shown how these can be realized either by the active memory infrastructure itself, or by an interplay of different components working on the memory content.

5.2 The Active Memory Infrastructure

In order to implement the visual active memory, I utilized the integration framework proposed by Wrede [168], the *active memory infrastructure (AMI)*. This infrastructure has been developed to facilitate the integration and coordination of intelligent systems on the basis of a central component, an *integration broker*. In the presented concept of a visual active memory, the integration broker is utilized to implement the active memory. The underlying infrastructure already provides the necessary *flexibility* and *availability*, which have been identified as requirements for the VAM in the previous section. In order to provide flexibility, XML [14] is utilized as underlying representation for data exchange and storage. A distributed architecture and an event notification scheme facilitates the availability of memory content for different processes.

The following explanations should not describe the AMI in detail, but should familiarise the reader with the concepts of this approach of information-driven integration. The *active memory infrastructure* (AMI) mainly consists of

(i) the *active memory* as an integration broker, that is built on the basis of a database back-end, and a dedicated client interface provide components access to the memory instance [170].

(ii) a coordination component [163] based on petri-nets [124] that allows a state-based control of the integrated system,

(iii) the *XML enabled Communication Framework* (XCF) [169] that allows to distribute components over several computing nodes by, besides others, a remote method invocation (RMI) pattern, and

(iv) a library named *XMLTIO* that supports developers with an XPath[29]-based API for simple XML processing.

By means of this infrastructure, so-called *memory processes* have access to the memory through the active memory interface as sketched in Fig. 5.3.

Following the concepts of this integration approach, almost all content in a visual active memory is stored and exchanged by means of XML documents. Hence, *memory elements* are generally XML documents. Furthermore, binary attachments can be linked to these documents to store, e.g., image patches, as required by the pictorial memory. Besides standard database-like methods like “insert”, “update”, “delete”, and “query”, the AMI server

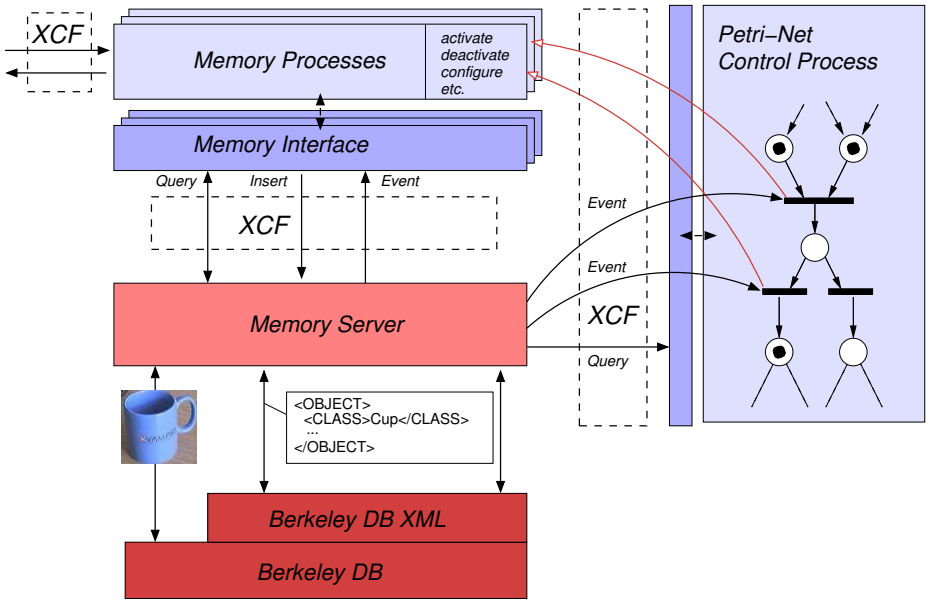


Figure 5.3: The active memory infrastructure according to Wrede [168]: Clients (Memory Processes) connect to the server component using the memory interface. A combination of the relational Berkeley DB and the native XML database DB XML is used to implement the repository for memory elements. XCF [169] is utilized to provide remote access and distributed processing capabilities. A global coordination component utilizing petri-nets is linked to the active memory server by means of so-called active memory guards.

provides also a flexible event-notification mechanism that enables information-oriented coordination. *Memory processes* can subscribe to events specifically for a selected set of memory elements through the active memory interface and get triggered, whenever such an event occurs. Such events belong to one of the following types: “insert”, “update”, and “delete”, corresponding to the respected methods mentioned above.

Example 5.4: *Reconsider the idea of action recognition outlined in section 3.4. The tracking of an object gets triggered, when an object is reliably detected in the view of the user. Instead of polling the memory for such an object memory element, the tracking component gets actively triggered by the integration broker, if a matching hypothesis is available.*

Access to the memory contents, and subscription to events is based on XPath. Using XPath, the problem of information selection and access can be tackled in a very flexible and declarative way. It allows memory processes to narrow down the information they are interested in. How this actually is done in an integrated system will be outlined in chapter 7. A small introduction on XML and XPath is provided in appendix A of this thesis. Furthermore, Wrede [168] discusses the advantages of XML and XPath from the perspective of software integration with a focus on cognitive systems in his work.

Additionally, Wrede [168] also proposes a generic framework that utilizes petri-nets [124] to represent system states and state changes for the global coordination of integrated systems. Depending on state changes, this component of the AMI allows to start, stop, and reconfigure other components implementing the AMI interface, or to execute any arbitrary commands to control the system behavior.

5.3 Representation of Memory Elements

After a memory as a central active component for cognitive vision systems has been motivated in general and the AMI has been presented as the basis for its implementation, I will now focus on the representation of content in this memory. Giving the flexibility of XML as basis for this representation and the structure of the VAM outlined before, the cornerstone has been laid so far. An implementation of these layers leads to the definition of basic types in the memory.

In section 5.1.3 different layers of a memory have been introduced and the stored entities have already been termed ***memory elements (ME)***. Now a more detailed definition and a proposal for the representation of such memory elements shall be given. Memory elements are the atomic fragments exchanged, stored and retrieved in the system. They are a basically represented as XML documents in accordance with the AMI concept. All different types of memory elements are stored and exchanged by means of the integration broker of the AMI.

It is a general achievement of the integration approach of Wrede [168] that no fixed and explicit definition of data types needs be implemented. Rather, memory elements can be easily extended, and modified at run-time. Although this is a great benefit in general, basic data structure need to be defined in order to let different processes operate on the memory content. No complex hierarchy of all different memory elements that are relevant for a complex system shall be defined, but still some basic types are required.

Generally, more complex memory elements are composed of simpler structures. A two-dimensional point is usually represented by two float values, to give a simple example. In order to facilitate exchange between different processes, the representation must be the same for all structures that have equal underlying semantics. In the following, I term data structures that are part of memory elements *attributes*. These term should not be confused with the XML attributes as described in appendix A.

All memory elements in the central VAM are not explicitly stored in a specific layer, although the type definition introduced in the following is picking up the conceptual structure outlined before. They are XML documents that can be distinguished by the actual information they carry. They are all stored and exchange in one memory to all links and references.

5.3.1 A Hierarchy of Basic Memory Elements

As a consequence of the above mentioned demand for some common structures, a hierarchy of basic memory element types is developed. It widely corresponds to the VAM concept introduced in Sec. 5.1.3 and shown in Fig. 5.4. Types that directly correspond to layers in the memory are highlighted in the diagram. The most general type is consequently termed "Memory Element", as every piece of content in the memory is a memory element. In the notion of the AMI concept, a memory element is an XML document. Each type in the hierarchy may require some mandatory attributes that are also shown in the figure in red color. Such mandatory attributes facilitate generic memory processes as while be discussed in chapter 6. Note that any specialization of these abstract classes can have further *optional* attributes assigned, as some are also shown in green in the hierarchy. As optional attributes can simply be added at any time facilitated by the underlying XML representation, any specialization of these generic types is thus supported by the AMI.

But besides attributes, some specific "treatment" rules and assumptions are associated with

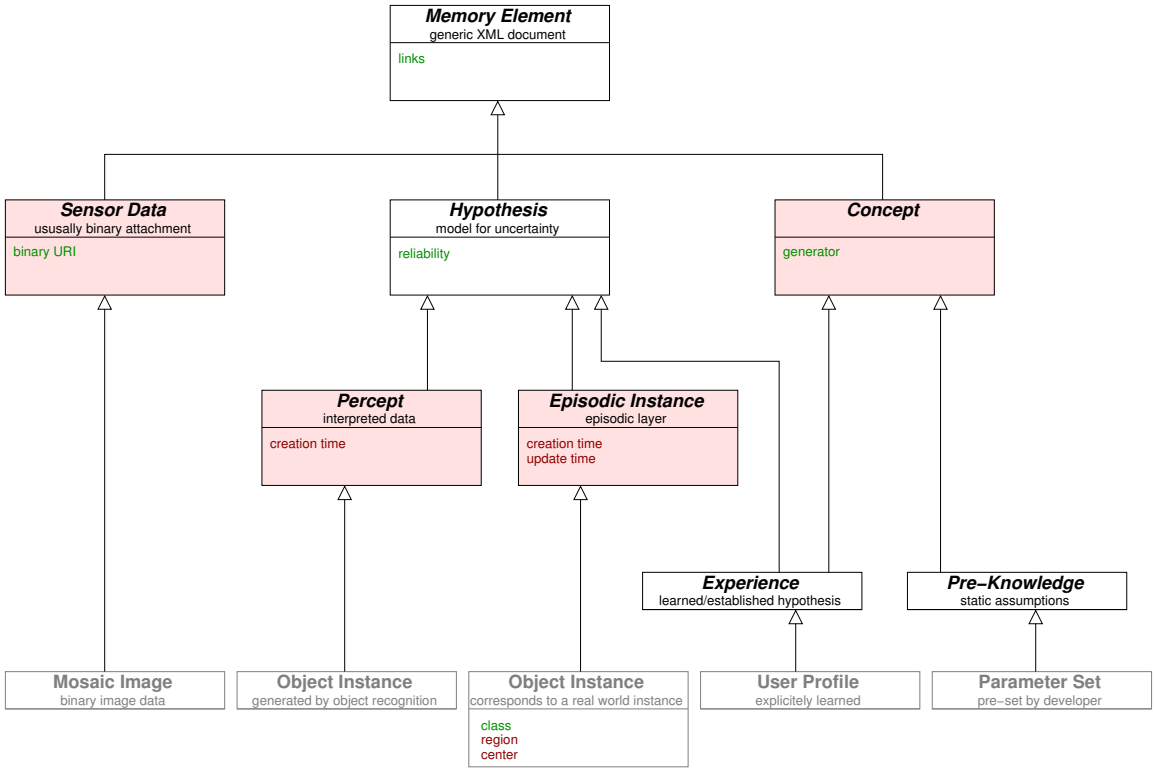


Figure 5.4: A basic class hierarchy of Memory Element (ME) types in UML-like notation. Some exemplary specializations are given (e.g. "User Profile") in grey. Mandatory attributes are displayed in red, some optional in green. Types with a light red background correspond to respective layers in Fig. 5.2.

the specific types accounting for the respective characteristics. These are not formally defined and shown in the figure, but reported in the following discussion of the basic types.

5.3.2 Basic Memory Element Types

It has been stated before, that there can often be *links* between different memory elements. For instance, the information about a recognized cup might be linked to the image of the cup itself, or to the general concept of a kitchen, which is modeled to contain cups. Therefore, each memory element can basically be linked to other memory elements in order to refer to a related one, as expressed by the memory element's optional attribute `links` in diagram 5.4.

On a first level of specialization *sensor data*, *hypothesis*, and *concept* are introduced. Let us now take a closer look at these three major specializations of memory elements. ***Sensor data*** has been introduced as having no meaning and being a less structured entity. It is a very generic type and can have many further specializations. In a vision system, sensor data usually comes in large chunks like images, videos, or mosaics to pick up the example of the pictorial memory described in Sec. 3.1. Therefore, sensor data most often has binary data attached, referenced by an `URI`[1] as proposed by the AMI.

Another fundamental type of memory elements is ***hypothesis***. It is introduced here to

[1] a Uniform Resource Identifier is used to uniquely identify binary attachments.

provide a most generic means to cope with uncertainty of perception and interpretation in vision systems. Generally, no information in a cognitively motivated memory should be considered as irrevocable facts, but rather as hypotheses with a certain `reliability`. As this a major foundation for most of the processes in the concept of the visual active memory, an own section (Sec. 5.3.4) is devoted to the subject of uncertainty and the hypothesis concept. Two specializations of hypothesis are introduced following the concept of the VAM.

Percept is introduced as a semantical memory element type in correspondence to perceptual layer of the VAM outlined before. Reconsider, that a percept carries a meaning and is usually situated, as it is the result of an interpretation process. Percept is usually a structured and at least partially symbolic type. In the VAM notion, it is associated with a specific time and is created as an interpretation provided by a dedicated process. Consequently, an additional mandatory attribute is added for percepts: the `creation time`.

In contrast to percepts, memory elements in the episodic memory, so called ***episodic instances***, have an extend in time and can be doubted or supported. They are not only generated once, but can be revised, updated and discarded. In order to account for the presumption that these memory elements are not facts, but the result of some reasoning or interpretation, they are also derived from *hypothesis*. Every hypothesis has a `creation time`, similar to percepts, but furthermore – to account for the property, that memory elements in the episodic memory can be revised and have an extend in time – `update time` is added to the list of these generic attributes. In accordance with the concepts of a episodic memory, episodic instances should have a direct correspondence to real world entities or events. An object, that is or has been apparent in the scene, for instance, should therefore directly correspondent to an episodic instance. How this can be achieved in the active memory approach will be presented in Sec. 6.4 for the example of a generic memory process.

Concept is also a structured type as the ones before, but has a much longer life time in the memory, and is not only valid in a specific scene or episode. But as has been discussed before, concepts can either by generated as a result of learning processes, or can be premodeled. The first type inherits also from *hypothesis*, since it can literally also be considered as very assured hypothesis. This particular specialization is termed ***experience***. As an example for experience, the acquired model of user preferences shall be given:
Example 5.5: *Sebastian is highly addicted to "Latte Macchiato". The assistance system might have learned from long-term observations that he always likes to have this drink when he is staring at his empty mug on the desk. Such knowledge is considered as experience in the notion of the VAM concept. In such a case the system will either inform Sebastian about the location of the closest coffee machine, or might remind him, that too much coffeine is harmful for his health.*
The other type of concepts are pre-defined ones that are given as facts and can not be doubted by the system. They form the so-called world knowledge or ***pre-knowledge*** of the system. Memory elements that contain pre-knowledge properties are considered to be static and indispensable. Spoken in the notion of cognition, this constitutes inborn knowledge that cannot be revised. In a computer system, configuration parameters, runtime settings, world assumptions, and others more are usually regarded as pre-knowledge.

Granlund [62] formulates the distinction between pre-knowledge and experience as a general problem in computer vision systems. On the one hand, knowledge is supplied externally by the developer, and on the other hand, knowledge is acquired by learning process. He states, that nowadays the optimal solution lies in-between these two extremes [61] and their "balance" is dependent on the specific scenario.

```
<OBJECT>
  <HYPOTHESIS>
    <GENERATOR>Object Recognizer BU(N)</GENERATOR>
    <TYPE>OR_RESULT</TYPE>
    <TIMESTAMPS>
      <CREATED value="3452345"/>
      <UPDATED value="3452398"/>
    </TIMESTAMPS>
    <RATING>
      <RELIABILITY value="0.6"/>
    </RATING>
  </HYPOTHESIS>
  <CLASS>Cup</CLASS>
  <REGION3D>
    <CENTER3D x="534.45" y="-285.33" z="545.21"/>
  </REGION3D>
  <REGION image="img_office210703_122">
    <CENTER x="367" y="285"/>
    <RECTANGLE x="335" y="245" w="65" h="80"/>
  </REGION>
</OBJECT>
```

Figure 5.5: Episodic memory element represented in XML. It incloses a highlighted element `HYPOTHESIS` containing the commonly shared attributes of all hypothesis memory elements.

5.3.3 An Exemplary Memory Element in XML

In Fig. 5.5 an example of an object hypothesis in the episodic memory is shown as it is represented in XML. This type has also been introduced as an example for a specialized episodic instance in Fig. 5.4. It is hierarchically structured to represent the different attributes this memory element is composed of. The highlighted element `HYPOTHESIS` contains alle shared attributes that are derived from the types “hypothesis” and “episodic instance”. One can find the attributes `CREATED`, `UPDATED`, and `RELIABILITY` here.

Attributes like `REGION` are themselves composed of other attributes, e.g., `CENTER`. These type definitions are common for all components that cope with regions. A region is not only contained in a result of an object recognition component. It could, for instance, also be the result of the visual tracking that has been discussed in Sec. 3.4. As both, recognition and tracking, encode for the same semantic, namely referring to a region in an image, both are also syntactically the same in the representation.

In the given example in Fig. 5.5, a class of the object has been determined. This does not necessarily need to be the case, as illustrated by the following example:
Example 5.6: *The object segmentation being part of the so-called VPL approach presented in Sec. 3.2.2 can detect objects without necessarily determining their specific class. But if the class can be ascertained later on an additional attribute* `CLASS` *can be added to the memory element.*
This does not effect the rest of the memory element representation; the additional information is just *added* to the existing memory element.

5.3.4 A Model for Uncertainty: Reliability of Hypotheses

One property of the *hypothesis* type shown in Fig. 5.4 has only been very briefly mentioned yet: its model for uncertainty. It has been stated so far, that hypotheses are no irrevocable facts in the memory and that they can be doubted or supported, but how this is represented is still an open question.

Up to now, memory elements have been considered as atomic entities that are either stored in the memory or are absent, or in other words, are either true or false. But this assumption does not hold for cognitive systems in general, since modules in such system usually do not provide complete accurate results. Thus, in a cognitive system results should never be treated as irrevocable facts, but really as *hypotheses* in the literal meaning of the word. In order to cope with information that implies a significant level of uncertainty, I propose to include a generic measurement to represent this reliability of a memory element. Therefore, in the XML representation, memory elements of type "hypothesis" contain a dedicated `HYPOTHESIS`-element which comprises a ***reliability value*** as shown in Fig. 5.5. To allow some general comparability this value is defined to take any real value in the interval $[\mathbf{0.0, 1.0}]$ that correspond to the reliability of the information contained in the memory element.

The worth of such a value should be obvious: If some proper measurement for the reliability of memory elements is available, the cognitive system does not have to finally decide about the truth of any perception on a very low level of interpretation. By means of this, the system can also propagate different hypotheses in parallel and use hybrid approaches to reason about these memory elements. Furthermore, introducing a measurement for uncertainty on a rather generic level also allows the realization of like-wise generic memory processes. As the reliability value is available for any memory element of type "Hypothesis", processes are developed that only consider this information and can therefore handle any kind of specialized hypothesis. A most relevant example for such processes is the "*forgetting*"-process that is described in greater detail in section 6.6. But also many other memory processes introduced in chapter 6 consider the reliability value.

Up to this point, the question of how the reliability of generated information can be assessed has mostly been neglected for a good reason: It is a tough problem that cannot be directly solved by any general approach. There exist interpretation or classification algorithms that provide some kind of confidence measurement themselves. Classifiers that are based on a probabilistic approach compute a probability for each interpretation, that can also be interpreted as a measurement of reliability to some extend. For other approaches, some heuristics can often be applied to compute a reliability value. But the comparability of any kind of measurement, that result from different interpretation processes, is arguable. Kölzow [90], for instance, proposes a transformation of independent measurements into a *propability space*. Therefore, he performs a supervised statistical analysis of each involved interpretation process in order to be able to compute a likelihood from an estimated distribution. By means of his approach, a simple model for *information fusion* based on normalized likelihoods can be established. Such reliability-based cue weighting is often proposed for different sensor fusion tasks, as for instance applied by Rosas et al. [137] for depth estimation. Nevertheless, it is still an open research question, how algorithm-independent quality of service (QoS) and general confidence measures can be established.

Note, that the assessment of the reliability of memory elements in a cognitive system does not need to be an bottom-up process as discussed up to now. Most often the *context* plays a significant role in order to evaluate the reliability of any information. The reliability values

can also be modified by other processes that perform some global evaluation and consider the different memory elements in context to each other. This is in direct compliance with the flexible processing paths introduced as typical for cognitive systems in chapter 2. An example of such a memory process, that analyzes the context in the memory, is presented in 6.5.

In this description of the active memory concept, I do not want to go into detail about that individual assessment performed by some specific interpretation processes. But it should be kept in mind, that accounting for uncertainty on a generic representational level allows to implement rather generic fusion processes as they are also discussed in [170].

The reliability has been introduced as an *optional* attribute in the hypothesis type and the decision has been made that every interpretation of perceived data is generally considered as a hypothesis. If the interpreting process does not provide any useful confidence measurement, the reliability information can be omitted without effecting the overall functionality of the memory system. If reliability information is not available, we follow a pragmatic approach and treat the hypothesis as a fact, corresponding to a reliability value of **1.0**. Nevertheless, a reliability value can be added to the memory element at any later stage even if it is not initially available; for instance, by the contextual analysis processes mentioned above, or by any other process that considers the particular memory element as less reliable.

5.4 Summary and Contribution

Developing integrated cognitive vision systems demand a conceptual architecture that accounts for flexible processing paths, learning, adaption, and memories. Motivated by requirements and assumptions arising from three different perspectives, I presented a *visual active memory (VAM)* as a conceptual architecture for cognitive systems in general and egovision assistance systems in particular. The concept is built around a structured memory consisting of sensory, perceptual, episodic, and conceptual memory layers, which together constitute the visual active memory. Almost all kinds of data, information, and knowledge exchanged and stored in the system are mediated through this central memory, in order to closely relate memory representations with processing schemes. Although the term "VAM" is often used to denote the central memory instance that serve all the different layers of the memory, the VAM concept is more. It comprises a close coupling between representation and processing schemes, the generic types of memory elements, and basic memory processes that organize the memory content.

The *Active Memory Infrastructure (AMI)* is utilized as the framework for the implementation of this visual active memory concept. It comprises of a central integration broker that has been proposed by Wrede [168] and allows flexible processing schemes and provides powerful coordination capabilities. Wrede [168] also proposed to use XML documents as basic entities for exchange data in cognitive systems. This proposal has been adopted for the representation of memory elements developed for the VAM concept. This memory representation has been carefully designed following insights from cognitive vision research, artificial intelligent, and also pragmatic engineering issues. Based on the four-layered VAM architecture, a hierarchy of basic memory elements has been proposed, which are represented as XML documents in the memory. Such memory elements are comprised of as mandatory and optional *attributes* facilitated by this XML representation. Generic memory processes can be developed that consider only common attributes of different memory element types, while the extensibility of XML documents allows other components to add

specific information to memory elements whenever it is available. In order to account for uncertainty in a generic and flexible way, the *hypothesis* type has been introduced containing a *reliability* attribute that will be considered by the memory processes presented in the following chapter.

The first major contribution of the visual active memory as a conceptual architecture for cognitive systems is the flexible and unifying representation of the memory content on different abstraction levels that allows generic memory processes. The second contribution lies in the powerful coordination capabilities inherited from the integration framework AMI, that closely couples representation in the memory with process coordination. The active memory basically connects perceptual and interaction functionalities, that have been presented in chapters 3 and 4, to integrated, interactive cognitive ego-vision systems. Resulting processing paths and dedicated memory processes that implement this connection will be discussed in the next chapter.

6 Memory Processes

The whole is more than the sum of its parts.
[Aristotle, 384 BC – 322 BC]

The VAM concept is not only comprised of the central repository containing memory elements following outlined type definitions. The functionality of an integrated system using the visual active memory concept is the result of an interplay of different *Memory Processes (MP)*. As stated before, such MPs are connected to the central memory instance and are coordinated by means of an event-notification scheme based on the flow of *memory elements*.

To give a rather pragmatic and general definition, every component of the system is termed memory process, if it is connected to the memory, utilizes the memory interface of the AMI, and follows the representational guidelines of memory elements introduced in the previous chapter. Memory processes are in consequence only loosely coupled to each other by means of the active memory, as usually no direct connection between these processes are intended. The representation in the memory and the coordination of processes are closely related by means of the information-oriented integration approach of the AMI.

6.1 Principles of Memory Processes

Some general principles of memory processes can be identified from the general requirement of the active memory introduced in Sec. 5.1.4. These are picking up some of the treatment rules associated with the basic memory elements as introduced in Sec. 5.3.1.

Submitting Memory Elements Usually the outcome of any kind of computations, perception, or reasoning will be submitted as a memory element. When a memory process submits to the memory, it generally takes the following requirements and principles into account:

- ▷ The memory process must consider the definitions for common data types and the type hierarchy introduced in section 5.3. Especially *mandatory* attributes must be provided consistently.
- ▷ The process should submit as much information as possible. Following the concept of optional *attributes* in memory elements, there is almost no reason why a memory process should retain pieces of information it has gathered. Other memory processes might simply ignore the optionally provided attributes, or make benefit of it. The performance issue potentially caused by larger memory elements is negligible, due to the active coordination using XPath filtering, as long as the increase in amount of exchanged data is not to excessive [169].

- ▷ A memory process must account for the requirements outlined in section 5.1.4. In terms of *reliability* this implies that it should provide any measurement of self-assessment regarding its provided outcome. To account for *maintainability*, memory processes should have a preference for *extension* rather than *creation* of memory elements. As pictured in example 5.6 on page 86, a process should add a specific *attribute* containing the extension of the former content.

Receiving and Interpreting Memory Elements Some principles for the receiving memory process are direct consequences of the principles introduced for the submitting side. The following principles are identified:

- ▷ A memory process should be as specific as possible in the its subscriptions to the memory instance. In order to avoid unnecessary trigger invocation the memory processes subscribes to exactly the information it needs. It also allows generic processes to ignore any structures in the memory elements not explicitly needed for the functionality. By means of the XPath-based definitions of subscriptions, the information that triggers a memory processes can be selected very specifically. This avoids unnecessary traffic between components and a waste of computational capacities.

- ▷ A memory process should be as generic as possible in the treatment of memory elements. As a consequence, always the same syntactic structure should be applied for the same semantic attributes in the memory elements.
  Example 6.1: *If some memory process requires a two dimensional region specification with respect to an image, as for instance provided by the object detector shown in Fig. 5.5, it should be able to cope with any kind of memory element that contains such an attribute. A memory process that has the task to compute certain region features like moments, mean color, etc. can perform its computation without knowing whether the regions originate from an object-detection algorithm or is the contour of the user's hand.*

  A certain level of generic abilities is obviously ascertained by the type hierarchy of memory elements discussed before. Consequently, generic memory processes can be designed to abstract from specialization and refer to the base classes' attributes in the hierarchy.

- ▷ A memory process must not break down, if some information is not available, but should *scale* with the amount of content provided by a memory element. Nevertheless, memory processes can rely on mandatory attributes to be present in a memory element, but should account for optional attributes by also providing a better or more complete outcome. An example of this use case has been given before on page 86, where the class of an object detection has been declared optional.

Summarized, it can be said, that memory processes should be as *specific* as possible in their subscriptions, *generic* in their interpretation, and *scalable* with respect to the amount of information provided as input. This allows to develop rather generic memory processes to be instantiated multiple times with different subscriptions and parameterizations. We will come back to this principle in the description of some more general memory processes in the remainder of this chapter. With these principles in consideration, the path is paved to built rather flexible cognitive vision systems on the basis of memory processes that are coupled by means of a visual active memory. The outlined principles have been taken into account for the designed of all memory processes described in the following. But before

we go into detail about these different processes, a detailed view at the process flow and different general types of memory processes shall be taken.

6.2 Asynchronous Processing Paths

We have seen that memory processes are only loosely coupled by means of the active memory infrastructure, and that processing paths only constitute temporarily and dynamically. So why should process paths discussed any further? From *cognitive vision systems* and *vision as process* paradigms in general, and from *ego-vision* in particular, some general thoughts regarding processing paths have to be considered.

Taking a closer look at the envisioned kind of systems, it turns out, that processing will take place at different time scales. For ego-vision it has been stated, that the human is integrated in the loop of the system, but what happens if this loop is slow?
Example 6.2: *In example 5.1 Zoe captured some views of her keys in order to teach them to the system. This is designed as an interactive task, in which the user directly sees, which object is focused by the system, as she is integrated in a feedback loop. But the training itself, might take some time, varying between some seconds and even hours, depending on the underlying learning algorithm applied in the system (see Sec. 3.2). Zoe would hardly accept a lack of other functionality or even a black screen in the mean-time. Therefore, the learning has to take place in a decoupled manner, not affecting the interaction loop.*
This example motivates different *asynchronous* processing paths that have different constraints in terms of response and processing time. Figure 6.1 sketches the visual active memory with different processing paths attached. At the lowest level, *perceptual processes* and *interactive processes* are introduced. These realize the loop in the notion of ego-vision systems in which the user is integrated. Accordingly, they implement mostly *reactive* behaviors of the system. Various examples of such processes have been presented before. They implement scene perception as outlined in chapter 3 or visualization and user interaction (chapter 4). But other processes exist, as for example learning processes, that operate on a larger time-scale, and that can even continue, when the system is not actively used. So such processes can be thought of in cognitive terms as “deep thinking”.

In between, there exist other types of processes that either *fuse* or *(re-)organize* some content of the memory. The task of such memory processes is to interpret the memory content, analyze it in a certain way, generate new information from the existing content, or to establish links between memory elements as a result of reasoning. For some of these processes, response time is also critical, namely if their results are required for a reactive processing loop. But many of these do not have to fulfill such hard constraints with respect to process time and work asynchronously on the memory content. An example for a fusion memory process has already been presented, but not explicitly termed so yet. The approach to determine the three-dimensional position of objects presented in Sec. 3.3.2 fuses information from pose tracking and object recognition in order to generate new information that is added to object hypotheses.

By means of the outlined discrimination of processes, the active memory concept shares aspects with the cognitive vision architecture proposed by Granlund [61], where a perception-action loop is defined, and processes on a symbolic and reasoning level are attached to these in an asynchronous manner, too. He introduced the perception-action structure as dealing with the here and now, while other processes consolidate and generalize in space and time. Note, that in ego-vision systems almost all processing paths form loops, either by means of obtaining and submitting information from or to the memory, or by means of the integrated

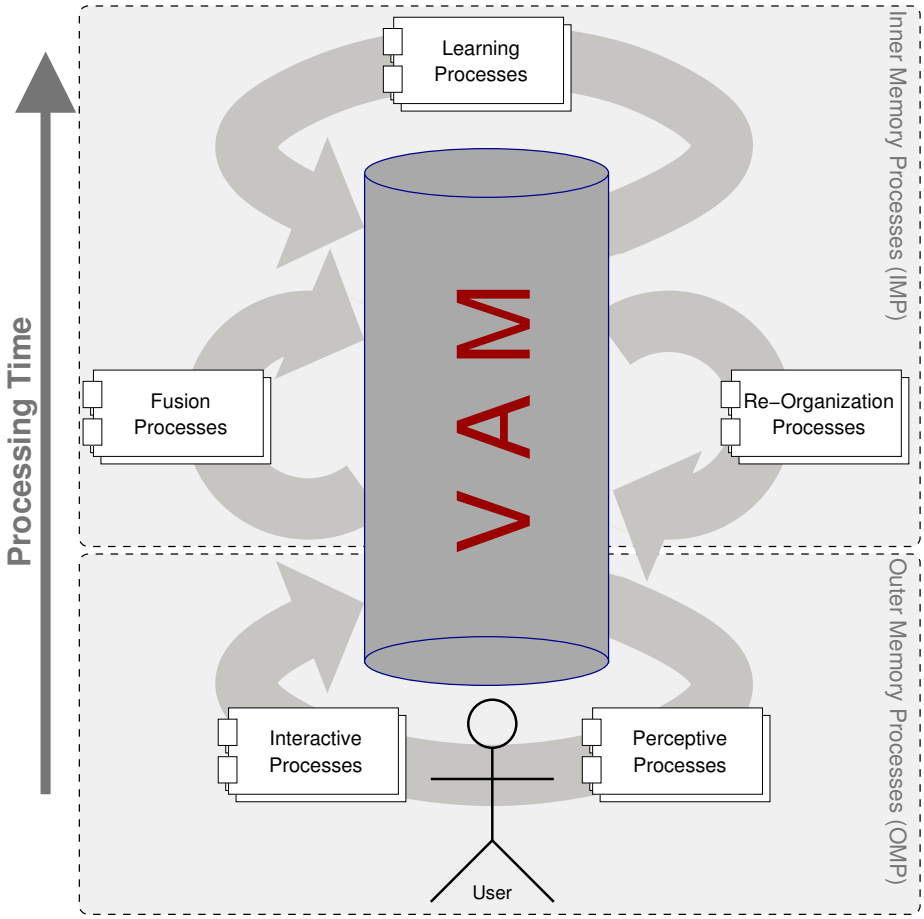


Figure 6.1: Asynchronous memory processes: Different processing loops exist in the active memory architecture and have different constraints in terms of processing and response times.

user.

6.3 Types of Memory Processes

Memory processes have been introduced to receive or query, submit, delete, and modify memory elements in the visual active memory in an asynchronous manner running at different time scales. But what triggers a memory process to submit or update content in the memory? The manipulation of the memory content conducted by some processes can have different causes. It might either be a result of some perception or interaction process. Or it can be a consequence of some kind of a reasoning or internal process that has been applied on the content of the memory. Furthermore, the outcome of a memory process does not necessarily need to be a manipulation of the memory content. It might also constitute in some form of articulation or manipulation that has an effect on the environment or the user of the system.

Following these argumentation, we take a closer look at different memory processes. ***Outer memory processes (OMP)*** and ***inner memory processes (IMP)*** are distinguished in the definition of the active memory concept. The differences between these two types

will be outlined in the following.

6.3.1 Outer Memory Processes

As shown in the schema sketched in Fig. 2.4 on page 15 already embodied vision systems need perceptual and articulatory capabilities, and by these means have a connection to their environment. In ego-vision systems this environment also incorporates the user. The necessary connection of the active memory architecture to this environment is provided by outer memory processes.

In Fig. 6.1, such processes are introduced as *perceptual* and *interactive* processes. In ego-vision systems, their processing loop is bound to a connection to the outer world and to the user. Picking up the arguments of the previous section, outer memory processes must thus be able to perform online and in (soft) real-time, as they directly affect the coupling between the user and the system.

Generally, OMPs can either be inbound or outbound from the memory's perspective. For the inbound case, OMPs realize *perceptual processes* that are connected to sensors to perceive the environment. Such OMPs can perform an interpretation of this data already to directly submit generated *percepts*, or they provide the perceived data to the sensory memory for later analysis by other memory processes. Examples of such inbound outer processes are all kinds of adaptors that encapsulate data perceived by sensors into memory elements. Furthermore, also components like the described object recognition, which perceives images from a camera and submits object percept memory elements, constitute suchlike outer memory processes.

Outbound OMPs utilize some kind of interaction device or a manipulator to articulate information to the outer world and the user, or to directly act in the environment, respectively. In ego-vision systems, processes that use a display to present information to the user are typical outbound processes. But also some kind of low-level behavior-based manipulator control is imaginable, when translating the concept of OMPs to the robotic domain, for example.

The explicit distinction between out- or inbound does not need to be existent for outer memory processes as there might be cases, where one process meets both definitions. Typically, interaction processes that realize the interface between the user and the system are OMPs comprising both communication strategies.

According to this definition, OMPs are substantial for the *embodiment* of cognitive systems in general. By means of outer memory processes, such systems become *situated* and *interactive*. Some OMPs have been discussed in this work already in terms of mosaicing, object- and action recognition, pose tracking, input modalities, and visualization in chapters 3 and 4, respectively. They all constitute outer memory processes in the notion of the visual active memory, and are connected to it accordingly as will be laid out in chapter 7.

6.3.2 Inner Memory Processes

The second type of memory processes are termed *inner* since they, in contrast to outer ones, have no connection to the outer world. IMPs are only connected to the active memory as illustrated in Fig. 6.1. They can be seen as an integral part of the active memory concept itself and hence really are *inner* processes in a most literal sense. Inner processes usually analyze the (partial) content of the memory and modify it. Their effect is only indirectly

apparent to the user, as they influence the behavior of OMPs.

IMPs are applied to implement reasoning strategies and deliberative tasks. They take some information as input and provide added value information as outcome. So they might either generate completely new memory elements, remove existing ones, or update memory elements in accordance with the *maintainability* requirement of the memory.

As outlined in the section before, IMPs have basically less strict regimentation in terms of processing and response times. Offline learning tasks can as well be modeled as inner processes, as internal reorganization, or reactive fusion of memory elements.
Example 6.3: *Object learning (see Sec. 3.2) is an example for offline learning realized as an inner memory process. It takes captured images from the memory to establish a model of the object's appearance that can itself be stored in the memory as soon it is available. From that moment on, an OMP object recognition can utilize this model to actually recognize the respective object.*
Other important IMPs will be discussed in the remainder of this chapter.

6.4 Hypotheses Fusion and Anchoring

With the definition of memory processes given, we will now turn to one of the generic inner memory processes of the VAM concept. As a consequence of the definition of the memory with its layers as proposed in Sec. 5.1.3 the question arises how *perceptual* and *episodic* memory can be linked as outlined in Fig. 5.2. Let us reconsider that the episodic memory is designed to contain information about the state of the world as it is or as it has been in the past. Therefore, the goal is to implement an *equivalence* between episodic memory elements and real world entities as far as possible. An object existing in the real world, for instance, should correspond to exactly one memory element in the episodic layer for a specific time slice. This equivalence must continuously be established and updated on the basis of the perceptions that are available from inbound OMPs and on the information already available in the episodic memory. As this is a rather general problem in any system coping with percepts in a temporal context, it is consequently tackled in a generic way in the VAM concept by designing an inner memory process termed *"hypotheses anchoring"*.

This term has been chosen to account for the affinity of this approach to the theoretic framework of *anchoring* first established by Coradeschi and Saffiotti [35] which strongly inspired my work to solve the problem of linking percepts to episodes. In various works [33, 34] Coradeschi and Saffiotti outlined their anchoring approach. In their notion, anchoring is the process of creating and maintaining the correspondence between *symbols* and *percepts* that refer to the same physical objects.

In contrast to their work, I generalize the anchoring idea to an approach that allows anchoring of perceptual entities that might be results of different inbound outer memory processes to episodic ones. Therefore, a much broader notion of anchoring has to be applied here. This generalization also shifts the approach towards concepts known from information or data *fusion*. As stated by Hall and Llinas [65], *"data fusion is the process of combining data or information to estimate or predict entity states"*. Bringing together ideas of data fusion and the anchoring concept of Coradeschi and Saffiotti [35] paves the way to use the active memory as a model for information fusion as we already published in [170]. Fusion can generally takes place on different levels as claimed by Sadeghi and Kittler [139], who identified data, feature, and decision as general levels for fusion. The VAM with its different layers arranged in a central memory instance provides an ideal basis for fusion processes on

a rather generic level. In the following, hypotheses fusion on the decision level is presented as an intrinsic part of the active memory concept to link – or, anchor – percepts to episodes. The envisioned memory process is therefore a fusion process in the notion introduced in Fig. 6.1.

6.4.1 Probabilistic Anchoring of Episodes

Inbound OMPs generally report their current perception of the environment to the memory. But reconsidering arguments given in Sec. 5.3.4 and 5.3.1, all perceptual information should generally be treated as *hypotheses* with some amount of skepticism, due to possible errors in classification and interpretation. Fortunately, there is usually not only "one shot" to perceive the environment correctly. Objects, for instance, are recognized at a rate of approximately 10Hz with the approaches outlined in Sec. 3.2. As it is very unlikely that the scene changes significantly at such high rate, several percepts can be fused in order to gain better and more robust results, and consequently a more accurate representation of episodes. As OMPs in general have no memory itself, they cannot access past events or results and in consequence cannot ascertain whether a percept results from the same entity in the scene as the one generated in the last iteration or frame.

Furthermore, there can also be several processes that can provide similar or coherent information. Objects, for instance again, can be recognized by different algorithms. Two different approaches have been introduced in Sec. 3.2. But all these percepts, resulting from several perception or different OMPs, might belong to one and the same entity in the environment and should hence also be mapped to one episodic instance in the memory. The idea of data fusion on the decision level here is to combine results from the different approaches and make benefit of it.

The mapping of percepts to episodes should however be a probabilistic one, to account for the uncertain character of perception in general. Perceptual memory processes have a probabilistic nature as well in their recognition capabilities as also in their response time. It cannot generally be taken for sure, that a perception is submitted at a fixed rate. Rather, perception is also probabilistic process with particular timing characteristics which should be accounted for by the envisioned fusion and anchoring approach.

Having motivated the general idea of fusion and anchoring, now a more formal and detailed explanation of the envisioned process is given. In the following, I will refer to the perceptual memory element simply as "percept", and to the one in the episodic memory as "hypothesis", as anchoring is mainly about doubting and supporting the hypothesis of the existence of specific episodic instances in the memory. Let us denote the set of all percepts as $\boldsymbol{\Pi}$ and the set of all episodic hypotheses as $\boldsymbol{X}$. The goal of the anchoring process is then to establish a correspondence, termed "anchor" $\boldsymbol{a(x, \pi, t)}$, between a percept $\boldsymbol{\pi} \in \boldsymbol{\Pi}$ and an episodic hypothesis $\boldsymbol{x} \in \boldsymbol{X}$ at time $\boldsymbol{t}$ in a most congruent way. Based on this anchor, $\boldsymbol{x}$ should be updated to reflect the new information available from the percept $\boldsymbol{\pi}$. Each available percept $\boldsymbol{\pi}$ is therefore assigned to exactly one hypothesis $\boldsymbol{x}$.

6.4.2 Matching

The first question, that has to be answered is to *which* existing hypothesis $\boldsymbol{x}$ a given percept $\boldsymbol{\pi}$ should be anchored. This question is basically seen as a problem of *matching*. First, it needs to be considered that not every type of percept can generally be anchored with any type of episodic memory element. So it is obviously impossible to fuse the perception of

an action directly with the hypothesis of an existing object, because the *domains* do not match. This would be like comparing apples with pears. A memory element, as well in the perceptual as also in the episodic memory, has domain-dependent *attributes* which are encoded in the respective memory elements as discussed in Sec. 5.3. For object hypotheses, these are usually attributes like class affiliation, position, and others more. Denoting the attributes of percepts and episodic hypotheses as $\boldsymbol{\Theta(\pi)} \subseteq \boldsymbol{P}$ and $\boldsymbol{\Theta(x)} \subseteq \boldsymbol{P}$ respectively, allows to model a pair-wise assignment of comparable attributes by

$$\begin{aligned} \boldsymbol{\Phi} : (\boldsymbol{\Pi} \times \boldsymbol{X}) &\rightarrow (\boldsymbol{P} \times \boldsymbol{P}) \\ \boldsymbol{\Phi(\pi, x)} &\subseteq \{(t_\pi, t_x) | (t_\pi \in \boldsymbol{\Theta(\pi)}, t_x \in \boldsymbol{\Theta(x)}\} \end{aligned} \tag{6.1}$$

with $\boldsymbol{P}$ being the set of all possible attributes. If $\boldsymbol{\Phi(\pi, x)} = \emptyset$, two memory elements basically can not be compared and thus never match. Otherwise, a *match measurement* $\boldsymbol{\delta}$ can be defined

$$\boldsymbol{\delta} : (\boldsymbol{\Pi} \times \boldsymbol{X} \times (\boldsymbol{P} \times \boldsymbol{P})) \rightarrow [\mathbf{0}, \mathbf{1}] \tag{6.2}$$

as a function comparing a percept and an existing memory element in the episodic memory on the basis of the pairwise assignments $\boldsymbol{\Phi}$. If $\boldsymbol{\delta(\pi, x, \Phi(\pi, x)) = 1}$ both memory elements are considered as a perfect match, $\boldsymbol{\delta(\pi, x, \Phi(\pi, x)) = 0}$ if they do not match at all.

The comparison of a perceptual and a episodic hypothesis involves the comparison of individual common attributes using the assignment function $\boldsymbol{\Phi}$. The overall value of $\boldsymbol{\delta}$ is composed of these different individual measurements. In our approach, all individual measurements are considered to be statistically independent of each other. Accordingly, the overall measurement is obtained by multiplying the individual ones

$$\boldsymbol{\delta(\pi, x, \Phi)} = \prod_{\boldsymbol{\phi \in \Phi(\pi, x)}} \boldsymbol{\delta_\phi(\pi, x)} \; . \tag{6.3}$$

If an optional attribute, that is defined by a given $\boldsymbol{\phi} \in \boldsymbol{\Phi}$, but is not available in either $\boldsymbol{\Theta(\pi)} \subseteq \boldsymbol{P}$ or $\boldsymbol{\Theta(x)} \subseteq \boldsymbol{P}$, the matching function shall return a default value $\boldsymbol{\delta_\phi^{default}}$. This allows especially to account for the idea of extending existing memory elements with information whenever it becomes available. A newly created memory element might only contain very few attributes, as additional might be added by some other memory process (cf. Ex. 5.6) later on. In practice, the default value $\boldsymbol{\delta_\phi^{default}}$ is heuristically established individually for each attribute assignment. The definition of $\boldsymbol{\delta}$ and $\boldsymbol{\Phi}$ is very flexible and allows also to compare different types of memory elements as long as some valid comparable attributes are available.

The match measurement $\boldsymbol{\delta}$ provides the general criteria on which the anchoring process can decide (i) which existing hypothesis matches a new percept best, and (ii) whether the percept matches one at all. The latter is usually decided on the basis of a heuristically determined threshold $\boldsymbol{\epsilon_m}$ applied on $\boldsymbol{\delta}$. The concrete implementation of the different attribute-dependent functions $\boldsymbol{\delta_\phi}$ is subject to application specific constraints, which will be discussed in Sec. 6.4.4 for the example of object anchoring.

6.4.3 Process Flow

As stated before, the matching function $\boldsymbol{\delta}$ is discriminating the process flow of the anchoring which is illustrated in Fig. 6.2. Hypotheses fusion and anchoring is implemented as an inner memory process in the VAM concept. The process subscribes for percepts to get triggered by the memory instance whenever a particular perceptual memory element is

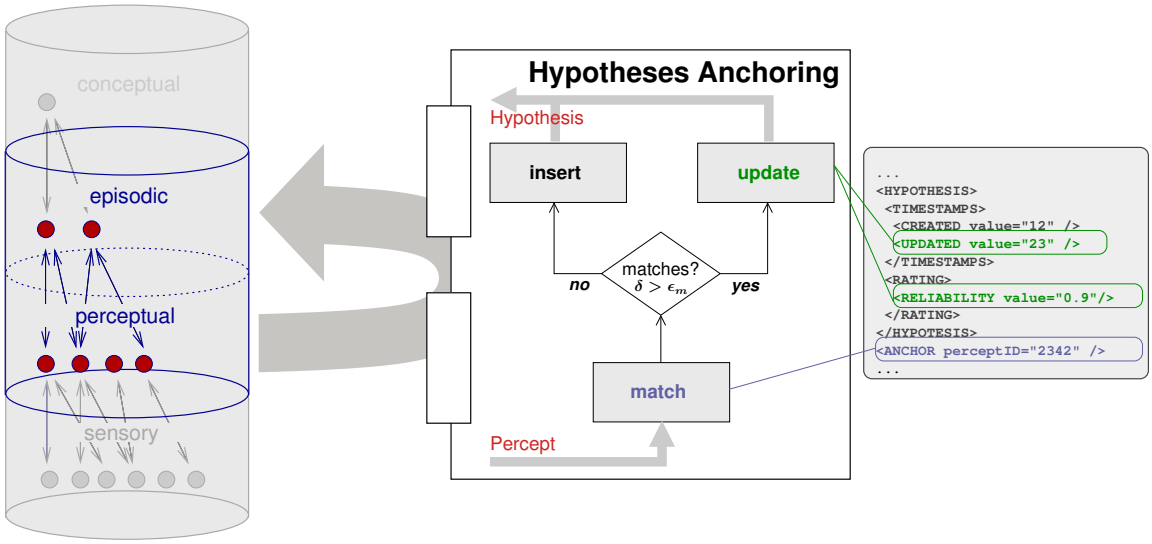


Figure 6.2: The inner memory process "hypothesis anchoring": New percepts in the memory trigger the memory process which can cause modifications in the episodic memory.

submitted to the memory. In that case, the new percept $\boldsymbol{\pi_n} \in \boldsymbol{\Pi}$ is compared to all existing hypotheses $\boldsymbol{x} \in \boldsymbol{X}$ using the match measurement $\boldsymbol{\delta}$ as outlined before. Two generally different processing paths can be the consequence of this matching procedure. On the one hand, the process may find an existing hypothesis $\boldsymbol{x_{best}}$ that matches the new percept best. On the other hand, it may not find a satisfactory match and is thus not able to anchor the percept to any existing hypothesis $\boldsymbol{x} \in \boldsymbol{X_D}$. Depending on this decision, a new hypothesis is inserted into the episodic memory or the matching hypothesis is updated, respectively. The anchoring process also adds an additional attribute to the memory element containing anchor meta-data regarding the established anchor $\boldsymbol{a(x, \pi, t)}$ that is needed to maintain a permanent link between the percept and the episodic instance.

The bottom-up scheme being the consequence of the subscription on percepts constitutes another major difference to the original anchoring approach, because Coradeschi and Saffiotti [35] follow mostly a top-down approach in their anchoring. Note however, that other memory processes are still allowed to generate hypotheses in the episodic memory directly, for instance as a result of a higher level reasoning process. Such hypotheses can simply be submitted and are automatically similarly anchored with percepts afterwards when any matching percepts arrive.

Although `insert` and `update` are in their detail implementation depending on the specific domains of the memory elements, the anchoring process generally updates or generates a respective hypothesis attribute as highlighted in Fig. 6.2. The value of the `UPDATED` information is set to the creation time of the assigned percept. The reliability is usually increased by a certain amount, since the percept supports the hypothesis. And finally, the `CREATED` timestamp is copied from the percept if a new episodic ME has to be created. The complete implementation of the anchoring process is summarized in algorithm 1.

The concrete definitions of $\boldsymbol{\delta_\phi}$, $\boldsymbol{\Phi}$, and the implementations of the `update` and `insert` functions are all depending on the domain of the hypotheses and the involved perceptual processes.

Algorithm 1 Hypothesis Anchoring

Input: Percept $\boldsymbol{\pi}$ from perceptual memory
Input: All memory elements $\boldsymbol{x} \in \boldsymbol{X}$ in episodic memory
Require: Threshold $\boldsymbol{\epsilon_m} \in [\mathbf{0}, \mathbf{1}]$ given
Require: $\boldsymbol{\delta_{best}} \leftarrow \mathbf{0}$ `// initialize best match value to zero`
Require: $\boldsymbol{x_{best}} \leftarrow \emptyset$ `// no best matching hypothesis is initially given`
for all $\boldsymbol{x}$ with $\boldsymbol{x} \in \boldsymbol{X}$ **do**
  $\boldsymbol{\delta} \leftarrow \mathbf{1}$
  for all $(\boldsymbol{t_\pi}, \boldsymbol{t_x}) = \phi \in \Phi(\boldsymbol{\pi}, \boldsymbol{x})$ **do**
    if $\boldsymbol{t_\pi}$ is an attribute $\boldsymbol{\pi}$ and $\boldsymbol{t_x}$ is an attribute in $\boldsymbol{x}$ **then**
      $\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} \cdot \boldsymbol{\delta_\phi}(\boldsymbol{\pi}, \boldsymbol{x})$ `// consider individual matching functions`
    else
      $\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} \cdot \boldsymbol{\delta_\phi^{default}}$ `// default if attribute is missing`
    end if
  end for
  if $\boldsymbol{\delta} > \boldsymbol{\epsilon_m}$ **then**
    $\boldsymbol{x_{best}} \leftarrow \boldsymbol{x}$ `// remember the best match`
    $\boldsymbol{\delta_{best}} \leftarrow \boldsymbol{\delta}$ `// remember the highest match value`
  end if
end for
if $\boldsymbol{\delta_{best}} > \mathbf{0}$ **then**
  $\boldsymbol{x_{best}} \leftarrow$ update$(\boldsymbol{x_{best}}, \boldsymbol{\pi})$
  update $\boldsymbol{x_{best}}$ in the episodic memory
else
  $\boldsymbol{x_{new}} \leftarrow$ insert$(\boldsymbol{\pi})$
  submit $\boldsymbol{x_{new}}$ into the episodic memory
end if

Timing Characteristics and Reliability Anchoring is seen as a process of continuous support or doubt of hypotheses in the episodic memory. To express uncertainty of a hypothesis a reliability factor has been introduced in Sec 5.3.4 as well as specific timestamp attributes for perceptual and episodic memory elements. These allow to consider temporal context and also the reliability of involved memory elements in a generic way.

Let us take a look at the role of the timestamps for the fusion and matching process. A hypothesis that is supported by perceptions continuously is expected to gain this support in a defined time interval. Of course, the length of this interval is dependent on the underlying perceptual memory process and also expected to vary more or less depending on several different factors like processing power, computational load, involved algorithms and others more. Hence, in order to answer the central question in the fusion approach, namely whether a percept matches an existing episodic instance or constitutes a new one, also has to consider the timing characteristics of the perceptual process. Simply, past episodes in the episodic memory are represented by memory elements that gained no support for a longer period of time. It depends on the task, whether past episodes should be kept in the memory for later retrieval, or shall be forgotten by the active forgetting process that will be discussed in Sec. 6.6.

In order to model the timing characteristics appropriately, an attribute-specific function $\boldsymbol{\delta_{time}}$ is introduced in the generic anchoring process. It can be applied for any perceptual input that is generated by a process that continuously submits new percepts as long as the particular cause in the scenario is apparent. The expectations regarding the charac-

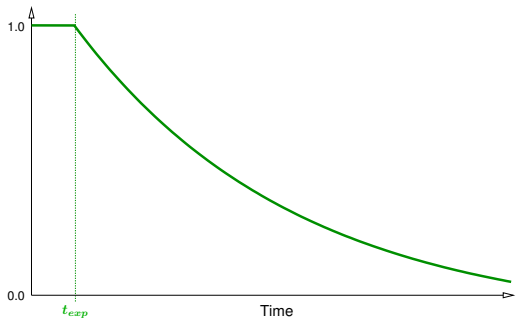


Figure 6.3: Adjusted exponential distribution $\boldsymbol{p_{exp}(t)}$ as a model for the temporal characteristics of the perceptual memory process.

teristics can be modeled using an interval $\boldsymbol{t_{exp}}$ of an expected refresh rate. The statistics of this interval between two updates, or supporting percepts, can be modeled as a shifted and rescaled version of the exponential distribution defined by

$$\boldsymbol{p_{exp}(t)} \quad = \quad \begin{cases} \boldsymbol{e^{1-\left(\frac{t}{t_{exp}}\right)}} & \text{if } \boldsymbol{t > t_{exp}}, \\ \boldsymbol{1} & \text{otherwise} \end{cases} \tag{6.4}$$

as plotted in Fig. 6.3. If a hypothesis gains support in less then the time given by $\boldsymbol{t_{exp}}$, the hypothesis gains full support and thus the value is $\boldsymbol{p_{exp}(t) = 1}$. Otherwise, the support is ebbing as longer time passes since the last associated percept.

This distribution is as well applied to compute a respective $\boldsymbol{\delta_{time}}$ for the matching process, as also to update the reliability value accordingly. The partial match measurement concerned with the time attributes is defined as

$$\boldsymbol{\delta_{time}} \quad = \quad \boldsymbol{p_{exp}(t_x^{updated} - t_\pi^{created})} \tag{6.5}$$

with $\boldsymbol{t_x^{updated}}$ denoting the last update time attribute of the episodic memory element and $\boldsymbol{t_\pi^{created}}$ denoting the creation time of the percept in the matching process[1]. By these means, it is very unlikely that a hypothesis in the episodic layer gains support by a new percept, if it has not been updated for a longer period of time. How long this period actually is allowed to be, is defined by the value of $\boldsymbol{t_{exp}}$, which can easily be estimated by measuring the submission rate of the respective perceptual process under controlled conditions.

The new reliability $\boldsymbol{\hat{r}_x}$ of an episodic memory element is updated using the same distribution incorporating the reliability of the assigned percept $\boldsymbol{r_\pi}$ and the previous reliability of the hypothesis $\boldsymbol{r_x}$

$$\boldsymbol{\hat{r}_x} \quad = \quad \boldsymbol{(1-\omega)r_x + \omega r_\pi \delta_{time}} \tag{6.6}$$

with a decay term $\boldsymbol{\omega}$ taking into account the history of the memory element. In practice, good results have been reported for $\boldsymbol{\omega}$ taking values between $\mathbf{0.3}$ and $\mathbf{0.5}$.

6.4.4 Anchoring of Objects

With the abstract functions $\boldsymbol{\delta_\phi}$, `update` and `insert` a flexible means has been introduced to serve as the basis for the anchoring of different entities. Now, we reconsider the original goal

[1] the corresponding XPath statements are `*/TIMESTAMPS/UPDATED` and `*/TIMESTAMPS/CREATED`, respectively

Attribute	Information Attr. (XPath)	update scheme *update*	matching model $\boldsymbol{\delta_\phi}$
2D pos. ($\vec{\boldsymbol{m}}$)	`*/REGION`	immediate copy: $\hat{\boldsymbol{m}} = \vec{\boldsymbol{m}}$	normal distribution: $\boldsymbol{\delta_{2D}}(\vec{\boldsymbol{m}}) \sim \boldsymbol{\mathcal{N}}\left(\hat{\boldsymbol{m}}, \boldsymbol{\sigma^2_{2D}}\right)$
3D pos. ($\vec{\boldsymbol{x}}$)	`*/REGION3D`	mean of last $\boldsymbol{n}$ percepts: $\hat{\boldsymbol{x}} = \frac{1}{n}\sum_{t=T-n+1}^{T} \vec{\boldsymbol{x}}$	normal distribution: $\boldsymbol{\delta_{3D}}(\vec{\boldsymbol{x}}) \sim \boldsymbol{\mathcal{N}}\left(\hat{\boldsymbol{x}}, \boldsymbol{\sigma^2_{3D}}\right)$
Class ($\boldsymbol{C}$)	`*/CLASS`	majority voting in histogram: $\hat{\boldsymbol{C}} = \mathbf{arg\,max}_{\boldsymbol{c}}\,\mathrm{hist}(\boldsymbol{c})$	confusion matrix: $\boldsymbol{\delta_{cl}}(\boldsymbol{C}) \sim \boldsymbol{M_{conf}}(\boldsymbol{C}, \hat{\boldsymbol{C}})$

Table 6.1: Applied statistical models and attribute-specific XPath statements for matching and updating in the object anchoring. The values for attributes of the episodic memory elements are denoted with a hat ($\hat{\cdot}$) and without for the percepts, respectively.

of anchoring as proposed by Coradeschi and Saffiotti [34] to some extend; the anchoring of real world objects. In the notion of the hypotheses fusion and anchoring this constitutes only a special, but nevertheless mostly relevant case and should also serve here as an example to clarify the general concepts.

In order to provide a profound probabilistic basis for the object anchoring approach we consider the anchoring as a problem of measuring the real world. Generally spoken, percepts contain measurements of attributes regarding the perceived object. The values assigned to the attributes thus constitute random variables and the process of perceiving the object is thus a random process. Accordingly, for most attributes, a statistical approach is applied to compute the match value between the comparable attributes of the percepts and the episodic memory element, and to determine the update values for the particular hypothesis.

When specializing the hypothesis anchoring approach towards specific types, the abstract functions $\boldsymbol{\delta_\phi}$, `update`, and `insert` have to be implemented. As comparable attributes $\boldsymbol{\Phi}$ considered by the object anchoring process, the (two-dimensional) position of the object in the image, its (three-dimensional) position in the scene, and its class affiliation (e.g,. cup, bike, etc.) are chosen.

For each attribute a different statistical model is applied to account for the specific characteristics in measuring their values. Table 6.1 displays these different models implemented as the basis to design the match functions $\boldsymbol{\delta_\phi}$ and the `update` functions, respectively. The implementation of the `insert` function is simple: Whenever a new hypothesis is generated from a perception, the attributes are copied according to a one-to-one mapping to generate an initial episodic memory element from the percept.

Next, the realization of the different attribute-wise match functions $\boldsymbol{\delta_\phi}$ and the respective `update` schemes is provided.

Spatial Attributes The 2D position of an object in the current image is always available as attribute, when a percept has been generated by any object detection or tracking process. However, the 2D position is very unstable in an ego-vision system and only valid for a very short moment in time. Even if the object itself is steady, the user might move the head arbitrarily and hence change the field of view of the system likewise quickly. This view-based coordinate system therefore only allows anchoring if the camera is more or less steady as well, and thus only is possible on the basis of short fusion time windows.

Taking a more scene-based perspective, in which coordinates of objects are given with respect to a global coordinate system, allows to really anchor steady objects in the scene based on their spatial position. With the approach presented in Sec. 3.3.2, global 3D coordinates can be established for objects even from the ego-vision perspective. However, this 3D coordinates might not always be available, since determining them is a much more complex process and more prone to errors than it is for the 2D image coordinates.

The object anchoring approach is designed to cope with both, 2D and 3D, spatial attributes and accounts for their specific characteristics only by selecting appropriate implementations of the abstract functions as shown in table 6.1. For the matching functions both take a normal distribution as the underlying statistical model to compute the match value. The variances $\boldsymbol{\sigma^2_{2D}}$ and $\boldsymbol{\sigma^2_{3D}}$ in this processes are different, depending on the estimated accuracy of the underlying perceptual process. The exact values are obtained in the integrated system and can be easily adapted if necessary. Note, that more sophisticated models, like a Kalman-Filter [80], which accounts for moving objects explicitly, can be plugged in here.

The *update* function, however, is different for 2D and 3D attributes. As stated before, the 2D image position is only valid for a very short time. Therefore, it is directly copied into the episodic hypothesis by the `update` function. For the 3D case, positions are expected to be much more stable and are only about to be changed due to manipulation of the object. In order to compensate measurement error, several percepts are accumulated to compute a fused result. However, as objects might move in the scene, the number of past percepts is restricted, resulting in a sliding mean computation incorporating the last $\boldsymbol{n}$ percepts to compute the 3D coordinate $\hat{\boldsymbol{x}}$ of the hypothesis. Again, $\boldsymbol{n}$ is dependent of the submission rate of the perceptual process. Good results are reported for $\boldsymbol{n \approx 10}$ for the algorithm described in Sec. 3.3.2.

Class Affiliation In contrast to the continuous, non-symbolic position attributes, the assigned class label is symbolic, but nevertheless also considered in the anchoring model of objects, if it is available. "Class" is an optional attribute, as explained in Ex. 5.6, and can in consequence be unavailable in a given percept. This might be the case if the perceptual process has detected an object without being able to classify it. Or, alternatively, the perception results from object tracking, so that no class label is determined.

Again, the recognition process that assigns the label is modeled as a random process, that can have some errors. Hence, the match function consults a confusion table $\boldsymbol{M_{conf}}$ of known objects to compute its result. The confusion table can be acquired by training the anchoring process in a supervised manner after the object recognition has been trained. Practically, a hand-crafted confusion table modelling a small probability for general misclassification exposed to work well, and avoids the extra effort of supervised training. In the simplest case, if the recognition process is considered as absolutely correct, this ends up in comparing $\boldsymbol{C = \hat{C}}$ and returns $\boldsymbol{p_{true} \approx 1}$ if they are equal, and $\boldsymbol{p_{false} \approx 0}$ otherwise.

The update function for such symbolic labels is a bit more complicated. As the nature of an object does not change from one second to another, a majority voting based on a fixed number of past assignments is applied to compensate for outliers that are caused by false recognition. Therefore a dynamic histogram of the last $\boldsymbol{m}$ assigned class labels is computed. The bin with the highest number of assignments is taken as result label.

6.4.5 Anchoring as Inner Memory Process

The anchoring process can, as any memory process, subscribe for different perceptual memory elements by means of specific XPath statements (e.g., `/OBJECT`). In table 6.1, the XPaths for the considered attributes for object percepts are shown in the second column. Memory elements containing at least some of these attributes can be treated as percept to the anchoring process.

The implemented anchoring can be subscribed for different XPath statements. By this means, it gets triggered whenever a memory element matching these XPaths is submitted to the perceptual layer. This flexibility allows not only to transparently integrate the two outlined object recognition processes presented in Sec. 3.2, but also to integrate the implemented object tracking algorithms presented in Sec. 3.4 in the context of action recognition. Both contain many of the considered attributes, whereas the object class can of course only be determined by the recognition approaches. By this means it is however possible, that an object hypothesis in the episodic memory still gains support and is updated while it is being manipulated, although the object cannot be ascertained reliably by the object recognition components. The object anchoring can transparently integrate these percepts, too, stressing the idea of flexible information fusion.

It should be further noted that the anchoring process scales with the number of attributes provided. If only the 2D position of an object is perceived, anchoring can work, but will produce less accurate results compared to anchoring on the basis of all possible attributes. But by means of the updated reliability values, the amount of uncertainty in the results can be directly represented in the memory elements and is in consequence also available to other processes, too.

6.5 Contextual Analysis for Consistency Validation

Context has been discussed as being relevant for the interpretation of perception already several times. Examples are the view contexts introduced in Sec. 3.3.3 and the planar scene decomposition (Sec. 3.1). With the anchoring introduced in the previous chapter, new percepts have been considered in context to existing episodic instances. But currently, all episodic memory elements are treated separately. There are no relations between them, although its is unlikely that they are independent of each other in reality. The content of the memory is generally not a collection of independent information fragments. On the contrary, relations and dependencies between these fragments are fundamental and form the *context* of the information stored in the memory.

In Sec. 5.1.4, *reliability* has been identified as a requirement for the memory, and it has been stated that the memory should not contain nonsense. On the other hand, it has been pointed out several times, that the perception is basically always uncertain and prone to errors. Picking up these statements again, it unveils that context can generally provide a means to identify errors and *inconsistencies* in the memory content. An example illustrates this assumptions for an episode of recognized objects:

Example 6.4: *The memory contains episodic hypotheses about a mug, a fork, a plate, an apple, and a razor pretended to be placed on a table. Although it is generally possible, this constellation is a bit unlikely. Actually, this memory content is a result of misclassification by the object recognition component that confused a knife with a razor, and assigned the wrong label to the object. The object recognition itself cannot notice its fault, but contextual analysis allows to detect it and ideally correct the mistake.*

This example indeed motivates to develop a generic memory process that allows to recognize inconsistencies and to identify memory elements that are probably wrong.

6.5.1 Contextual Models for Episodes

Although contextual analysis is generally valuable in various kinds, I here focus on the episodic layer of the memory. In the above example 6.4, *expectations* have been made that apply acquired knowledge about the world. In that given example, the knowledge can be formulated as follows: "A razor is more unlikely to appear in a spatial context of a fork, a mug, a plate, and an apple, as a knife." How can this knowledge be formalized in order to be applicable in the active memory?

Contextual analysis has been considered for quite a while in research to stabilize or facilitate recognition processes. Murphy et al. [110] evaluate holistic contextual features and local object detection jointly to improve as well the contextual determination as also object detection itself. Moore et al. [108] analyzes the context of objects and actions and can evaluate evidences in both direction to facilitate action and object recognition likewise. Common to both approaches is, that some model that relates the different perceptions to each other is applied.

In the context of the VAM, I define such a model of relations between elements in the episodic memory as *functional dependency concepts (FDC)*. An FDC relates episodic hypotheses to each other and allows to consider their context in order to evaluate the overall consistency of the memory content regarding the underlying model. In order to capture the characteristics of the relations between the different memory elements on the basis of statistics, *Bayesian networks (BN)* are utilized as underlying model. BN are directed acyclic graphs of random variables annotated with conditional probability distributions. The complete theory of these graphical models is explained in [78]. They are well known in research as model for conditional dependencies in various tasks, including context reasoning in particular. Bayesian networks proved to be qualified for multi-modal scene understanding [161], in the field of multi-cue object recognition [125] and in fusion of different domains, such as speech and vision [129]. Kittler et al. [86] also propose to apply Bayesian networks in a hierarchical memory architecture but focus on reasoning to interpret the content.

In the following I describe how Bayesian networks can be applied to implement consistency validation.

In order to realize FDCs on the basis of Bayesian networks, I basically consider the existence of specific memory elements as nodes in a BN. Hence, the existence of a specific memory element is considered as an assignment to the variables of the network. Therefore, all variables take values from the binary set $\{\boldsymbol{true}, \boldsymbol{false}\}$. In the context of Bayesian networks, these assigned values are termed *evidences* $\boldsymbol{e} = \{\boldsymbol{e_1}, \boldsymbol{e_2}, \ldots \boldsymbol{e_m}\}$. It should be noticed here, that Bayesian networks can also have so-called unobserved nodes, that are not get values assigned in this procedure, but that are needed to model specific dependencies between memory elements. In the example explained in Sec. 6.5.2, this issue will be picked up again.

Detecting Conflicts

After evidences have been assigned to the network, the posterior distribution of the variables is computed by probabilistic inference using the junction tree algorithm [111]. Then, a

conflict value *conf* can be calculated as a kind of *emergence measure* defined in [79]:

$$conf(\boldsymbol{e}) \quad = \quad \boldsymbol{log_2} \frac{\prod_{i=1}^{m} \boldsymbol{P(e_i)}}{\boldsymbol{P(e)}} \; . \tag{6.7}$$

The overall joint probability $\boldsymbol{P(e)}$ can be computed from the propagation of the network. This joint probability of the network is expected to be less than the product of the individual, independent probabilities $\boldsymbol{P(e_i)}$ of each finding in case of conflicts. In this case of conflicts, the evidences are not explained by given the model in this case and the value will in consequence be $conf(\boldsymbol{e}) > \boldsymbol{0}$. The probability, that this evidences occur independently from each other is higher in this case, than the probability, that they are dependent as defined by the model.

Uncertainty and Reliability

In order to cope with the uncertainty of episodic hypotheses, *soft evidences* instead of hard evidences are used for the observable nodes of the Bayesian network. Instead of enforcing values exclusive from the set $\boldsymbol{\{true, false\}}$, we allow each variable to have an evidence-vector $\vec{\boldsymbol{e}} = (\boldsymbol{e_{true}}, \boldsymbol{e_{false}})^{\boldsymbol{T}}$ with $\boldsymbol{0 \leq e_{\{true,false\}} \leq 1}$ and $\boldsymbol{e_{true} + e_{false} = 1}$. The concrete value of a node's evidence is controlled by the reliability of the episodic hypothesis represented by that node. The more reliable the hypothesis is, the "harder" is the evidence. For a reliability factor $\boldsymbol{r = 1}$, the evidence is set to $\vec{\boldsymbol{e}} = \boldsymbol{(1, 0)^T}$ according to

$$(\boldsymbol{e_{true}}, \boldsymbol{e_{false}})^{\boldsymbol{T}} \quad = \quad \boldsymbol{(0.5(1 + r), 0.5(1 - r))^T} \, , \tag{6.8}$$

and to $\vec{\boldsymbol{e}} = \boldsymbol{(0.5, 0.5)^T}$ for $\boldsymbol{r = 0}$, which is equivalent to an unobserved variable with no assigned evidence.

Solving Conflicts

By means of the conflict value, a means has been presented to detect whether specific memory context does not fulfill expectation encoded in the Bayesian network model. How can this conflict now be solved in the memory? The answer is simple: By reducing the reliability of the involved memory elements. Because of the soft evidences that incorporate the reliability of the hypotheses, this will indeed result in a reduced conflict value. When detecting a conflict the process decreases the reliability $\boldsymbol{r_i}$ of all hypotheses $\boldsymbol{i}$ that are modeled in the respective FDC according to

$$\bar{\boldsymbol{r_i}} \quad = \quad \boldsymbol{f \cdot r_i} \qquad \text{with } \boldsymbol{0 < f < 1}, \tag{6.9}$$

which in consequence decreases the conflict value. This allows other processes to potentially re-validate memory elements and probably increase their reliability value again. This might for example occur due to some new support by means of the hypotheses anchoring. Or it will cause the specific memory elements to be discarded from the memory by the forgetting process that will be presented as last inner memory process in this chapter.

An alternative idea is to compute the posterior probability (the *belief*) for each node in the network and compare it to the assigned evidence in order to identify the "guilty" memory elements, as the one which belief and evidence values have the largest difference. This one can than be discarded from the memory directly in order to decrease the conflict value. But this idea has not yet been investigated in detail.

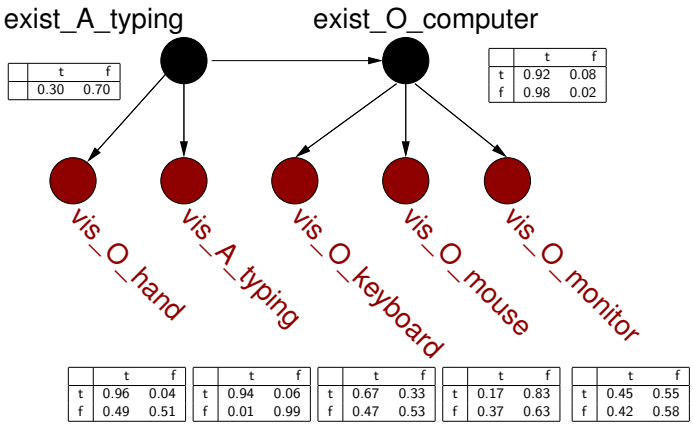


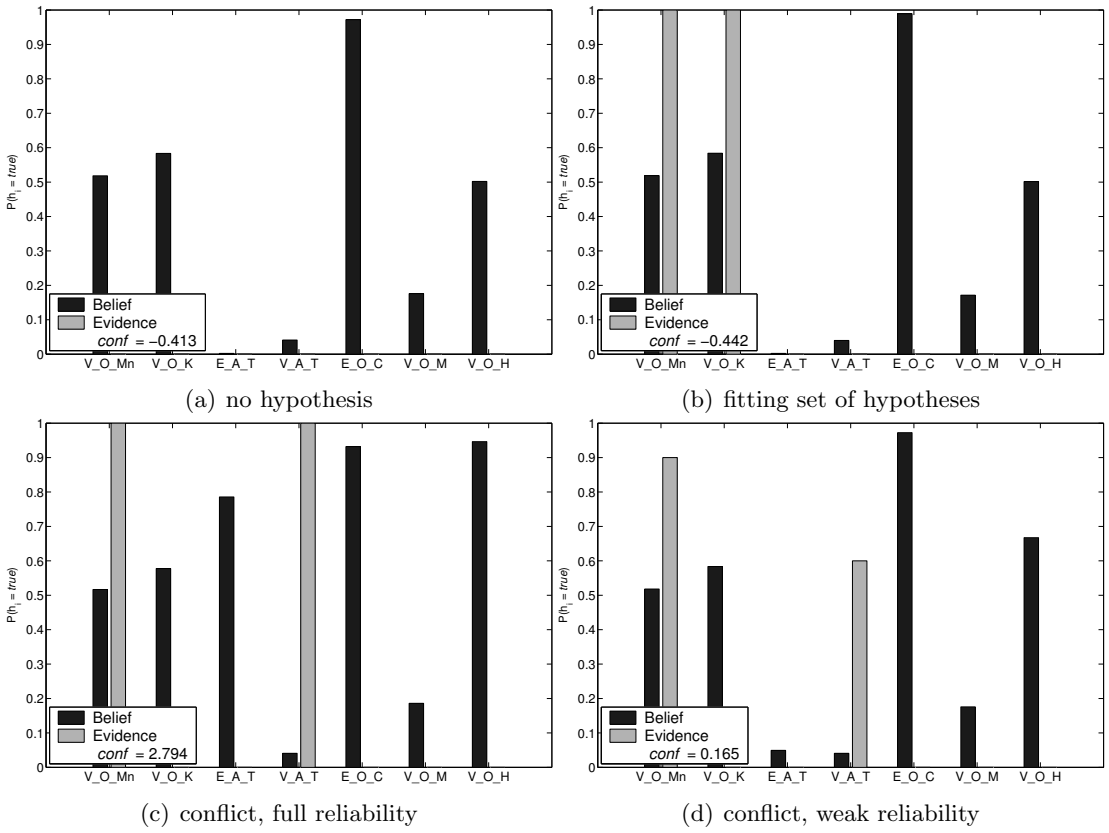
Figure 6.4: The Bayesian network for the exemplary setup scenario: Observed nodes are shown in red, unobserved in black. Conditional probability tables are shown for each node, as they are obtained in the trained phase.

Parameter Estimation

Although the structure of an FDC is defined manually by means of the underlying structure of the Bayesian network, its parameters are estimated from training data. This parameter estimation can easily be done mostly in the same way as the analysis of the memory content in regular operation. All that has to be done, is to switch the consistency validation memory process into the learning mode and let it analyze the content of the current episodic memory in situations that have been assured as being free of errors. For different situations, the corresponding hypotheses relevant for an FDC to be trained are queried for existence in the memory. The respective soft evidence obtained from the reliability values are assigned to the particular variables of the Bayesian network and the internal probability tables are estimated from these values. Additionally, any unobserved nodes need to be manually allocated in the training phase which can be done interactively by the system, e.g., by means of questions to the user like, "Is there a computer in the scene?". This does not need to be done for every training step, because an EM-algorithm according to Lauritzen [92] is applied to estimate the parameters of the Bayesian network from only partially observed data. Alternatively, FDCs can be trained in an offline fashion using manually annotated video data in order to eliminate perceptual errors explicitly.

6.5.2 An Exemplary Context Model

An example that relates an action and different objects shall clarify this approach: Imagine the scenario of a typical computer setup. "Typing" is an action, that is inherently plausible in the context of computer and that is also implicitly related to a hand, that performs this action. Furthermore, one will agree, that todays computers usually have a keyboard, a mouse, and a monitor. Putting these assumptions into an FDC results in a Bayesian network as it is displayed in Fig. 6.4. This figure also displays an example of the modeling paradigms used to integrate action and object hypotheses with Bayesian networks. All nodes starting with ***vis_*** denote *observable* variables, whereas ***exist_***-nodes are *unobserved* and can only be inferred by the process. Observable nodes correspond to memory elements in the episodic memory, which can be checked for existence and respective values can be assigned. They are usually modeled as being dependent from respective unobserved nodes, a



(a) no hypothesis (b) fitting set of hypotheses

(c) conflict, full reliability (d) conflict, weak reliability

Figure 6.5: Beliefs and *conf*-values of four exemplary cases.

apparent in the example network for the observed nodes ***vis_O_keyboard***, ***vis_O_mouse*** and ***vis_O_monitor***. Since actions typically require some kind of object context (as it is true for ***exist_A_typing*** and ***exist_O_computer***), this functional relation is modeled as directed arc from the action node to the object node in this example.

We exemplarily trained this FDC with some data obtained from annotated video sequences. For each assignment of the memory, the conflict value $conf(\boldsymbol{e})$ and the beliefs $\boldsymbol{P(h_i = true)}$ for each hypothesis $\boldsymbol{h_i}$ are calculated by propagating the evidences obtained from a current simulated memory content through the Bayesian network. Resulting values for different memory configurations with their respective evidences are displayed in Figure 6.5. Diagrams 6.5(a) and 6.5(b) depict the results for a consistent memory content in the sense of the evaluated FDC, since $conf(\boldsymbol{e}) < \mathbf{0}$ is true for these two configurations. On the one hand, 6.5(a) shows the case of having no relevant hypothesis available, while on the other hand, 6.5(b) includes a high-reliable hypothesis on ***vis_O_monitor*** (v_O_Mn) and ***vis_O_keyboard*** (v_O_K), which support each other very well.

On the contrary, the two configurations 6.5(c) and 6.5(d) held conflicting hypotheses, as it is indicated by $conf(\boldsymbol{e}) > \mathbf{0}$. Both configurations contain hypotheses of ***vis_O_monitor*** (v_O_Mn) and ***vis_A_typing*** (v_A_T) but no hypothesis on ***vis_O_keyboard*** (v_O_K) which violates the trained model that expects a keyboard to be visible while typing is recognized. As stated before, in case of a conflict, the reliability of a hypothesis is decreased. The effect of incorporating the reliability using soft evidences is depicted in 6.5(d). Doubting hypotheses by lowering their reliability ($\boldsymbol{r_{v\_A\_T} = 0.6}$) allows the reduction of the conflict potential.

6.5.3 Consistency Validation as Inner Memory Process

The presented approach is implemented as an inner memory process in the visual action memory. This process does not register on any specific events as the ones described before. Instead, it *queries* the memory content from time to time at a fixed rate of usually $\mathbf{1}\boldsymbol{Hz}$. It performs its analyze and updates the reliability value of respective memory elements, if a conflict has been detected according to the scheme introduced before. Furthermore, the `UPDATE` timestamp is set accordingly to reflect that this memory element has indeed been examined and modified. Hence, it can be seen as a *re-organization* memory process, that applies a *deliberative* analysis of the memory content.

6.6 Keeping the Relevant — Forgetting the Useless

Up to now, the content of the memory has always grown. The processes discussed so far submitted memory elements or updated the content of the memory, but never removed anything from it. This is consistent, because none of the individual processes really can independently decide which memory elements are still needed by others. But it has already been mentioned in the general discussion of the memory concept in section 5.1.3, that it is a foundation of memories, that they have a limited capacity. Hence, a *forgetting* is required, that avoids an overflow of the memory's capacities, and that restrict its content to useful information. Baddeley [8, page 5] states, that

> *the process of forgetting is one whereby the important features are filtered out and preserved, while irrelevant or predictable detail is [...] destroyed [...].*

This definition is a very general one, and the label "irrelevant" may be attached to dedicated features – or memory elements in the notion of the visual active memory concept – as a consequence of many different causes. What is considered to be irrelevant highly depends on the specific point of view. By the definition of the layers in the memory given in section 5.1.3, a first distinction is possible. Different expectations regarding the time to live, relevance, and the linkage to other memory elements have been introduced for the different layers and memory types. Obviously, any forgetting process has to consider these expectations and must treat the memory elements accordingly.

It has been stated, for example, that perceptual memory elements usually only have a very short time to live, except they are linked to some other content. Hence, memory elements of this type can mainly be discarded based on the timestamps information they contain and their links to other elements.

For the episodic memory, the intended equivalence between memory elements and real world entities has already been discussed. As a consequence, such memory elements last a longer time in the memory. Depending on the task, they should be removed, as soon they have no longer a corresponding entity in the real world, or they should still be kept in the memory for later retrieval if they are considered as still being relevant.

The anchoring process outlined in the last section, provides no means to directly decide whether a hypothesis is irrelevant and also cannot discard an outdated one from the episodic memory. But it can indirectly cause memory elements to be removed, by assigning a low reliability and not updating them anymore. If the episodic memory shall only contain current episodes, any hypotheses, that is not reliable, or has not been updated for a longer period of time, can be considered as irrelevant, just because it is no longer based on a

real world entity in the scene. Alternatively, the episodic memory can be marked as being outdated and is left in episodic memory.

As no process can independently decide whether a memory element is irrelevant and should be removed, the visual active memory concept proposes a generic memory process for that purpose. An active forgetting is implemented as inner memory process accounting for the different aspects described above. This forgetting process is inherently "inner" as it only discards memory elements based on their individual content. Due to the information-driven integration approach of the active memory infrastructure and the consistent compliance of the memory processes to the hypotheses concept, the implementation is rather simple. The process is triggered periodically to discard irrelevant and outdated entities from the memory. It can be thought of as a *garbage collector*. What is considered irrelevant can simply be specified by XPath statements thanks to the information-oriented representation of the memory elements. An example shall clarify the approach: Invoking the remove method of the memory interface with the following XPath discards all memory elements that have not been updated since a given time (in this example **12323**):

```
/*/*/TIMESTAMPS[UPDATED/@value<12323]
```
(6.10)

Note, that this XPath statement fits any episodic instance in the definition of the type hierarchy shown in Fig. 5.4. Therefore, forgetting processes implemented on this basis a very generic ones. Similarly, more specific XPath statement allow to actively forget according to the other schemes outlined above. It is also possible to only discard episodic instances concerned with objects from the memory, while recognized actions are remembered, for instance. Likewise, a threshold for the reliability value can also be applied in order remove unreliable memory elements.

In an integrated system, several instances of this generic forgetting IMP are running, each specifically discarding memory elements according to the definitions of the respective layers in the memory architecture and the given task. The proposed forgetting functionality itself is rather simple. Note that it actually performs no interpretation of the content of the memory itself. All interpretation of the information is delegated to the active memory infrastructure by means of the XPath specifications. Only the interplay with other generic components like *hypothesis anchoring* (section 6.4.1) and *consistency validation* (section 6.5) unveils the potentialities of this basically simple forgetting implementation. It is a foundation of the active memory concept that memory processes usually do not directly remove memory elements, but only indirectly cause a removal by the forgetting process by lowering the reliability value, for instance. This ensures, that no memory element is removed that is still needed, and that a referential integrity can be assured, if the respective forgetting process accounts for it.

6.7 Summary and Contribution

This chapter picked up the concepts of the visual active memory to discuss memory processes that actually make this memory an "active" one. While the previous chapter focused on coordination and representation, this chapter was mainly concerned with algorithms and processing on the basis of the proposed representations. Therefore, I started this chapter by identifying general processing paths in an interactive system built on the basis of the VAM concept.

Afterwards, I distinguished *inner* (IMP) and *outer* memory processes (OMP). The main

difference is that inner processes exclusively work on the internal memory representation. They do not have any direct connection to the outer world; this connection is provided by outer memory processes. Almost all functionalities presented in chapters 3 and 4 therefore constitute outer memory processes.

As a major contribution in this chapter, I described three generic inner memory processes, that together with the VAM concept define the cognitive architecture of an active memory. They implement organization and assessment of memory content, fuse information, and relate memory elements to each other.

In detail, I showed how percepts can be mapped to episodes by means of *hypothesis anchoring*. Furthermore, it has been presented how the *context* of memory elements can be analyzed in order to detect conflicts in the episodic memory. Finally, *forgetting* has been introduced as a generic process to discard unreliable or outdated information from the memory. This chapter showed how information from outer memory processes is stored, (re-)organized, fused, and discarded in the memory. The interplay of memory processes that are coordinated by the active memory infrastructure paves the way to develop cognitive vision systems for different application scenarios, in particular for assistance. The next chapter will outline how the different memory processes work together in a specific scenario.

7 System Architecture of the VAMPIRE Assistance System

The cheapest, fastest, and most reliable components of a computer system are those that aren't there.
[Gordon Bell, *1934]

Up to know, the concept of an ego-vision assistance system and of cognitive vision systems applying visual active memory concepts has always been looked at from a very abstract perspective. Different approaches to perceive the environment from the ego-vision perspective and to interact with the user have been presented. And with the VAM and the concept of inner and outer memory processes an approach for the integration of the system in general and the interplay of its components has been introduced.

7.1 System Capabilities and Scenario

It is now time, to substantiate all these concepts and approaches and to show how they form an integrated system that develops abilities of a cognitive system and can be purposeful as an assistant. Therefore, the use case of general assistance is broken down to a more specific scenario in which a system is envisioned that can assist the user in the manipulation of arbitrary, rigid objects. These objects are placed on top of a table. This table environment, the tasks are accomplished in, defines a planar sub-scene as outlined in Sec. 3.1 and can be seen as a *view context* as defined in Sec. 3.3.3. Taking the VAMPIRE AR gear outlined in Sec. 4.1 as the hardware basis, different capabilities have to be integrated on that platform to come up with a wearable assistance system. It shall be able to recognize objects, step-wise instruct the user what to do with these objects, and supervises the correct accomplishment of these instructions. From the perceptual point of view, all capabilities to implement such a system are at hand using the approaches presented in chapter 3. Visualization capabilities have been implemented that allow the system not only to display instructions but also to highlight referenced objects and even reference objects outside the current field of view using arrows (see chapter 4).

Summarized, the realized assistance system has the following capabilities:

(i) In order to achieve the intended flexibility, it will be possible to teach the system relevant objects in an online fashion as outlined in Sec. 3.2.2. *Feedback* by means of displaying captured object views is provided to the user as shown already in Fig. 4.3(a) on page 66 in order to avoid that a wrong view of the object is included in the training data set. Furthermore, pre-known objects can also be detected by the object detection approach (Sec. 3.2.3).

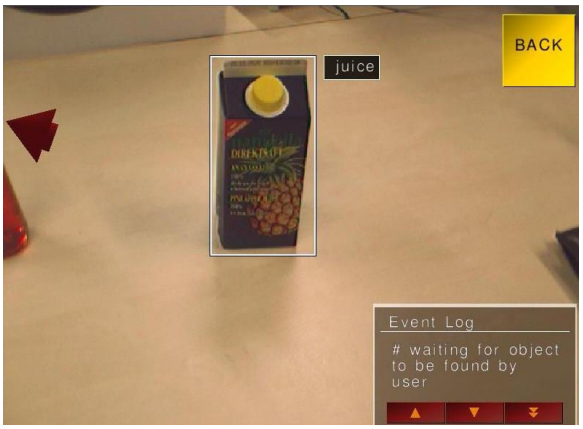


Figure 7.1: The user's view in the assistance system working on a table top. Objects are recognized, objects outside the current field of view are referenced by arrows.

(ii) The system assists in simple object manipulation tasks. A task is subdivided into sequences of distinct actions of object manipulations. The assistance system checks their respective accomplishment using the action recognition of Sec. 3.4 and prompts the user if she or he is performing incorrectly. Otherwise, it continues by prompting for the next step. An example of this prompting in a case where the user has manipulated the wrong object can be seen in the upper left of Fig. 4.3(d).

(iii) The system is able to compute the three-dimensional position of objects in the scene using the approaches outlined in Sec. 3.3.2. This allows to direct the user's attention and guide to the requested object by means of arrows when it is outside the current field of view as shown in Fig. 7.1.

(iv) The system is context- and situation-aware in its visualization and information presentation. First, the table top is seen as the space in which a task is carried out. With this context given, the system can compensate for the restricted application area of the integrated object recognition approach that works only reliably on mostly uncluttered backgrounds. The results of the VPL object recognition (Sec. 3.2.2) are thus only considered to be reliable if they reside inside the table view context. In other contexts, its perceptions are regarded as less reliable and in consequence are not displayed to the user.

(v) The user can interact with the system using speech, mouse and head gestures to control and enter system modes, trigger system functionalities and accomplish interaction cycles.

7.2 System Architecture

The different functionalities are not implemented as a monolithic block. Instead the envisioned functionality of the system results from the interplay of the different components. The system architecture must be built upon an adequate integration framework and software infrastructure. As the whole system is based on the VAM concept, the resulting *system architecture* of the ego-vision assistance system has been developed jointly with Sebastian Wrede, who proposed the AMI as the basis for the VAM approach [170].

A system architecture is generally defined as

> *a representation of a system in which there is a mapping of functionality onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and human interaction with these components.*

Following this definition proposed by Carnegie Mellon University's Software Engineering Institute glossary [20], the mapping of the outlined functionalities to components and their interplay shall be presented in the remainder of this chapter.

In order to account for the specific requirement of cognitive vision the system is implemented following the concepts of the visual active memory picking up insights presented in chapter 5. Consequently, almost all involved components constitute memory processes; they apply information-oriented representations using XML, mediate almost all information through the visual active memory, and are coordinated using memory events. Fig. 7.2 presents a sketch of the system architecture. It does not provide a complete picture of the interplay of all components, but illustrates which communication patterns are used by which components. Therefore, these different ways of communication and coordination are shown in different colors. Event-based communication is indicated by green color, red lines represent memory access using remote method invocation (RMI), and direct XCF-enabled communication is illustrated using ocher lines. In order to clarify the layout figure multiple parallel lines are integrated into one bold line. These different patterns are all provided directly by the *active memory infrastructure* [169, 170].

Using the XML-enabled communication framework (XCF)[169] location transparency is achieved. This means that the different components are spread over several computational nodes connected via high-speed network in order to overcome limitation in processing power and to achieve an adequate performance facilitating online usage of the whole system. In its final expansion stage the system has been run on five laptop computers.

7.2.1 Interfacing the AR Gear

On the top of Fig. 7.2, the AR gear is shown in a box together with two sole components that actually are no memory processes. The *visual interaction server (VIS)* implements visualization and feedback capabilities as outlined in Sec. 4.2. It provides an RMI interface that allows to pass visualization commands and manages GUI interaction in terms of mouse interaction, see Sec. 4.3.

The second component on the right in the same box, *image server*, provides captured images to any component that requests them. It is designed as a server that transfers images using XCF communication patterns only on requests, implementing a "pull" strategy. By this means it allows different components to request images according to their respective demands regarding size, color space, and so on. These demands vary between the different visual perceptual components, as some require only gray-scale images, while others need only a sub-sampled color image, for instance. Applying a pull strategy here allows to reduce network load, as only the respectively required amount of data is transmitted.

7.2.2 Outer Memory Processes

On the right and left of Fig. 7.2 all *outer memory processes (OMP)* are outlined. They are identifiable by the "O" printed below each component symbol. The OMPs on the

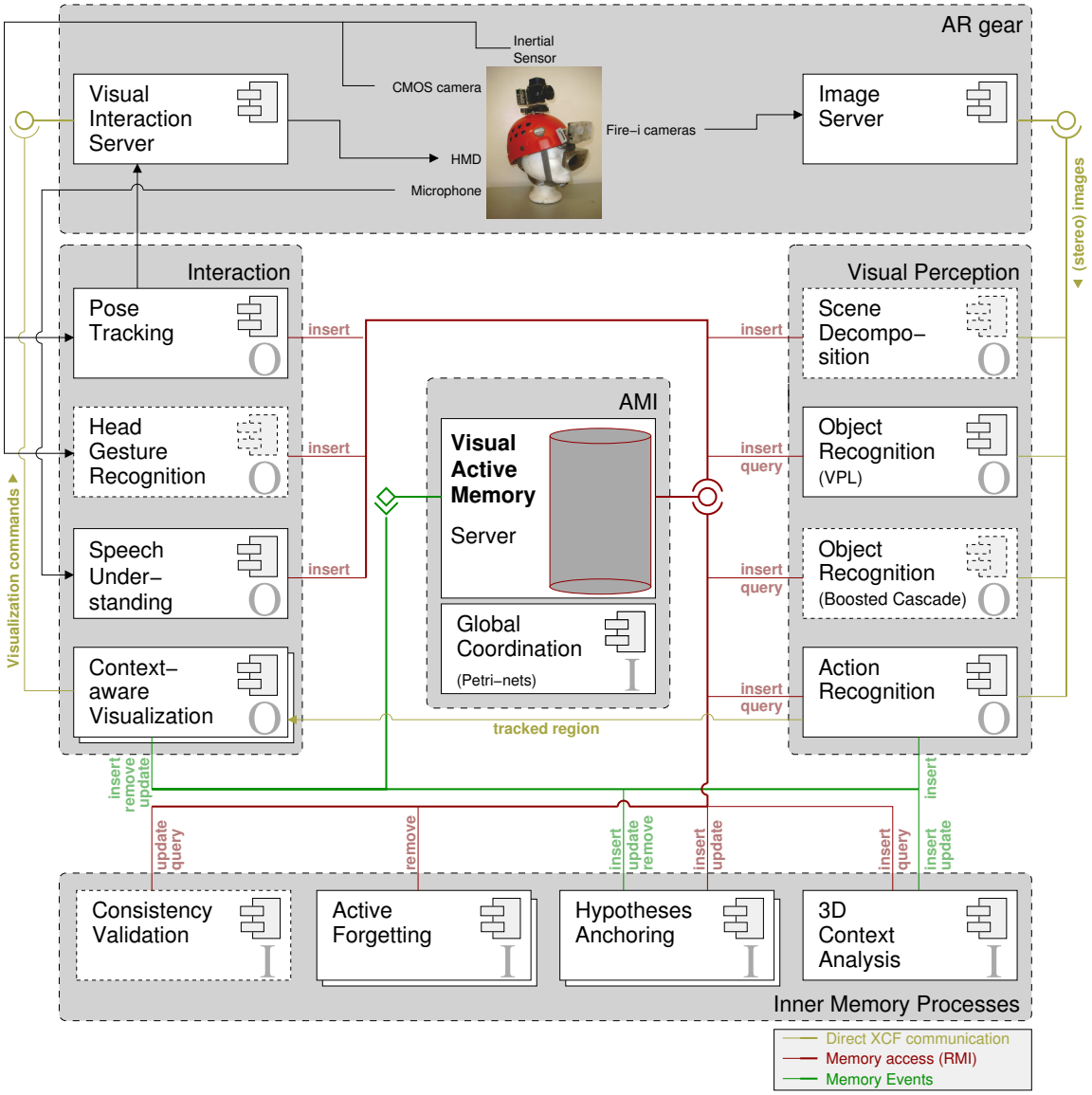


Figure 7.2: System architecture of the assistance system: Most inter-process exchange is mediated through the visual active memory. Event-based communication is indicated by green color, red lines represent RMI-based memory access, and direct XCF-enabled communication is illustrated using ocher lines. Components with dashed lines are not part of the core system that is evaluated as a whole in chapter 8.

right are mostly considered with the perception of the environment. Accordingly, they implement functionalities that have all been subject to chapter 3. The left box contains OMPs that implement interaction capabilities of the system, and have accordingly mostly been described in chapter 4. An exception constitutes the component "pose tracking" which has been introduced as a subject of 3D perception in Sec. 3.3.1. But pose tracking rather belongs to the class of techniques that perceive the user than to the environment perception processes. It has therefore been placed on the left side.

Visual Perception

Let us now briefly step through the four perceptual memory processes that are integrated in the system. The component "scene decomposition" requests stereo images as input and applies the techniques described in Sec. 3.1 to compute both, a decomposition of the scene into planar sub-scene and sets of mosaics as a compact pictorial representation. The component submits both results to the VAM whenever they are available. The component is drawn with dashed lines to indicate that it is not an indispensable part of the integrated system and therefore not integrated in the core system that has been evaluated in user studies. A decomposition of the scene into planar sub-scenes can also be pre-defined as outlined in Sec. 3.3.3 as long as the application environment in which the assistance takes place is stationary.

The two "object recognition" approaches discussed in Sec. 3.2 are both integrated in the assistance system. They analyze the most recent (monocular) image and submit their results in an asynchronous fashion whenever objects have been detected and classified. They also perform queries on the VAM in order to access the trained models that are computed and stored in the conceptual layer of the VAM as the result of the outlined training processes. Similar to the scene decomposition process the "boosted cascade" approach (Sec. 3.2.3) is not integrated as a core component, due to its discussed limitations.

As last component in the perception section of Fig. 7.2 "action recognition" is integrated. As for all other perceptual memory processes it also requests the most recent captured image from the image server to track the objects of interest. It computes the trajectory of the manipulated object and in consequence classifies this trajectory into a known activity as outlined in Sec. 3.4 In order to initialize the tracking of an object, this memory processes needs to be triggered, when an object to track is located in the current image. Therefore, action recognition also uses the event-driven notification scheme provided by the AMI to get triggered whenever such an object is available. This process will be explained in detail later in Sec. 7.3.2. If an action has been recognized, the result is again made available to all other processes by submitted it to the VAM. It should be noted that there is a direct link from the action recognition component to the context-aware visualization. The relevance of process feedback has been discussed in chapter 4. This bypass of the memory is introduced to allow feedback of the tracking procedure as fast as possible. Although the delays resulting from a mediation of information through the memory is usually short, tracked objects are manipulated and hence moved rather quickly. The visualization of the tracking process is a very time critical issue. Delays here directly affect the behavior of the user, who will slow down as the system apparently cannot take pace with the user's movement. This effect is also reported in the conducted user studies presented in Sec. 8.4. Therefore, information about tracked regions are directly transmitted to the component which is responsible for the visualization to minimize the delays.

Interaction

In order to implement the user interface, several outer memory processes that either perceive the user or articulate information are substantial parts of the system.

"Pose tracking" is implemented as outlined in Sec. 3.3.1 including the extension for fusion of data from the inertial sensor as proposed by Ribo et al. [133]. Accordingly, the pose tracking component is directly connected to as well the inertial sensor as also the CMOS camera. It computes the current 6-DOF user pose and inserts it into the VAM at a constant rate as long as it can be reliably computed.

Outer memory processes implementing two input modalities are "head gesture recognition" and "speech understanding". Their implementation has been discussed already in Sec. 4.3.2 and 4.3.1, respectively. Both insert memory elements in the VAM containing information about the semantic of the users articulation. Expressing denial or affirmation by either speech or head gesture will cause the same memory element to be submitted to the VAM.

The outer memory process (OMP) "speech recognition" is directly connected to the wireless microphone of the AR gear in order to recognize and understand natural speech. It should be noted that head gestures constitute only an optional input modality. All semantics that can be expressed using head gestures can also be expressed via speech or GUI actions. Their recognition does therefore not constitute a substantial component in the core setup of the assistance system.

The "context-aware visualization" in contrast is very crucial for the interactive loop integrating the user. Its task is to display (selective) information taken from the VAM to the user. It therefore invokes methods on the visual interaction server in order to, e.g., annotate recognized objects with their respective label (Fig. 4.3(b)). The visualization OMP makes use of the event-driven coordination as illustrated by the green connection to the VAM in Fig. 7.2. As selected information that becomes available or is modified in the memory needs immediately be articulated to the user a polling strategy is inappropriate in this case. Instead the process subscribes for the modification of objects using the XPath `/OBJECT` to get triggered when a recognized objects needs to be displayed. Whenever object-related information like the position or class is subject to change, the visualization process can promptly update the user's view. As stated in Sec. 4.2.4 it is important to only display reliable and task-related information. Therefore, the context-aware visualization component utilizes the information-driven coordination capabilities that also allow the interpretation of information by the VAM itself as will be detailed in Sec. 7.3.1. In order to account for the different visualization demands like highlighting an tracked object or provide feedback about recognized objects, the component is instantiated multiple times with different parameterizations and XPath subscriptions.

7.2.3 Inner Memory Processes

Inner memory processes (IMP) have no connection to the outer world and only communicate with the VAM directly. But they are essentially connecting the inbound and outbound outer memory processes and provide the core functionalities of the system. They are plotted with an "I" below the component symbol in Fig. 7.2.

Scanning the different IMPs in that figure from right to left, first leads us to a component termed "3D context analysis". The task of this IMP has already been outlined in Sec. 3.3.2. It subscribes to objects that are submitted to the perceptual layer of the VAM and also

to "pose" memory elements generated by the "pose tracking" process. With the pose information and the information about planar sub-scenes and view contexts queried from the VAM it can compute the three-dimensional position of objects. Following the argument of *maintainability* raised in Sec. 5.1.4 the 3D context analysis does not generate a new memory element for this information but extends the XML structure of an existing object hypothesis with the respective memory element attribute `REGION3D/CENTER3D` as shown in Fig. 5.5. It furthermore considers the 3D view contexts as outlined in Sec. 3.3.3 in order to modify the reliability of the specific object hypothesis depending on the view context as described at the beginning of this chapter in Sec. 7.1.

The next two IMPs and their high relevance have been discussed in detail previous in Sec. 6.4 for the "hypotheses anchoring" and Sec. 6.6 for the "active forgetting", respectively. Hence, I will only explain their integration in the system and their coordination here. The hypotheses anchoring can generally be instantiated for different types of perceptual memory elements fusing and anchoring these to episodic instances. Percepts are usually generated from the perceptual memory processes shown in the right of Fig. 7.2. In the concrete instance of the assistance system, the anchoring is implemented for object hypotheses. Therefore, the anchoring component subscribes for the insertion of object hypotheses, tries to anchor these according to the concepts introduced in Sec. 6.4.4, and updates or generates an object hypothesis in the episodic layer of the VAM. As memory elements in the episodic VAM can generally also be submitted, modified, or removed by other memory processes, e.g. the forgetting process, the anchoring also subscribes for memory events affecting episodic object hypotheses to consider these changes in the anchoring process.

It should be noted, that the "hypotheses anchoring" does not remove episodes from the memory. But this statement holds for all memory elements in the system as removing is subject only to the "active forgetting" process. Congruently, it is the only memory process that invokes the "remove" method on the VAM server. The forgetting process makes use of the powerful information-oriented representation of memory elements by simple removing memory elements that match a given XPath in fixed rate. The forgetting process can be instantiated with different parameterizations in order to account for different forgetting schemes, expressed as different XPaths. In the present system, unreliable information should generally be discarded from the VAM. Therefore a forgetting process removing all elements matching the XPath `//RELIABILITY[@value<0.1]` is part of the architecture. Furthermore, object percepts are removed from the memory according to their `CREATED` timestamp as they are only valid for a specific image while anchored episodic objects with valid 3D coordinates and recognized actions can be kept for a longer period of time. So must reliably detected objects not be removed, as such information is required to implement the guide functionality introduced in the scenario description at the beginning of the chapter.

The left-most IMP shown in Fig. 7.2 is the context-aware consistency validation outlined in Sec. 6.5. It is not essentially necessary, but can be plugged into the system to detect conflicts in the episodic layer of the memory. As for the outlined scenario contextual model are only of minor interest it is not part of the core system.

7.2.4 Operation Modes, Exceptions and Coordination

Local coordination between different memory processes is provided using subscriptions and the event-based notification schemes of the VAM itself. But in order to realize the different functionalities that are envisioned at the beginning of this chapter, different system states that require different configurations and parameterizations of the memory processes have to

be considered. The global petri-net based coordination component has been introduced as a part of the AMI already in Sec. 5.2 and is applied here to realize more complex context-dependent coordination of several components running in parallel.

State changes are triggered by modifications in the VAM which can be caused by interaction with the user or a result of a perceptual processes. The accomplishment of a step in a task reported by the action recognition triggers a transition in the petri-net in a similar way as the selection of a menu item in the GUI performed by the user will do. Thus, the same transition can basically be triggered by either explicit interaction, as also implicitly by an analysis of the current situation encoded in the memory content.

In our system, the user can basically switch between an assistance and a teach-in mode. In the first case the system instructs and supervises the user in task accomplishment. Therefore, the action recognition is activated, recognized objects are highlighted, and visual instructions are displayed. Accordingly, the global coordination component reconfigures or activates the respective components. In the second mode, new objects can be trained. The user is requested to capture several views interactively which are stored in the pictorial memory. After explicitly assigning a label to the set of captured views, another transition in the model triggers object learning in the object recognition components, which query the stored view from the memory. Furthermore, the tasks to accomplish by the user with assistance provided by the system are modeled using petri-nets (cf. Sec. 3.4). Each individual action of a task corresponds to a place in the petri-net and its accomplishment results in a transition in the net. Details regarding the coordination using petri-nets can be found in [172].

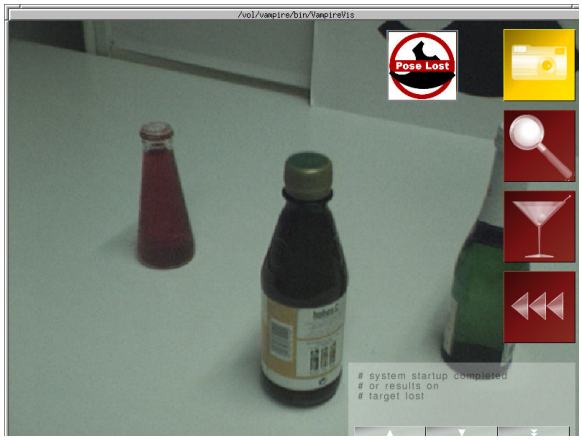


Figure 7.3: “Pose lost”.

It is also a duty of the coordination component to supervise the correct performance of other components by analyzing their contribution to the memory content and to cope with exceptions. For instance, the correctly determined pose is important for the correct function of the “3D content analysis” component, for example. If no pose is available, the guiding functionality can not be provided, objects positions cannot by determined in 3D, and in consequence anchoring works less reliable and view context cannot be discerned. Therefore, the user should be informed about any occurring problem in the pose tracking. In Sec. 4.2.2 the role of feedback has been discussed in general. Whenever no pose is available, the coordination components triggers the visualization server to display an icon as shown in Fig. 7.3, demanding the user to re-focus the “cheese target” to re-initiate the tracking process. This constitutes one specific, but rather important feedback about internal errors, that facilitate error-recoverability by interaction.

7.3 Exemplary Processing Paths

The general system architecture has now been presented. It is at a glance rather simple and well-structured, as most connections and data flow are mediated by the active memory. But the specific interplay between different components is somewhat more complicated. The

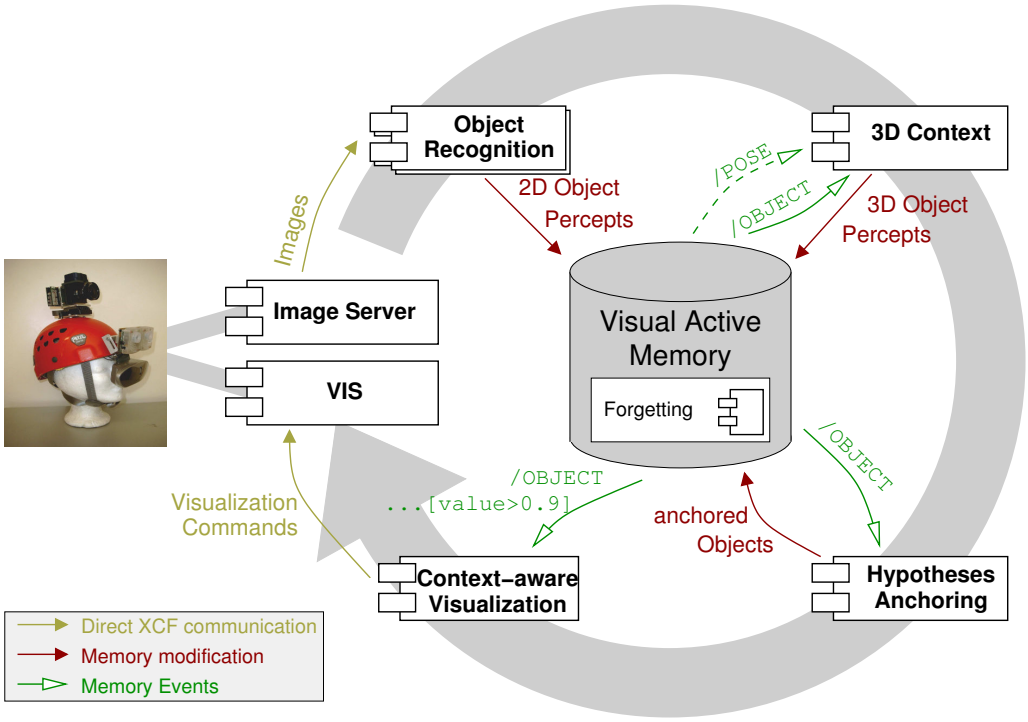


Figure 7.4: Data mediation in the cognitive assistant: Process loop from image capturing to object visualization.

advantage of the visual active memory approach lies exactly in the loose coupling of the components, letting respective connections and interplay form in a self-organized manner to some extend. In this section, two exemplary processing paths that originate from two different functionalities of the integrated system are presented and discussed. This will underline the major concepts of the active memory approach and also to clarify the general interplay of components.

7.3.1 From Images to Object Visualization

Many interesting aspects of how the system mediates data and coordinates components can be explained by following the path of the visual percept of an object from being captured by the camera to its visualization in the AR gear. In Figure 7.4 this path is outlined as logical data flow indicated by a large circular arrow. The component “Image server” is connected to the AR gear and serves images. Together with the “visual interaction server”, it closes the interaction loop. The “object recognition” component recognizes objects in the image and submits these as percepts to the memory. The component “3D context analysis” has subscribed to such percepts and gets actively triggered by the active memory whenever an object has been recognized. It adds 3D information to the percept if this can be computed from the current pose, which the component is also subscribed to. It furthermore considers the current view context in order to update the reliability of the percept according to its 3D position. In the given scenario, where relevant objects are only expected in the context of the table, the reliability of objects outside this plane is reduced. The component “hypotheses anchoring” is subscribed to the insertion of object percepts. Whenever it gets triggered by the insertion of a new percept it anchors it to the episodic memory and inserts or updates

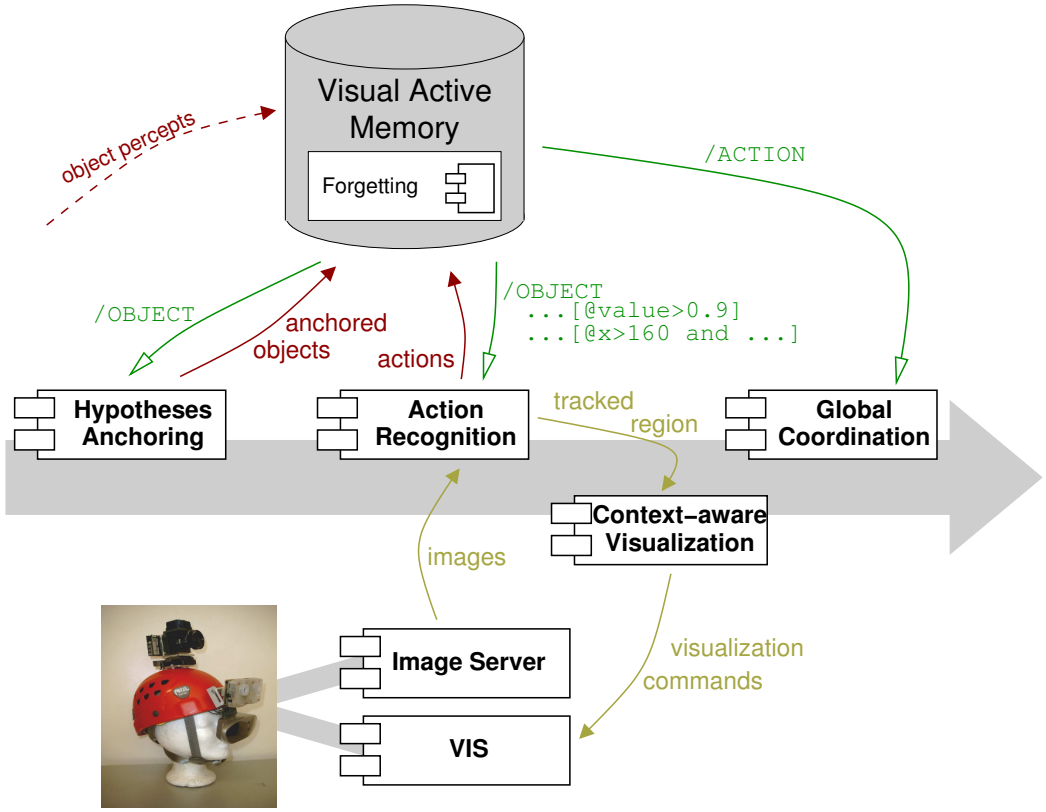


Figure 7.5: Triggering action recognition when a reliable object is in focus.

a respective episodic hypothesis.

Following the path further, the particular hypothesis in the episodic memory triggers the "context-aware visualization". Now, the advantage of the information-driven representation and integration concept of the VAM becomes obvious. It has already been discussed, that the user should only see objects that are reliably recognized. By subscribing the "context-aware visualization" process only to object hypotheses with a reliability greater than **0.9** this selection of reliable objects is already performed by the active memory itself. The corresponding XPath statement here is `/OBJECT[HYPOTHESIS/RATING/RELIABILITY@value>0.9]`. So the visualization process only gets triggered for reliable objects. In conjunction with the "3D context analysis" component, which modifies the reliability of objects depending on the view context, it also allows to limit the visualization of objects to specific view contexts as envisioned in Sec. 3.3.3 and explained above. Finally the visualization process invokes a remote method on the visualization server "VIS" to display the anchored, reliable object hypothesis to the user.

The forgetting process is sketched as a part of the active memory in Fig. 7.4 and 7.5 to indicate that it affects all the memory elements that are inserted into the memory by the different processes.

7.3.2 Triggering Action Recognition

As a second case study it is presented how action recognition gets triggered in the system. The logical process flow here is a more linear one as sketched in Fig. 7.5. Re-

consider the idea that a user usually focuses an object of interest before starting to manipulate it as it has been proposed in Sec. 3.4. In the given example this assumption is directly encoded in the subscription of the "action recognition" component to the VAM. This simply allows to implement the required model of shared attention, that is used to identify the objects of interest. Action recognition registered on reliable (condition `/OBJECT[HYPOTHESIS/RATING/RELIABILITY@value>0.9]`) and centered (`/OBJECT[REGION[CENTER[@x>=160 and @x<=480 and @y>=120 and @y<=360]]]`) object hypotheses that are available in the episodic memory. If any object hypothesis that fulfills these conditions is inserted or updated in the episodic memory the action recognition gets triggered and initiates tracking of this object in the image stream received directly from the "image server". As feedback for the user, the tracked region is directly visualized by the "context-aware visualization" component.

When the "action recognition" has recognized an action, it submits an `ACTION` percept to the memory. The "global coordination" component is amongst others subscribed to the insertion of actions (`/ACTION`) and hence gets triggered to validate whether the performed action has been correct. Depending on this validation the user is either prompted to proceed with the next step of the task or is informed about the wrong accomplishment of the current step.

7.4 Summary and Contribution

In this chapter, many of the different concepts proposed in the previous ones accumulated into an assistance system for object manipulation tasks. Building a system is not an end in itself, but a prerequisite to study integrated systems in real scenarios and also to gain insights with respect to the appropriateness of individual approaches.

Developed in collaboration with Sebastian Wrede [168], I presented the architecture for a system that can perceive objects and actions, instruct the user by means of augmented reality visualizations, and can be interactively used. It has been shown how the concept of the visual active memory allows to build this integrated system on the basis of interrelated memory processes that each implements specific functionalities demanded by the envisioned application scenario.

The core system is composed of five outer memory processes realizing perception and interaction capabilities of the system, and of three inner memory processes, connected by the central visual active memory. Four optional components have been presented and it has been shown how these can be added to the architecture, although they are not inherently necessary for the core functionalities.

Exemplary processing paths that implement core functionalities by an interplay of the different memory processes have been presented to clarify how the system actually works. The resulting cognitive assistance system can serve as a basis for evaluation of the different components in a realistic setting as also for studies from a systemic point of view as will be discussed in the next chapter.

8 Evaluation

Success does not consist in never making mistakes but
in never making the same one a second time.
[George Bernard Shaw, 1856–1950]

In Sec. 1.2 of the introduction, three central questions have been promised to be answered in this work. In the subsequent chapters answers have been proposed, but whether these are appropriate and convincing for the outlined problems has not been discussed yet. The proposed and implemented approaches must undergo some serious evaluation in order to prove their suitability for cognitive ego-vision systems that can be applied in assistance tasks. Of course, it is not only the goal of evaluation to prove the appropriateness of the proposed methodologies but also to identify their drawbacks and limits. Therefore, a combination of as well quantitative as also qualitative benchmarking is required and will be presented in this chapter.

As stated in Sec. 2.5 I will follow a scenario-driven evaluation approach in order to assess the system as a whole as well as the different components that have been proposed and implemented. The goals of the evaluation corresponding to the central questions of my work can be condensed in these following questions:

- Are the proposed perception approaches mainly presented in chapter 3 appropriate for ego-vision systems? Are they applicable in a real world scenario and how well do they cope with challenges of ego-vision scene perception outlined in Sec. 2.4.2?
- Is the system architecture using a visual active memory on the basis of the active memory infrastructure an appropriate means for the integration of the system?
- Does the ego-vision paradigm work for assistance systems, and are user's able to benefit from the close coupling to the system? Are the interaction capabilities feasible for the tasks to solve?

The first question is generally concerned with the individual components and can to a great extend be answered for each approach separately. The latter two in contrast require the system to be evaluated as a whole, incorporating also the user as an integral part of the evaluation setup.

Consequently, I take a threefold perspective on the challenge of evaluation, focusing on (i) the assessment of individual perception capabilities, (ii) the evaluation of the integrated system, and (iii) the appropriateness of the system in collaboration with the user for assistance scenarios. This chapter is coarsely outlined along these three perspectives. First, I will present and discuss evaluation results of different components for the perception from the ego-vision perception that have been presented in this work. Afterwards, we turn to the evaluation of the integrated system, focusing first on a system perspective and then on the user perspective.

8.1 The Bartender Scenario

Following the idea of scenario-driven research, a well-designed scenario is required that allows to study the questions posed. Assistance in object manipulation has been defined as a target the methods proposed in this work should help to reach. For the evaluation, this broad scenario needs to be narrowed down to actually design evaluation setups that can be used to conduct studies in. Consequently, we applied and evaluated the developed assistance system for a more specific object manipulation task: It should assist users in mixing and preparation of beverages or cocktails. This task might appear rather artificial at a first glance, but provides a scenario that is (i) familiar to most users, (ii) involves different objects and their active manipulation, and (iii) can easily be simulated in office or laboratory environments. Furthermore, objects in this scenario are generally easy to handle and allow also user studies with non expert users.

In consequence, all studies presented in this chapter have been conducted in this so-called "*bartender scenario*" or are at least motivated by it. Of course, the scenario mostly affects the studies of the integrated system, but as far as a specific scenario is demanded for the evaluation of the specific components, this bartender scenario has been chosen.

8.2 Assessment of Individual Memory Processes

Ego-vision induces several specific challenges on perception capabilities. These have in detail been discussed in chapter 2 and several different solutions have been proposed for these in the following. Although the different processes are mostly evaluated individually as said before, I will still follow an scenario-driven approach in the evaluation. All components are not evaluated with artificial or simulated data, but in real world settings, focusing on similar assistance scenarios as outlined in the previous section. The different approaches have all been designed to use only sensory equipment of the mobile VAMPIRE AR-gear presented in Sec. 4.1 and are consequently evaluated using input obtained from its camera(s) or the other sensors.

However, an assistance system as the one developed in the VAMPIRE project is composed of many different components; most of them have been discussed in chapter 3 and constitute an issue of research for their own Some have developed by project partners and have been exhaustively been evaluated by them. Accordingly, I will not present any evaluation results of these components as long as they have been already evaluated in similar scenario already and proved their appropriateness. These evaluation studies conducted by developers of the respective components have been one of the selection criteria to decide which approaches are suitable for the to be integrated in the system to implement envisioned functionalities.

Looking at chapter 3 once again it turns out that the object recognition presented in Sec. 3.2.2 has been evaluated in a scenario similar to the one discussed here by Heidemann et al. [71]. Also the 3D pose tracking approach briefly presented in Sec. 3.3.1 has been evaluated using the VAMPIRE AR gear by Ribo et al. [133] and Stock and Pinz [152]. The accuracy of the 3D object position computation is directly dependent on the accuracy of the pose tracking. It can analytically be determined depending on the distance of the object and the composition of the scene. Therefore no individual quantitative evaluation of this component seems necessary, which has been approved by the user studies (see Sec. 8.4).

I will focus the evaluation on that individual components, that I have mainly contributed to. This leaves the implementation of a pictorial memory on the basis of mosaics and the

action recognition approach as subjects to more detailed evaluation regarding the perception components discussed in chapter 3. In chapter 4 components for the interaction with the user have been presented. Although, most of these are generally more subject to evaluation in user studies, I will give some individual results regarding head gesture recognition here. That approach has especially been designed to account for specific needs in ego-vision interaction and constitutes a novel input modality comprising classification on the basis of inertial data, and shall therefore be assessed individually. Finally, evaluation results for the inner memory process that allows to validate the memory content and to detect conflicts by contextual analysis are reported in the following.

8.2.1 Mosaicing

The mosaicing approach as proposed in Sec. 3.1 provides a compact, non-redundant pictorial representation of a scene in terms of mosaics of plane sub-scenes. The focus of the evaluation is on the quality and the consistency of the mosaics as they are the final result of the presented procedure that is submitted to the pictorial memory. The integration of new pixel data into the mosaic strongly depends on the preprocessing steps, namely *Scene Decomposition*, and *Plane Tracking*. Especially the scene decomposition plays an important role as plane tracking is based on its results. Errors occurring in this processing step are spread to all the following stages and result in erroneous mosaics.

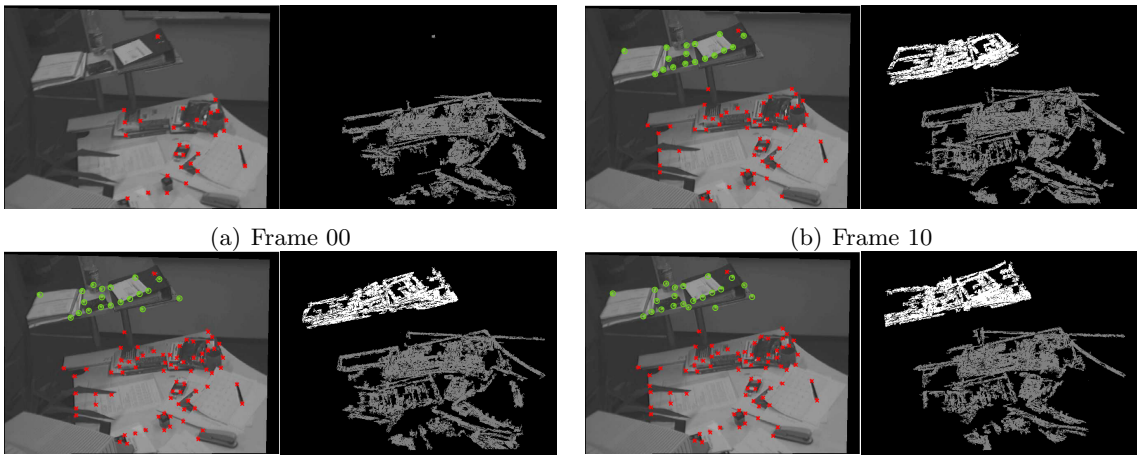
Results

For this specific evaluation, we left the restricted bartender scenario in favor for a more generic one. The mosaicing approach has been tested in different indoor environments, usually offices. The bartender scenario is too restrictive to unveil the potential of this approach, but nevertheless we still focus on assistance systems in any indoor environment here. The generalization of the scenario for this individual component already indicates how the whole system can be applied in less restricted environments in the future. Basically, the scene in such environments is composed of several almost planar sub-scenes which need to be identified and tracked by the system in order to integrate each into a mosaic. Hence, the basic assumptions hold at least for large parts of the scene as we will see in the following.

Fig. 8.1 presents the results of the scene decomposition of a sequence in an office environment. The decomposition has been limited to two dominant planes here. The desk shown in the figure has two planar surfaces which both are detected correctly. The tracked feature points are highlighted in each frame (left images) and the propagated textured areas of the planes are shown in different gray shadings (right images). Note that in frame **00** only one plane is detected, but ten frames later further points and the second plane are added and tracked from now on.

Fig 8.1 also allows us to discuss what happens with errors. In frame **00** one can see an error resulting from incorrect coplanar grouping results. One point on the elevated platform of the desk has been assigned to the main plane of the desk, indicated by a red cross. Unfortunately, the error is not corrected in the successive frames, but what also can be seen is that it does not have serious consequences on the construction of the mosaics. The propagation using dense matching cannot integrate the neighborhood of that point since it does not fit the plane model associated with that point. So the error is restricted to a very local area.

An interesting positive effect of only integrating image parts that belong to the same plane



(a) Frame 00 (b) Frame 10

(c) Frame 20 (d) Frame 30

Figure 8.1: Decomposition of the scene into two planes: Each left image displays the tracked feature points. Respectively, on the right, the textured regions of the planes are shown.

into the mosaics is depicted in Fig. 8.2. Because the carton box in the foreground of the scene does not belong to the same plane as the table with the journal lying on it, the box is omitted from the mosaic and the occlusion is eliminated. This allows to create complete views of partially occluded objects in the scene.

But how can we assess the quality of the approach? In the motivation in Sec. 3.1, parallax effects have been introduced as a problem we have to face when moving a camera arbitrarily. By integrating only planar sub-scenes into mosaics, this problem should be avoided. Hence, if a decomposition of the scene into planes would be perfect, one would expect no parallax error in the resulting mosaic for static scenes. But due to the approximately planar nature of extracted sub-scenes, errors will occur, especially at the borders of flat objects (e.g. a flat book lying on the table) as well as at the edges of extracted planes. In Fig. 8.3 some of these artefacts are highlighted. The mosaic presented here has been constructed from a sequence of 180 frames with a renew frame rate of 10 frames. That means, that every tenth frame is integrated in the mosaic while the frames in between are only used to track the plane. It can be seen, that several objects on the desk plane have not been integrated into the mosaic since they violate they planarity constraint of the dominant plane. But often their borders are integrated because the feature points on these borders still fairly match the plane model. If the camera is moved around, these points of course produce parallax errors, even if the median applied on the mosaics will smooth the result. An example of a parallax error can be seen quite well in the rightmost part of the region marked by circle 2 in Fig. 8.3. But also the other highlighted areas in that figure show some artefacts caused by objects that do not fit the plane model.

Analyzing this parallax error quantitatively requires to define an error measurement. Here, we apply the *relative parallax error*

$$\epsilon = \frac{|\{\boldsymbol{\Delta}(x,y) > \boldsymbol{\sigma} | \forall (x,y) \in \boldsymbol{P}\}|}{|\boldsymbol{P}|} \tag{8.1}$$

which is defined as the amount of pixel-wise differences $\boldsymbol{\Delta}(x,y)$ above a threshold $\boldsymbol{\sigma}$ between the tracked plane and the so far integrated mosaic, normalized by the size of the plane

(a) Initial frame (b) Plane image (c) Final Mosaic

Figure 8.2: An example for occlusion elimination (detail view).

image $\boldsymbol{P}$. The threshold $\boldsymbol{\sigma}$ is chosen to neglect noise in the original image data. To calculate this error measurement, the mosaic is warped into the current frame. In Fig. 8.4(c), the evolution of this error measure is plotted over the whole sequence which start and end frames are shown in Fig. 8.4(a) and 8.4(b), respectively. Each of these two figures 8.4(a) and 8.4(b) presents the current plane map on the left and the so far integrated mosaic on the right. In the center, the current differential image is shown which is used to compute $\boldsymbol{\delta}$ for the respective frames. As expected, the parallax error rate increased while the mosaic is

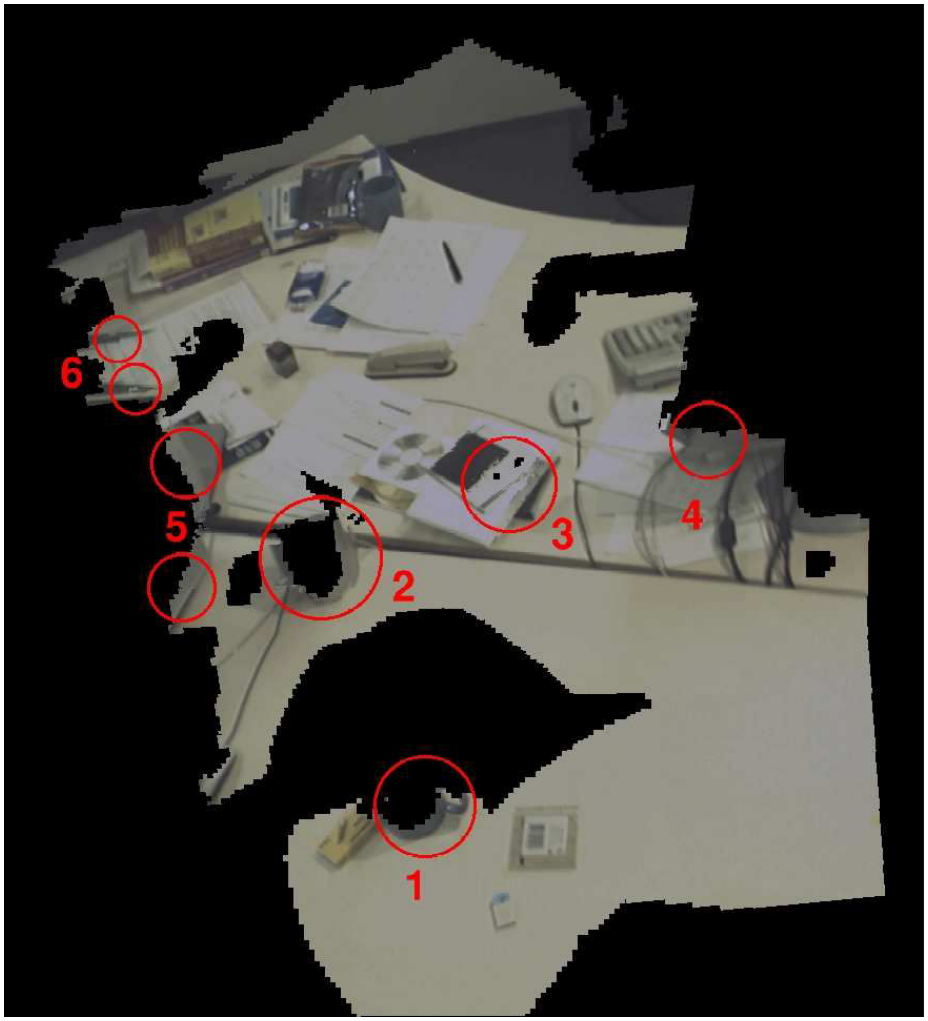
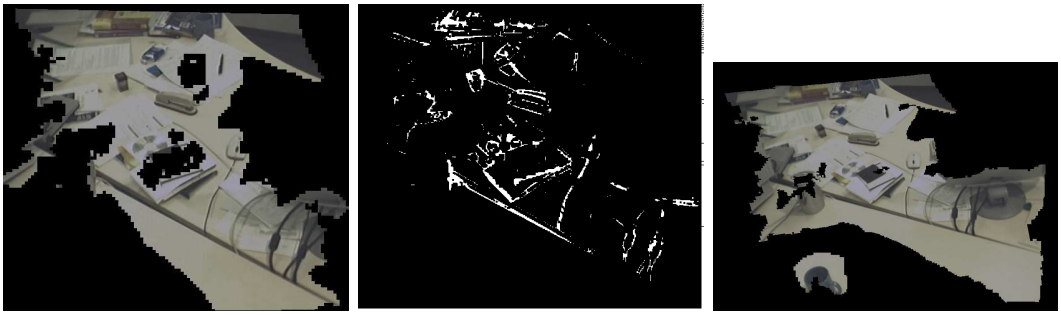
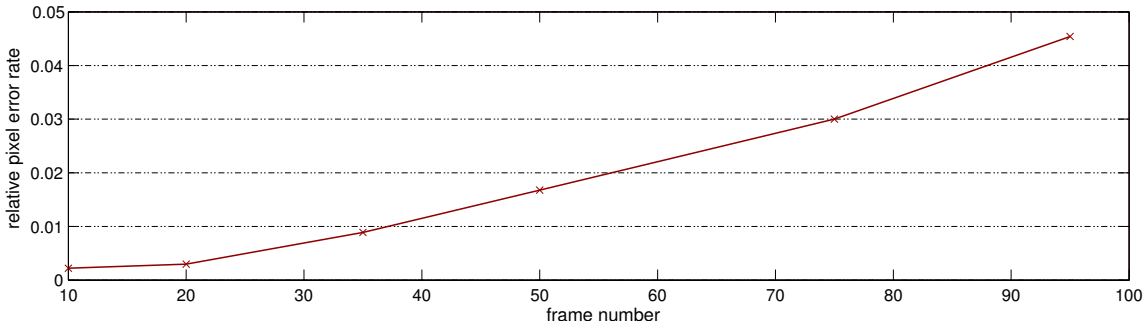


Figure 8.3: Several pixel errors in the scene due to parallax effects.

(a) Frame 10



(b) Frame 95



(c) Evolution of the relative parallax error over a sequence

Figure 8.4: The parallax error (center) is computed as difference between the single image of the tracked plane (left) and the warped mosaic (right). Errors appear as white pixels.

growing, but even in the last frame 95, errors only occur at the edges of the objects, as can be seen in Fig. 8.4(b). This error is still below **5%**, which turned out to be a typical result for the amount of camera motion that has been performed in this sequence. Nevertheless, the parallax error increases as long as new perspectives are integrated in the mosaic, because the problematic parts of the scene are still existent.

Discussion

The parallax errors in the mosaicing approach have been reported as being rather small and especially only for local, small areas of the mosaic. These are typically parts in which the plane constraint is violated, causing these parallax errors. However, most of the scene is represented without any error, as can be seen in the examples. Hence, mosaicing of planar sub-scenes generally provides a powerful means to develop a compact, non-redundant pictorial representation of the static parts of the scene.

But some drawbacks have been identified when applying the mosaicing system online. Planes, that have once been detected can robustly be tracked over long sequences of images. But if a plane gets lost anyhow, because tracking fails or the plane is completely out of sight, the recoverability of the process is still poor. Although, we tried to recover a plane by comparing each newly detected plane to existent ones using correlation on the image map, this functionality is still in its infancy.

It can be noticed, that the propagation of the plane does not stop at the border of the desk, e.g. at the top-most area of the generated mosaic in Fig. 8.3. The propagation into texture-less areas conducted as final step of the scene decomposition resulted in this apparent errors. As the border of the desk still fulfills the planarity constraint, non textured neighboring regions have been assigned to this plane, too. Looking at the parallax error in this part of the scene in Fig. 8.4(b), for instance, unveils that this is only a minor problem for the mosaic generation. The error is apparently rather small. Fortunately, the fact that there is no texture in the image that allows to assign this region to another plane also yields a small error in mosaic integration. The reason is, that the visual information that allows to distinguish different plans is the same as the one that causes errors in the mosaic integration. In consequence, this leads to convincing results in the approach.

This evaluation focused on static scenes so far. The approach in its current implementation would suffer from dynamics parts a lot and is indeed expected to give unsatisfactory results. Möller et al. [107], for instance, proposes a solution to cope with dynamic scenes by explicitly modeling the dynamic parts separately from the mosaic. But up to now, we apply the mosaicing approach during an initialization phase, in which the user looks around to familiarize the system with its environment. Here, a compact visual representation of the static scene can be acquired. The scene decomposition computed in this phase can be used to define view contexts as discussed in Sec. 3.3.3. Furthermore, the compact visual representation of the collection of planar mosaics allows to easily detect changes in the visual appearance of the scene. The representation can be updated from time to time, if necessary.

8.2.2 Action Recognition

As stated in Sec. 3.4, the action recognition in the system is generally composed of different components, namely the object recognition which provides the necessary context to classify *activities* into actions, and the analysis of the objects absolute movement to recognize these

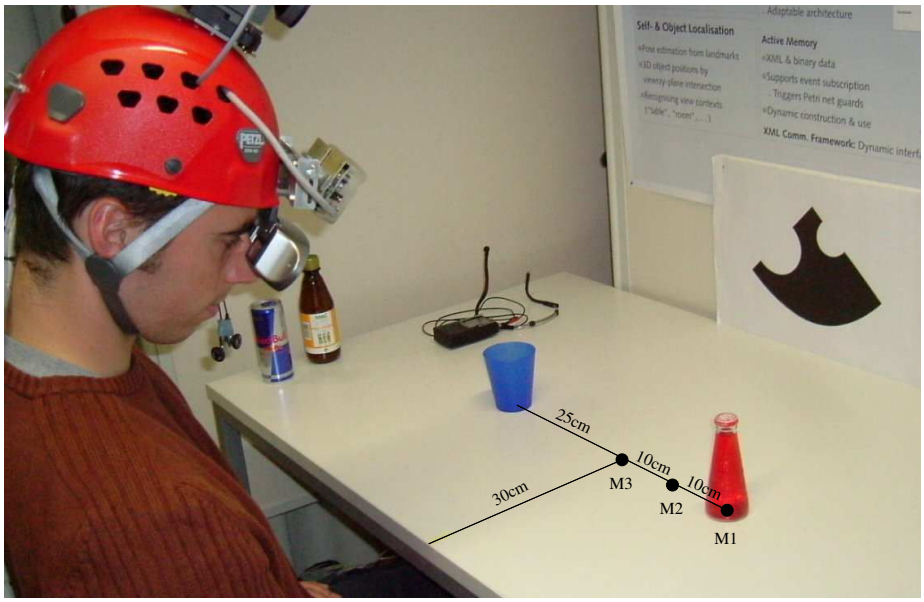


Figure 8.5: Experimental setup for action recognition: The action starts from different controlled positions $\boldsymbol{M1}$, $\boldsymbol{M2}$, and $\boldsymbol{M3}$.

activities beforehand. In the following evaluation we take the object recognition for granted, and focus on the evaluation of the activity recognition.

Generally, evaluation of activity recognition is a tough issue as it involves the user. It does not make sense to evaluate action recognition without a human performing the actions. The evaluation has been split up into two studies. The tracking accuracy has already been ascertained separately and results of this study have been presented in Sec. 3.4.2 to guide the selection of the appropriate algorithm for the integrated approach.

In the evaluation of the activity recognition composed of region tracking, background motion compensation, and recognition module, I follow a twofold strategy. First, a study is conducted under rather controlled conditions to obtain insights about the general capabilities of the chosen approach and to establish an optimal parameterization of the algorithms. Second, the system has been tested under realistic settings involving different objects and actions in the bartender scenario.

Study I: Results under controlled conditions

The first studies have been conducted under rather controlled conditions, but already as user studies using the VAMPIRE AR gear. Computational restrictions have however been evaded and the system has been evaluated offline on recorded image sequences. By that means, we could work with the full frame rate available from the camera of $\boldsymbol{15Hz}$. As tracking approach, the kernel-based tracker has been chosen in this study and the tracking has been initialized on the object by hand. Hence, only a translational motion model has been applied here. The actions are furthermore conducted in a controlled setup as illustrated in Fig. 8.5.

Motivated by the scenario of the bartender assistance, the task for the user is to perform a "pouring" activity. Besides, we also recorded other sequences in which the object is

	left	right
M1	100	95
M2	92.5	100
M3	87.5	97.5
Mean	93.3	97.5

Table 8.1: Detection rates depending on starting point and direction.

arbitrarily moved around on the table, or just left at one place with the head moving around. The activity models have been acquired by computing the mean trajectory from a small dataset of less than ten samples. These samples have been removed from the complete set to obtain the test set. The subject is instructed to grasp the object on the right and to pour its content into the mug on the left[1]. The action has been performed starting from different initial positions denoted as $\boldsymbol{M1}$, $\boldsymbol{M2}$, and $\boldsymbol{M3}$. We conducted this study with eight right-handed persons who each accomplished the task five times starting from each position, resulting in **120** positive samples. Negative samples have been randomly generated from the video sequences in which no action was performed.

The goal of this study was twofold. We did not want to actually distinguish different activities here, but focused on the issue of rejection and false positives. It is important to distinguish relevant known manipulation activities from arbitrary movements of the object. The first goal is to establish a threshold $\boldsymbol{p_{thres}}$ plotted in Fig. 3.24(c).

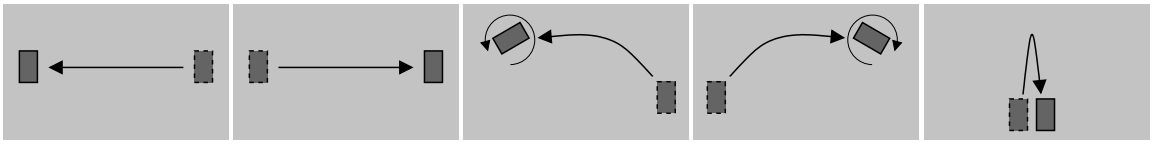
The second goal is to determine the recognition rate based on this threshold. The threshold has been chosen to achieve a zero false positive rate, which means that the system should not recognize an activity if the user only performed an arbitrary movement with the object. A small number of seven of all 120 trials had to be removed from this study, because of tracking losses due to occlusion or extreme rapid movements. Generally, tracking losses have only been observed in **1.7%** of all trials, which underlines the robustness of the kernel-based tracker.

Given these preconditions, table 8.1 presents the recognition rates achieved for the three different start positions. The two columns "left" and "right" in the table are due to the fact, that we evaluated the two components of a pouring activity individually. Users first conducted the movement to the left towards the mug, and afterwards retreated the bottle again to the right. To model the complete pouring activity one can either require both sub-activities to be detected correctly in sequence or otherwise rely on the correct detection of at least one. In our approach, we chose the second alternative, since the false positive rate is small allowing us to benefit from the increased recognition rate of this interpretation alternative. Some more evaluation experiments in the setup of this first study are reported in [134].

Study II: Result of a less constrained scenario

As a second study, we made a step towards an integrated system by distinguishing different activities and also let these be conducted in less pre-defined settings. The results of this study have also been presented in [67]. The experiments carried out in this study had the goal to rate the overall quality of action recognition in a bartender scenario. Therefore, we

[1] actually, subjects never really poured, as we wanted to avoid trouble with floating liquids in our experiments. But they have been told to behave as they would do it.



(a) move left (b) move right (c) pour left (d) pour right (e) shake

Figure 8.6: Basic activities recognized by the system.

asked subjects to perform "pouring", "moving", and "shaking" actions with different objects that have randomly been placed on the table. Again, we distinguished moving and pouring either to the left or to the right in our studies, corresponding to forth and back in the execution of the activities, respectively. These five different movements are sketched in Fig. 8.6. Models to recognize these activities have been established from one prototypical example each. Parameters of the CTR classification algorithm have been set as estimated in prior studies. In this study, we compared the two different tracking approaches, namely hyperplane and kernel-based tracking introduced in Sec. 3.4.2, to evaluate which one yields better recognition rates. In contrast to the fully integrated system, the object detection has been simulated in this setting as well as in the first study by initializing the tracking by hand. Hence, wrong object classification and detection has been evaded in the experiments.

In contrast to the study presented before, we have to face the issue of segmentation in a realistic setting. While before each sample has been hand segmented and each activity has been considered individually, now complete uninterrupted sequences covering the performance of each activity three times are examined. Hence, we not only have to consider *which* activity has been recognized, but also *when*. In order to be applicable in online systems, the activity recognition must detect an activity promptly when it is performed. Therefore, we manually annotated the image sequences with time-stamps, indicating the exact moment of activity completion as ground truth. In the evaluation an activity has only be treated to be correctly recognized if the class of the activity is correct and if it has also been detected within a two second window around the annotated time-stamp.

The study has been conducted with three different subjects that are all rather familiar with the system. Each subject performed all actions three times. Five trials in total had to be removed from the test set, because subjects violated the basic assumption of the approach: They did not keep the object in the field of view during the manipulation. These leave a total of **40** trials left for evaluation.

Results are presented in table 8.2 for both, the kernel-based and hyperplane tracking approach. For each activity the total number 'n', the correct recognized ones 'c', substitu-

activity	**kernel-based**						**hyperplane**					
	n	c	s	d	i	ER	n	c	s	d	i	ER
pour left	9	9	-	-	-	0.0	9	9	-	-	1	11.1
pour right	8	7	-	1	1	25.0	8	7	-	1	-	12.5
move left	8	5	3	-	-	37.5	8	7	-	1	-	12.5
move right	6	3	1	2	-	50.0	6	4	2	-	1	50.0
shake	9	8	-	1	-	11.1	-	-	-	-	-	-
$\sum$	40	32	4	4	1		31	27	2	2	2	
%		**80.0**	10.0	10.0	2.5	22.5		**87.1**	6.5	6.5	6.5	19.4

Table 8.2: Results of activity recognition evaluation.

	Kernel-based					Hyperplane				
	pour left	move left	pour right	move right	shake	pour left	move left	pour right	move right	shake
pour left	100					100				
move left	37	63					100			
pour right			87					100		
move right			16	50				33	67	
shake					89					-

Table 8.3: Confusion matrix showing substitutions in activity recognition; results in [%].

tions 's', deletions 'd', and insertions 'i' are listed. Based on these values an *error rate* $\boldsymbol{ER = \frac{(s+d+i)}{n}}$ is calculated taking into account not only misclassifications but also incorporating false positives (i) and false negatives (d). These result from activity recognitions that occurred outside the defined acceptance window of two seconds, and from model end probabilities incorrectly lying below the threshold $\boldsymbol{p_{end} < p_{thres}}$, respectively.

In this study, tracking worked reliably for the kernel-based tracker in all trials. For the hyperplane tracker, robustness of the tracking was indeed an issue. Especially for the "shaking" activity, it was impossible to track the manipulated object in a longer sequence without losing it. Therefore, no meaningful results for this activity could be determined. The reason are presumably blurring effects that are caused by the fast movement of the object while it is shaken. The hyperplane tracker is much more sensitive to such kind of image distortions than the kernel-based tracking approach.

In the last row of table 8.2, overall rates can be found. The recognition rate is printed in bold face. It can be seen, that apparently activity recognition works better with the hyperplane approaches (**87.1%** recognition rate) than for the experiments applying the kernel-based tracker (**80.0%**), despite the fact that it cannot be applied on fast motions. Let us have a closer look on this observation by looking at the confusion matrix shown in table 8.3. The confusion matrix provides more detailed information about the substitutions (s) indicated in table 8.2. It is displayed, which activity has actually been recognized in relation to the annotated ground truth. The latter is denoted in the rows, the recognition result in the columns of the table. The difference to **100%** in each respective row is due to the insertions and deletions that occurred. What directly can be seen is that only some classes are confused by the recognition process. Both tracking approaches are perfectly capable to distinguish the direction (left, right), but sometimes confuse whether it is a pouring or a moving activity. This is no surprising result for the kernel-based tracker, as the translational models for "pour left" and "move left" are expect to be rather equal, as well as the models for "pour right" and "move right" as sketched in Fig. 8.6. The moving activities are likely to be confused with pouring for this approach as indicated by the result in the left part of table 8.3.

For the hyperplane tracker with its model incorporating also rotation as a feature, the results are expected to be more promising with respect to the discriminating capabilities. This expectation is fulfilled for the "pour left" and "move left" activities, which can now be distinguished perfectly. The bad rate for distinguishing the same movement but accomplished to the right are however a bit surprising, because the hyperplane is also not recognizing these with good performance. This might be explainable by the observation, that the tracking is much less accurate with the hyperplane tracker, especially with respect

to the rotational part of the object's motion.

Discussion

The results of the two studies are quite encouraging with respect to the applicability of our approach to the bartender assistant scenario. Some basic activities can be distinguished and any kind of relevant activity can reliably detected. The good recognition rates result from models that have been acquired from very few training samples prove the high generalization ability of the approach.

However, significant differences between the two tracking approaches need to be discussed. The results of the studies conducted with respect to the performance of the action recognition sub-system presented here, widely correspond to the insights gain from the evaluation of the two tracking approaches presented already in Sec. 3.4.2. The hyperplane tracker theoretically has a higher potential to discriminate different activities, but suffers from reduced robustness. Its applicability in real world scenarios is therefore restricted.

Results obtained for the kernel-based tracker are a little less convincing in terms of recognition rates, but proved to be robust enough for the scenario. In practice, it is more important that the action recognition generates only very few false positives, as this would trigger the system to assume that an action has been performed and to proceed to the next instruction. The focus is therefore more on the detection, that an action has been accomplished and fewer on the determination which action it actually was. This detection can pretty reliably be achieved as indicated by the few insertions in the second study and the high detection rates presented for the first study.

Based on the evaluation of the action recognition approaches, the optimal solution for the action recognition component integrated in the core assistance system has been identified as follows: The kernel-based approach has been selected for the visual region tracking. The threshold $\boldsymbol{p_{thres}}$ has been set in order to minimize false positive detection rates. And finally, orientation-dependent activities like pouring and moving are considered to be recognized as soon as either the "towards/left" or the "back/right" movements are submitted to the memory by the action recognition component, ignoring which one exactly has been performed. This to some extend also allows to be more independent from right- or left-handed execution of the activities.

8.2.3 Head Gesture Recognition

Head gestures have been presented as a means to control the GUI of the visual interaction server in particular and also to express affirmation and denial in general interaction situations. In order to be of any use in such situations and be accepted by the user as a useful input modality, the gestures must be reliably detected. As for the action recognition, rejection of any arbitrary movements that not constitute any meaningful head gesture is important. It would dramatically decrease the acceptability of head gestures as an input modality if an important question posed by the system is accidently being answered by the user, because she or he just moved the head to think about the question, for instance.

To assess the recognition capabilities of the head gesture recognition component an individual study has been conducted. The study is conducted using VAMPIRE AR gear. For this study, only the data obtained from the inertial sensor is analyzed to recognize head gestures. The quantitative performance evaluation was carried out by means of a series of

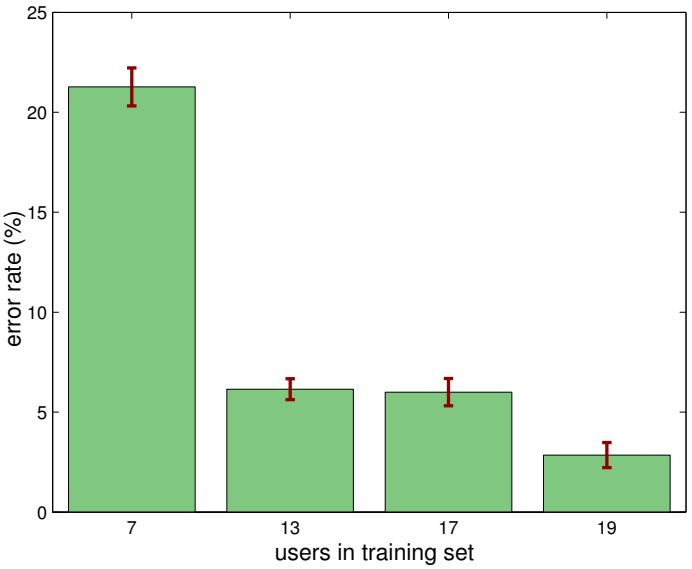


Figure 8.7: Best recognition results (measured in terms of error rate) obtained from considering various sized training sets. The (95%) confidence intervals are drawn at each bar.

user experiments.

In cooperation with López Romero [96] inertial data was recorded while the subjects performed these respective head gestures. Altogether 26 subjects (14 female, 12 male) were asked to perform the gestures *nod*, *shake*, *up*, *down*. Besides that, we also recorded data from persons while they keep their head almost steady (denoted as *quiet* in the following), move it arbitrarily (*noise*), or perform rapid movements (*jump*), like standing up or jumping. This data has been manually annotated according to the performed gestures and serves as training and test set for the evaluation.

While the rationale for considering the gestures *nod*, *shake*, *up*, and *down* was motivated in the discussion of the interaction examples in Sec. 4.3.2 already, the reason for considering pseudo gestures like *quiet*, *noise* and *jump* is not immediately apparent. These motion pattern have been chosen to obtain models for frequently reappearing patterns in the inertial sensor data which, however, do not carry a meaning with respect to the interaction tasks. Models representing these patterns are necessary to implement rejection classes in the system, in order to distinguish semantically rich movements from others. Hence, recognizing these so-called "pseudo head gestures" reduces the false positive rate in meaningful head gesture classification. They are currently not associated with any semantic and thus do not trigger any specific action in the system.

Results

For the evaluation of the HMM-based head gesture recognition technique, we split the users into two groups and used the data from one group to train the classifiers; the data acquired from the other group was used for testing.

The system was trained with the chosen training data as outlined in Sec. 4.3.2. Fig. 8.7 displays the results in terms of error rates in single gesture recognition for different partitions of our data into training- and test groups classifying into all seven classes. Obviously, even

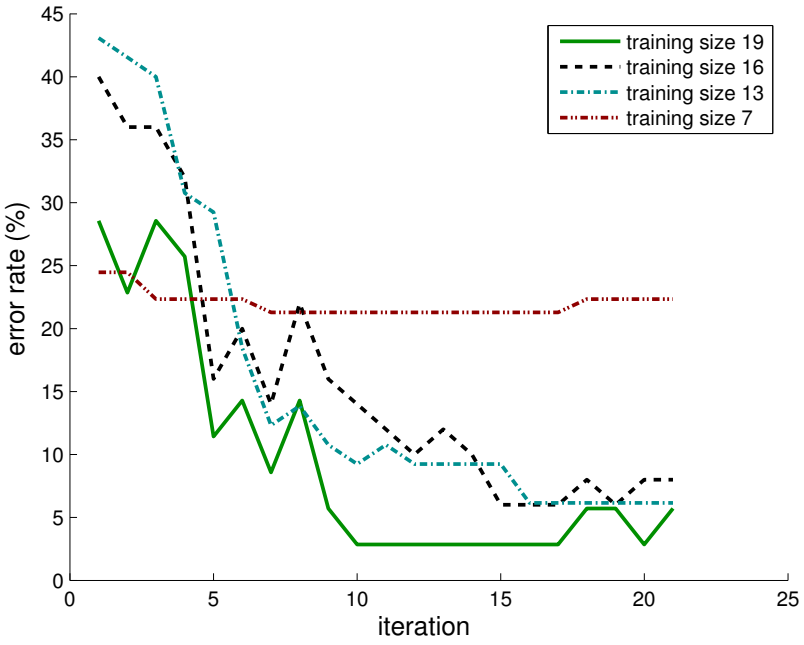


Figure 8.8: Error rates with respect to the number of training iterations on four different training sets.

training with only few data recorded from just 13 different subjects still results in a reliable recognition performance (error rate: 6.2%). If more data was used for the training, even better results could be achieved: for training based on the data gathered from 19 users, the error rate drops to 2.8%. These results are encouraging and hint that even better performance might be achieved if the gesture classifier was trained with yet more examples.

I furthermore studied the progression of the error rate with respect to the number of training iterations. The results of these studies are presented in Fig. 8.8. The figure shows that for all experiments the lowest error rates are achieved after 15 HMM training iterations. Training beyond 15 iterations resulted in over-fitted classifiers which exactly reproduced the training data, but showed limited generalization capabilities and are therefore of little use in practice.

Discussion

From the reported findings it can be concluded that HMM-based classification of inertial motion tracker data using Fourier descriptors provides a convenient and reliable method for head gesture recognition. Though training is straightforward and inexpensive, the resulting classifiers yield low error rates. Although the range of head gestures is indeed somewhat limited to specific semantics, they can especially accompany other input modalities, for instance as a fallback solution for erroneous speech recognition.

Furthermore, the so-called "pseudo head gestures" *quiet*, *noise*, and *jump* can be beneficial for the integrated system no only as rejection models. The class *quiet* can, for instance, constitute a significant cue in order to let the system recognize that the user is either currently focusing something in the scene or mentally busy. Whatever the reason behind the steadiness might be, it can at least be used to assure, that the current visual input is not blurred and can be considered rather stable. Although this aspect has not been investigated

any further yet, it seems that an analysis of the inertial data yields improvements also for the general perception of the system. It might especially be useful to consider visual input that has been gathered during a period of rapid movement less reliable in the active memory.

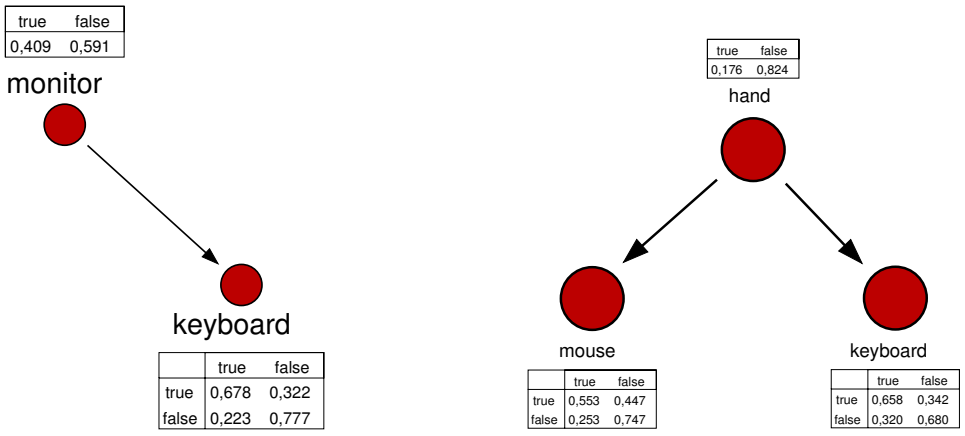
8.2.4 Memory Consistency Validation

The memory consistency validation has been presented as an inner process that interrelates episodic memory elements. The evaluation has not been conducted in the bartender scenario, but with some rather simple object constellations in office setups. In order to evaluate the consistency validation approach, some basic *functional dependency concepts (FDC)* have been defined. The evaluation is based on annotated video sequences, but conducted in the visual active memory framework to prove the general applicability of the approach. However, the object recognition memory process has been substituted by a simulator that submits object percepts to the memory at a rate of 10Hz, simulating the object recognition process as realistic as possible with respect to the submission rate. This allowed to abstain from specific object recognition issues and allows also to study objects that are hard to detect with the approaches presented in Sec. 3.2.

The consistency validation regards individual episodic instances in context to each other and checks if they fulfill a given expectation model. Although the memory process can handle many different types of hypotheses generally, we concentrated on context of object hypotheses for this evaluation. The structure of the evaluated FDCs is designed by hand.

A Model relating Keyboard and Monitor

The simplest example of a FDC contains of only two nodes which model the dependency of two hypothesis types. The particular network is presented in figure 8.9(a) along with its parameters obtained in the training phase. This FDC models a dependency between the

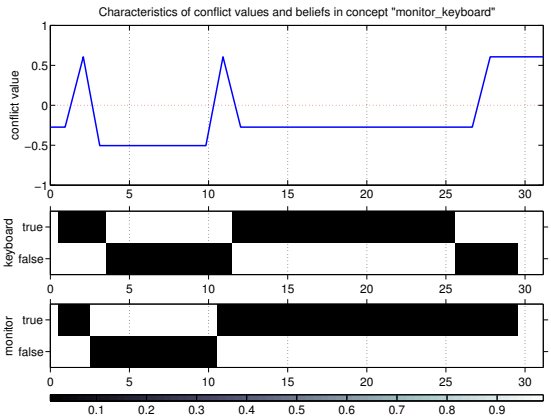


(a) Monitor and keyboard

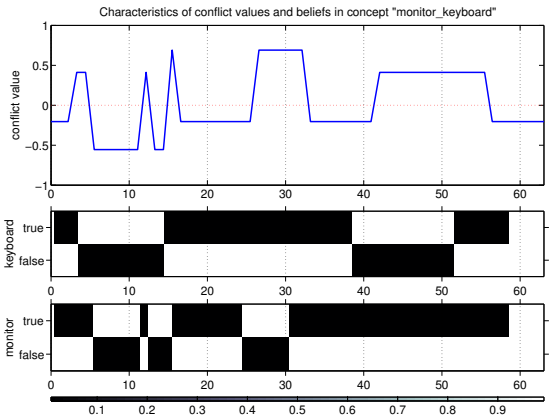
(b) Hand, mouse and keyboard

Figure 8.9: Trained network for FDCs hand_mouse_keyboard and monitor_keyboard.

existence of a monitor and computer in the same spatial context. This spatial context is given by the field of view of the user. Note, that only the structure of the network is given as pre-knowledge in the memory, but the network parameters are learned from the training video sequences.



(a) Memory content fits the model quite well.



(b) Many conflicts indicate that the hypotheses do not fit the expectation.

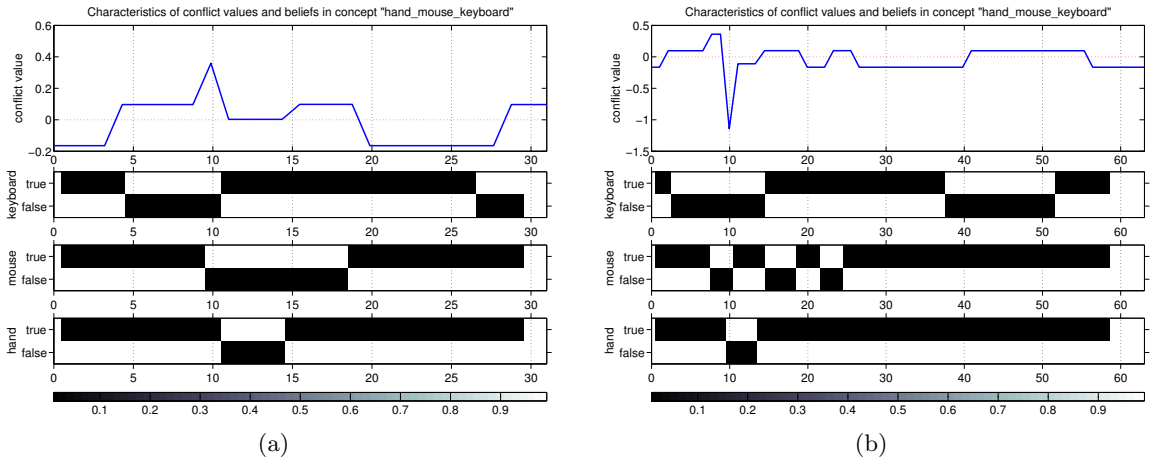
Figure 8.10: Results for the FDC "monitor_keyboard".

In this example, the parameters have been estimated from about one minute annotated video data taken while looking around in two different offices. The memory process for the consistency validation ran at a rate of $\mathbf{1}\boldsymbol{Hz}$ in these studies. In figure 8.10 results are displayed for two different annotated sequences not being part of the training sequence. Each diagram displays the *conf* value on the top and below the posteriori believe computed by the propagation of the underlying Bayesian network. The left plot shows a sequence which fits the model quite well, while on the right we find results for another test sequence with many conflicts detected. Obviously, the system has learned that there is a correlation between the existence of a monitor and a keyboard. This correlation is that strong that the system detects a conflict if only one of these two is recognized, which is the case at timestamps 2 and 11 in the left figure 8.10(a) with a positive conflict value of $conf = \mathbf{0.6}$. Obviously, the system has learned that the concepts keyboard and monitor belong together and usually can be found in the same spatial context. For the results in the right figure 8.10(b) we took a sequence which does not fit the model very well. The in average higher conflict value indicates that the learned concept is not suitable for all office scenes. The data may be exceptionally or the model is probably not generic enough which is presumably due to the very simple network structure with only two nodes. A learned correlation does not need to be typical for all contexts.

A Model relating Hand, Mouse, and Keyboard

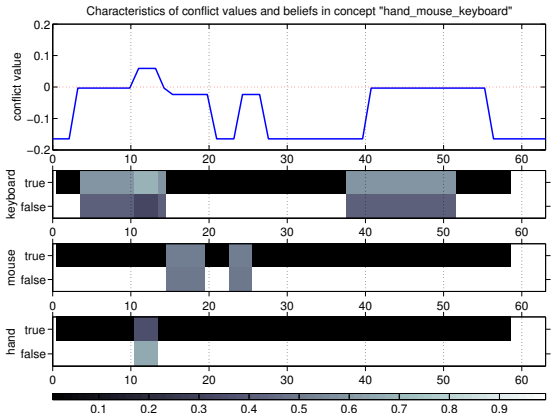
Another FDC represented by its network structure is shown in figure 8.9(b) with the tables of the learned parameters displayed at the respective nodes. This concept learned that the visibility of a hand in the field of view is correlated to the existence of a keyboard or mouse in the scene. Both test sequences displayed in figure 8.11 indicate quite high conflict potential according to this model. This is explainable since the model was trained from sequences which contained mainly typing actions, where the user was sitting in front of computer, while the two test sequences contain data taken from just looking around and touching several objects in the office. Thus, the context is a different one and the model detects many conflicts.

Let us now have a look at the effect of reducing the reliability of the participating hy-



(a)

(b)



(c) Conflict validation with reliability factor of 0.5 for all existing hypotheses (compare to figure 8.11(b))

Figure 8.11: Results for the FDC "hand_mouse_keyboard".

potheses, as all examples up to now have been computed with the reliability set to **1**. As explained in Sec.6.5 the use of soft evidences allows to incorporate the reliability into the computation of the conflict value. Accordingly, the evidences assigned to the node are now soft evidences. For the results in figure 8.11(c) we assumed a reliability of **0.5** instead of **1.0** for the hypotheses in the memory. As a consequence, the beliefs shown in the sequence plot are not only black and white, but show the computed a posteriori belief for nodes when a hypothesis was existing in the memory at a given time. The conflict value is notedly reduced in contrast to the one in figure 8.11(b). The slightly different shape of the curves is explained by the asynchronous communication of the memory process with the active memory.

Discussion

The results so far show that the implementation of an inner memory process for consistency validation that relates episodic instances to each other works well and allows to detect conflicting hypotheses. It has been shown, that Bayesian networks are capable to model world assumptions and expectations regarding specific observations. Although only very simple models have been evaluated so far, more complex ones have already been designed and also proved appropriate. However, the network structure has always been manually defined, but

the parameters have been estimated from real world data. Methods for structure learning of the Bayesian network have not been exploited yet, because the particular structure turned out to be quite irrelevant for the considered cases. But structure learning methods are available [111] and can be applied easily, especially for only small numbers of variables as presented in this evaluation study.

But it is still an open issue, how the proposed means to solve conflicts by decreasing the reliability value of involved memory elements works on a larger scale. Several different memory processes independently modifying the reliability values of different memory elements may lead to interference and, in the worst case, to chaotic behaviors. In consequence, the means currently applied to solve conflicts works well for small, non-interrelated FDCs, but needs further investigation for more complex scenarios.

8.3 The Integrated System

After having discussed the evaluation of some relevant components individually, we will now turn back to the integrated system. In the introduction to this chapter, I set out the goal to evaluate whether the proposed system architecture is appropriate for the envisioned cognitive assistance systems. But this is a rather difficult issue, as we have to face a so-called "empirical gap" [72]. Different components can be assessed individually in a qualitative and quantitative manner, but how they work together in the system and whether their combination is better than any alternative cannot easily be proved. The empirical gap opens between the assessment of the individual components, which can in fact most often by evaluated in an empirical way, and the empirical analysis of the system's overall functionality and usefulness with respect to a given task. In between, any inference regarding the effects of a specific combination or the integration approach usually cannot be drawn. An evaluation by systematically varying the composition of the system and assess their respective appropriateness is not feasible, due to the combinatorial explosion caused by the huge number of degrees of freedom.

Alternatively, different approaches can also be evaluated by comparing them to each other. Obviously, systems can only be comparatively evaluated if different approaches exist that all try to tackle a common challenge. In the recent past, computer scientists set out more and more in establishing such scenarios. An example for such a benchmarking scenario is the so-called RoboCup [117], in which robot teams of different sizes and appearance compete in soccer-like games. It has become rather famous to study and assess approaches in robot mechatronics, collaborative team behaviors, and intelligent robot architecture to mention only a few. Of course, a system can be rated in comparison to others by let the system compete against each other.

Another scenario to evaluate complex integrated computer systems composed of many different components and functionalities is the "DARPA Urban Grand Challenge" [43] which recently gained famousness. The envisioned goal set out for the current challenge is an autonomous vehicle that can cover a distance of approximately 100 km from a defined start point to an end point in less than six hours completely autonomously using different sensors to perceive the environment. In 2005, this challenge has been solved for the first time by Thrun et al. [155]. Unfortunately, no such established scenario does exist for cognitive assistance systems, impeding the direct, comparative assessment of the proposed integrated system.

Thus, other ways of evaluation need to be found in order to prove the appropriateness of the presented approach. But how can this question be answered then? I claim a rather simple, but still extensive benchmark here:

The system must work and have a purpose!

Of course, proving this statement for a given system does not necessarily allow to rate how good it works compared to others, and does not preclude that any other better solution might exist. But it allows to prove the general appropriateness, and also to gather insights about specific problems and drawbacks that need further investigation.

The evidence that the system actually works is provided by the fact that the system has been presented running live many times on exhibitions, scientific workshops, lab demonstrations, and project reviews. Having various unexperienced people successfully working with the system in different environments with varying conditions can be considered as an evidence for its wider range of applicability. Impressions from two different scenarios

(a) A prototype shown at the IST (Information Society Technology, a research priority of the European commission) event in The Hague, Netherlands in 2004.

(b) At the VAMPIRE industry workshop 2005 in Pommersfelden, Germany.

Figure 8.12: Impressions from different live demonstrations of the integrated system at different occasions.

in which the system has been used are shown in Fig. 8.12. These pictures also show the system distributed over several laptops in different environments. Altogether, the VAMPIRE assistance system has approximately been used by more than 30 different people and has been demonstrated to many more. Although this is not a prove of the systems in terms of systematic evaluation, these demonstration still constitute one of the most valuable sources of insights regarding further improvements and development of the system. And they give evidence, that the system indeed does *work*.

But in order to evaluate our system more systematically, I will discuss some aspects that guide the evaluation in the following:

(i) **Functionality** is concerned with the question, whether relevant functions are available to solve a given task and whether they work appropriately. Conducting a series of studies with different users will unveil, which functionalities are necessary, appreciated, or dispensable. But functionality analysis is also concerned with *precision* and *robustness* of the involved approaches. It will be assessed how well a given functionality works in the given scenarios. This analysis is therefore task-dependent.

(ii) **Reactivity** is covering speed issues, response times, and so on. As we are facing issues of an online ego-vision system, it has to be analyzed how the performance of the system is with respect to computational speed and which effects it has on the quality of assistance provided by the system.

(iii) **Usability**: The aim of usability studies is "The gathering of information within a specified context on the usability or potential usability of an interface, and the use of that information either to improve features within the interface and the supporting material or to assess the completed interface" [127]. Hence, usability studies are mainly considered with user interface, interaction capabilities, and comfort from a user's perspective. But it furthermore covers also aspects of the hardware setup.

In an integrated system, the precision and quality of the individual components will ob-

viously affect the overall quality of the system. If the quality of the overall system would be determined by the independent quality of the individual components, this would have dramatical consequences. If any error in the computation of one of the components would cause an error in the whole system that cannot be compensated, the quality would indeed be very low, because the success rates of components are considered to be independent of each other in this case. In consequence, even if every component would exhibit a rather optimistic success rate of $\boldsymbol{P_i = 90\%}$, the quality of an integrated system of only $\boldsymbol{n = 5}$ independently working components would statistically drop to $\boldsymbol{(P_i)^n \approx 59\%}$. Such a system would be almost useless. Although this is a theoretical consideration, it is obvious that redundancy and error recoverability are crucial for integrated system.

In order to account for these issues, coping with errors and uncertainty is directly regarded in the design of the system by means of the ego-vision paradigm and the hypotheses concept. To give an example, the user can trigger re-training of objects if she or he has noticed that a specified object is not accurately recognized under the given conditions. Furthermore, redundancy and alternative solutions to reach designated goals are implemented. The mouse, for instance, has been presented as "fallback" input modality, accompanying speech input.

From these considerations we drew the consequence, that an overall evaluation must necessarily include the user as she or he is an inherent part of the process. This holds especially for ego-vision systems with their close coupling between user and system. Accordingly, we conducted user studies for a final and integral evaluation of the system. The study is expected to yield insights regarding as well the selection and composition of components, the proposed system architecture, and the system integration using the active memory infrastructure. Therefore, the studies have been designed and conducted in close collaboration with Sebastian Wrede, who conceived the integration framework. Results of these studies have also been reported in a joint publication [163].

In order to still gain insights regarding the proposed design of the system, the studies have to be carefully designed to draw relevant conclusions. In the following section, I will outline, how these studies have been conducted and will present results.

8.4 User Studies

The user studies have two major goals. First, they should assess the system architecture for its appropriateness. Therefore, we reconsider the four aspects outlined in the last section. And second, they should provide insights about the cooperation of the user and the system by means of the ego-vision paradigm. In cooperation with researches that have experiences in evaluating integrated systems by means of user studies [12, 135, 83], we designed a study that should shed some light with respect to the different foci of evaluation outlined before.

8.4.1 Experimental Setup

The experiments have been carried out in the bartender scenario that has been presented in Sec. 8.1. Fig. 8.13 shows the basic setup for the studies. A "cheese" target is positioned in front of the user to compute the 3D pose. As interaction devices a microphone and a wireless mouse are used as already proposed in chapter 4. Head gestures as presented in Sec. 4.3.2 are not applied as interaction modality in this user study to reduce the complexity, and due to their limited semantics. Almost all interaction goals can be achieved via speech

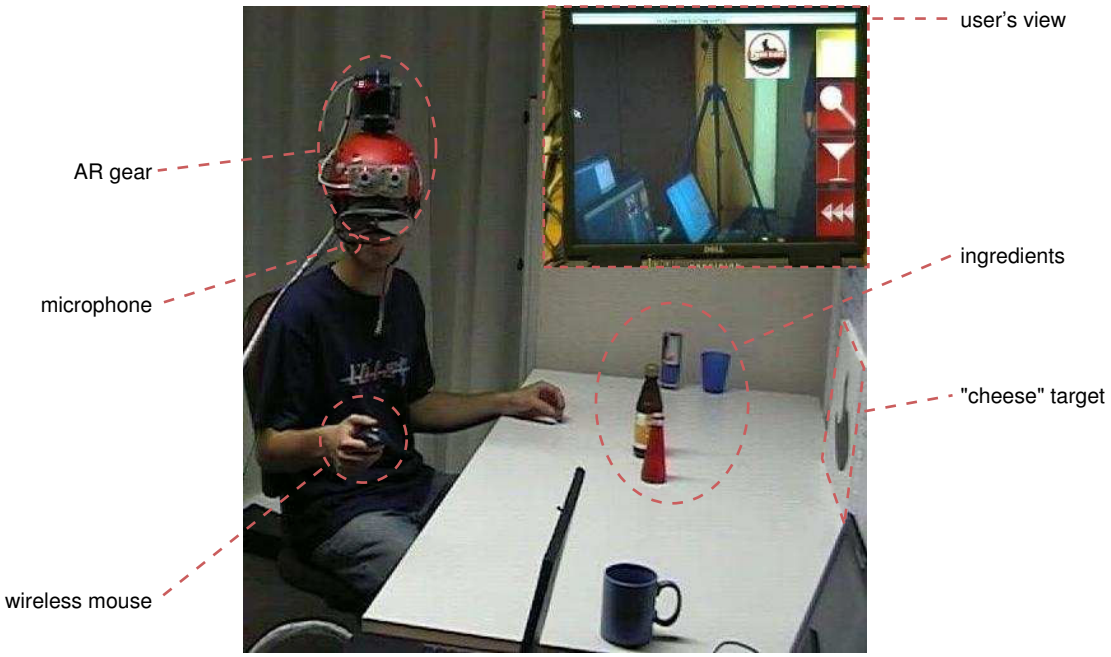


Figure 8.13: Experimental setup for the user studies: Subjects have been recorded during the evaluation sessions from this perspective for analysis.

input or mouse interaction in the given scenario.

The system architecture that is deployed in the studies has been sketched in Fig. 7.2. The components sketched with a dashed line are not part of the "core" system, that is evaluated here. In order to be able to conduct user studies with reasonable effort and to draw meaningful conclusions from studies, the system must not exceed a certain complexity. Therefore the automatic decomposition into planes and the generation of mosaics has been left out in favor of a pre-modeled view context in terms of the table plane.

For the trials in the user studies, we designed specific tasks to be accomplished by the subjects. The story board for the studies basically defines two phases. Each subject was given some time to adjust and familiarize oneself with the augmented reality setup before entering these two phases.

(i) **Teaching Phase** In this first phase, the subject should teach some objects to the system. Initially, we clear the system's memory before each trial. The user enters the "teaching" mode interactively by controlling the graphical user interface by either speech, saying "Let me teach you some objects", or by using the wireless mouse. This phase is explicitly explained to the subject before the experiment. In order to ensure the same amount of instruction for every participant in the study, a specially prepared instruction video is shown to every subject in advance. In the learning mode, the system gives visual feedback about the different objects it can perceive, but not yet classify as already shown in Fig. 4.3(a). The user should now interactively teach the objects by about 4-5 views of each object and labeling it by speech (saying, e.g., "This is champagne") after be requested by the system. The object learning is done in background, and the system informs the user, when this learning has finished (usually in less than one minute). Immediately afterwards, the system is able to recognize these objects and memorize their 3D positions on the table. This procedure is repeated for

all ingredients on the table.

After this teaching phase has been left, the system is able to recognize objects on the table and to determine and memorize their 3D position in the scene. The user receives permanent feedback from the system regarding recognized objects, so that the user can in this phase re-enter the "teaching" mode to update the system's object representations interactively by adding more views and re-initiate the object training.

(ii) **Assistance Phase** In the second phase, the system is applied as an assistance system. In contrast to the first phase, this phase is not explicitly introduced to the subjects by an instruction video. They are only told, what the system should be good for, namely assisting in mixing beverages, but the concrete interaction is not demonstrated in advance. By this means, we assess how self-explanatory the system is. Recipes have been pre-modeled and have been composed of two ingredients each, but were not known to the participants in advance. Hence, they have to "blindly" follow the instructions provided by the assistance system in order to complete the given task. In the assistance phase, the user is not only instructed in terms of the next step that has to be accomplished, e.g., adding a shot of juice to the cocktail, but furthermore is guided to the previously memorized position of the object. This guidance to the next ingredient by means of arrows has been presented in Fig. 4.3(c). To investigate familiarization effects, this second phase has been carried out twice with different recipes.

8.4.2 Evaluation Methods

Concerning the evaluation a combination of three methodological approaches as proposed by Dix et al. [47] appears promising in capturing the evaluation goals. They proposes the following axes for evaluating systems with a focus on human computer interaction:

Experimental evaluation: Experimental conditions will be set up in a framework of controlled variables. Differences in humans' behavior will be assigned to the differences in conditions. This method allows to gather information about specific questions and to test hypotheses.

Evaluation via observation: The users are observed or recorded during their use of the system. This approach allows to gather information about what they do when they interact and how they do it. This information can later be used in a task- and performance-oriented analysis. One can gather insights from this study in terms of reasons for problems, identification of error-prone aspects in the usage, and serious interaction problems. These insights provide most valuable input for further improvements.

Evaluation via questionnaire: Applying this method, written or oral interviews are conducted with the users after they take part in the experiment. The goal of this approach is to find out more about the users' subjective opinions about the system and the interface, and directly asses the specific functionalities from an user's perspective.

In our study we focused on the latter two methods, as the first is only suited to give answers to very specific questions, which have not been the goal of this specific evaluation. The systematic variation of specific variables in the system leads to an combinatorial explosion as soon as a higher number of variables is examined. A basis for the selection of interesting

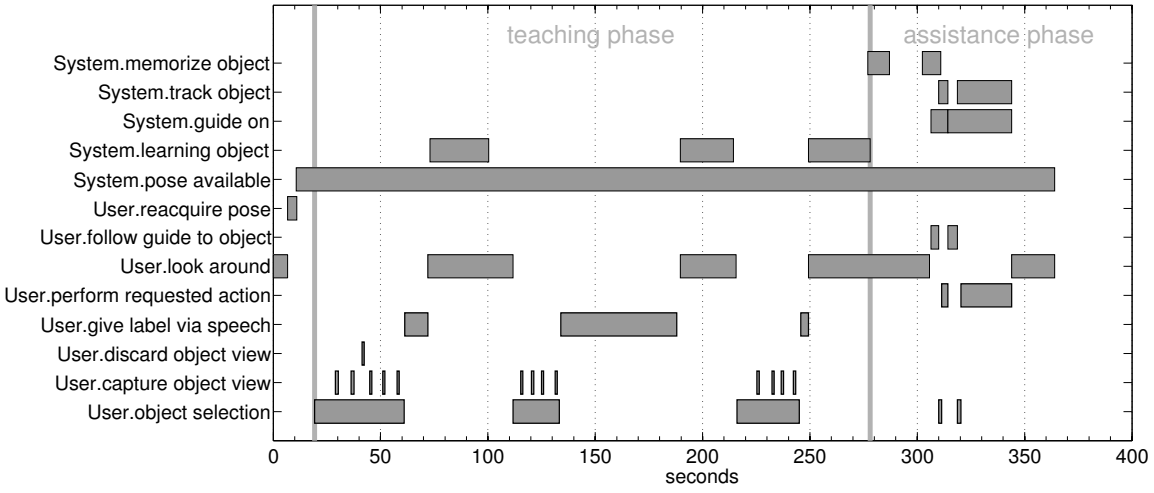


Figure 8.14: Annotation of user actions and system activities taken from one of the user study sessions conducted for evaluation.

parameters to vary was not given before this study, but however can be established with the results of this study at hand.

During the experiments all user interaction has been recorded by video cameras for later analysis following the "evaluation by observation" method. Afterwards, the subjects were asked to fill out a questionnaire yielding an assessment on a qualitative scale and also individual comments regarding the presented system. The questionnaire is attached to this thesis in appendix B. As the study has been conducted with German participants, the questionnaire is in German, too. Appendix C contains the English translation of the respective questions.

The quality is measured in discrete steps ranging from 1 (usually the best achievable result) to 5. The free-text fields are useful not only to evaluate the quality of the current system but also provide beneficial hints and suggestions for the development of future applications.

Given a specific task, some basic measures can be collected in the study giving evidence regarding the goals of the evaluation. In the study, we can ascertain

- ▷ a binary value, indicating whether the task could be *successfully* accomplished or not,
- ▷ the amount of *time* the completion of the task took, and
- ▷ repeating *error patterns* especially from the video recordings indicating problematic functionalities.

In order to obtain these figures and to facilitate evaluation by observation, sessions can manually be annotated. We utilized ANVIL [84], an annotation tool for multi-modal interaction, to annotate system and user activities in the course of time. In order to analyze how much time is spent on which part of the tasks such annotations, as exemplarily shown in Fig. 8.14, can provide a valuable and compact representation. By means of this analysis, typical patterns can be identified. In the given example, the user trained three different ingredients to the system, while he had apparently a problem labeling the second one, as this phase took more than a minute. Due to problems in speech understanding the user

only managed this interaction task after many unsuccessful attempted. It is also apparent, that in this experiment the pose tracking succeeded all the time after the initialization at the very beginning of the experiment.

8.4.3 Results

I will now discuss the results with respect to the evaluation aspects outlined before utilizing the presented methods. The study was conducted with eleven subjects. All participants in the studies used the VAMPIRE system for the first time, although most of them have a background in computer science and are students at Bielefeld University. All of them were able to solve the task given. However, the performance was quite diverse. Time consumption ranged between 15 minutes and 90 minutes. These two figures can be considered as outlier. For most subjects the whole experiment including instruction, familiarization with the device, and video presentation took about 30 minutes.

The answers for the quantitative measures obtained from the questionnaires have been transformed into pie charts, which are shown in the figure 8.15 to 8.18. The free form comments of the participants have been collected and manually clustered to draw meaningful conclusions from these.

Functionality

Most users claimed that the system fulfills their expectations they had to the most extend as indicated by the chart in Fig. 8.15(f). The overall collaboration was moreover rated rather good (Fig. 8.16(b)) by all participants.

Taking a closer look at the specific functionalities of the system it unveils, that object marking has been rated positively in majority (Fig. 8.17(f) and 8.17(b)). This assessment concerns not only the object recognition component, which obviously classifies the objects quite reliably, it also should be seen with respect to the complete visualization loop that has been detailed in Sec. 7.3.1. The interplay between "3D context analysis" and "hypotheses anchoring" avoided to bother the user with false positives and wrong classifications that nevertheless have been produced by the object recognition especially when the users looked around. So it confirms the appropriateness of our concept of fusing and rating hypotheses in the active memory as this allows to neglect unreliable information and thus reduces the mental load of the user to some extend. Wrong object labels would have annihilated the usefulness of such markings.

The guide functionality which comprised the guidance of the user to the next ingredient by means of errors was not noticed by most of the test persons, as we learned from the subject's comments. We assume that this is due to the fact that the scenario implicated no need for this specific functionality, since the position of the small number of ingredients was memorisable by the subjects easily. As the guiding functionality is part of the assistance phase, it has not explicitly been explained to the subjects in the study, and its intention seemed not be implicitly apparent. As a consequence, results regarding this functionality presented in Fig. 8.17(c) and 8.17(d) have not been encouraging yet.

We have not asked users concerning the quality of action recognition explicitly, but many different observation could be made in the study with respect to this functionality. Most remarkable is the rather high rate of false negative action detection which is significantly higher than reported for the evaluation of the individual action recognition component

presented before. This seems to be due to the fact, that the threshold $\boldsymbol{p_{thres}}$ and also the other parameters reported in Sec. 3.4.4 have not optimally been chosen and affect the recognition quality. The parameterization for the action recognition has been obtained from studies under rather controlled conditions as reported in Sec. 8.2.2. Obviously, the aberrations from the trained models are significantly larger in the integrated online system. False positives, on the other hand, have only rarely been observed. As a consequence of this unbalance, subjects occasionally had to perform the same action several times, before the system detected it correctly. Nevertheless, the system was finally able to recognize the actions of all subjects.

Another component that has been identified as being critical for the functionality of the whole system is "pose recognition". As outlined in Sec. 7.2.4, many other functionalities are dependent on a correctly determined pose. Therefore, a special feedback mechanism has been implemented prompting the user to re-focus the target (cf. Fig. 7.3). Although all users understood this prompting correctly, re-initializing the pose tracking was not always successful in a first attempt. It could be observed in many trials, not only conducted in the course of this study, that many problems in the functioning of the whole system arise from problems in pose tracking.

Further comments and answers indicated that the textual instructions (Fig. 8.18(d)) and step-wise guidance in preparing the beverages were widely appreciated and understood by the subjects. Subjective feelings about the system's learning and memorization capabilities have been reported as rather good as indicated by the charts in Fig. 8.18(a), 8.18(e), and 8.18(f).

Reactivity

Reactivity is mostly an issue of functionalities that require real-time visualization. In particular, these are object annotation and the visualization of the tracking process. The object markings have been experienced as being accurate (see Fig. 8.17(a)). This is an evidence that the processing path sketched in Sec. 7.3.1 is reactive enough for the user to work smoothly with the system.

But for the visualization of the tracking process, we observed delays that had effects on the users. Although, the data exchange in this case is surpassing the memory and directly fed into the visualization component as shown in the architectural sketch in Fig. 7.2, delays could not be evaded here. As the object movement is usually quite fast when performing actions, even short delays of less than $\mathbf{100}\boldsymbol{ms}$ in the processing path are notable, and furthermore have an effect on the accomplishment of the tasks by the user. It could be observed, that due to the delayed display of the tracking visualization, users slowed down their own movements. Here, the close coupling between system and user by means of ego-vision becomes apparent. This reduced execution speed of the actions is also one reason for the reduced recognition capabilities in action recognition as reported above. The models have been acquired on the basis of a regular execution speed and can only cope with variations in the applied limits (cf. Sec. 3.4). This impact on their movement has also been experienced by the users themselves as reported in chart 8.16(c).

Usability

The overall system usability is rated fairly well as depicted in Fig. 8.16(d) with nobody judging the system to be difficult to use. When considering the input modalities it becomes

apparent, that especially the usage of the mouse wheel for navigating the GUI is convenient to most of the users (cf. 8.18(b)). The familiarity of the users with mouse interaction itself and the reliable functionality of this modality have been reported as main reasons for this assessment. In contrast, speech is not favored as input modality by most subjects. As all of the subjects were non-native English speakers, the results for speech recognition quality vary greatly as depicted in Fig. 8.18(c), probably because our speech recognizer has been trained on a native American-English Wall-Street Journal corpus. It could be observed, that subjects had to repeat their verbal utterances quite often to achieve their goal. Here, poor recognition performance due to the above reasons, let to the low acceptance of this input modality. Furthermore, the predefined lexicon proved to be too restrictive for an intuitive interaction via speech.

Following the idea of ego-vision systems integrating the human into the loop, subjects were asked about the transparency of the system's states (Fig. 8.17(e)) and the quality of feedback given by the system (Fig. 8.16(a)). Both have been rated positively. The operational course of system handling was mostly comprehensible (cf. Fig. 8.15(e)) for the subjects. Nevertheless, a demand for further external explanation exists (Fig. 8.16(f)). Obviously, self-explanatory is an aspect that can further be improved, although the design of the GUI is rated as very self-explaining already (Fig. 8.15(b).

Furthermore, it turned out that subjects performed much better when accomplishing the task a second time in slightly modified matter. The subjects had the overall feeling of a short familiarization time (Fig. 8.15(d)) and a steep learning curve when working with the system. This impression has been supported by the finding that more than two third experienced the task much easier in their second run (cf. Fig. 8.16(e)) and also consumed significantly less time.

In the statements of the subjects and as well in the ratings (see Fig. 8.15(c)) the hardware setup of the AR gear showed up as an aspect demanding for further improvements, although the subjects had some familiarization time before really starting with the given tasks. As the AR gear is a research prototype, it is rather bulky and heavy, and therefore also affects the movement of the users.

8.5 Insights and Discussion

The assessment of the individual components presented in Sec. 8.2 proved their fundamental appropriateness for ego-vision systems, and their performance has already been discussed in the respective sections. Now, insights gathered from user studies and from the performance of the integrated system as a whole shall be discussed.

From a technical point of view it can be stated that the possibility to conduct studies with unexperienced users already is a most important positive assessment of the system architecture. The integrated Vampire assistance system allowed the subjects to use it reliably in several trials. Furthermore, numerous successful live demonstrations back that the system is basically working.

The conducted user studies unveiled the strengths and drawbacks of the implemented system with respect to the given task. They have provided valuable feedback as well for the system design as for ergonomic aspects. In summary, it can be considered a success that all participants of the study were able to complete the demanded tasks and found the assistance system quite helpful. From these findings, it can be concluded so far, that the system actually is working and that it has a purpose in accordance with the evaluation goals

set out in the last section. Partly, some features of the system needed further explanation, so that users became aware of them. However, most of the users were able to interpret any system failures correctly and to act accordingly to overcome them. By means of process feedback the user is motivated to never give up, as she or he presumably feels as an inherent part of the system.

It is also a general confirmation of the proposed concepts that all subjects reported that they enjoyed using the system, though some drawbacks with respect to usability and comfort had to be accepted. This positive assessment basically approves the general idea to develop assistance systems that take an ego-vision perspective and integrate the user into the processing loop. Especially the close coupling using shared perception and augmented reality provided benefits for the collaboration in terms of feedback and error recoverability.

The studies generally proved that the proposed system is reactive enough to realize an interactive loop by means of augmented reality. Still, the time-critical visualization loop can cause short delays and in consequence hamper smooth and completely natural behavior of the user. The effects seen for the object tracking feedback in action recognition gave valuable hints to this conclusion. Some short delays between perception and visualization can never be avoided for fast movements in video see-through augmented reality setups. It seems to be more promising to find other ways of feedback for suchlike use-cases, instead of trying to keep track in real-time.

The hardware setup of the AR gear turned out to be another aspect for further improvements. It is too bulky to be applied in everyday scenarios and longer runs. Alternatives for future developments in augmented reality are nowadays already in sight with much smaller hardware. Furthermore, transparent displays that do not impede the visual perception of the user have improved a lot in recent years and allow less intrusive augmented reality setups.

The proposed system architecture is confirmed by several findings of the studies. First, the proposed interplay of inner and outer memory processes could be proved by the successful evaluation of the object annotations in the user's field of view. This functionality is the result of the interplay of many different components as presented in Sec. 7.3.1. Also, the other exemplified processing path (Sec. 7.3.2) that describes how object tracking is triggered by centrally focusing an object, proved to work reliably and was accepted by the users.

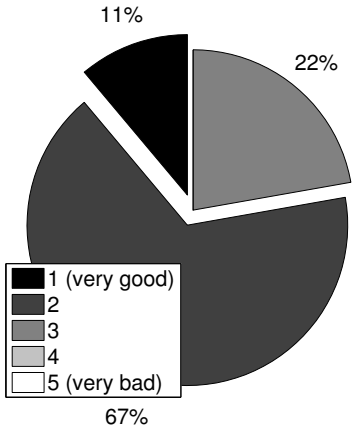
The four-layered design of the visual active memory turned out to be appropriate for the organization of the memory content and allowed a structured processing. Percepts have been inserted into the memory and anchored to reliable episodic instances. These stayed in the memory and got updated by the hypothesis anchoring process as long as matching percepts arrived in the perceptual memory. This is proved by successful evaluation of the object annotations again.

For object learning, users interactively inserted selective image patches of objects into the pictorial memory in order to initiate an asynchronous learning of this object. The resulting object representation could be stored in the conceptual memory of the VAM. The learning capabilities of the system have also been positively noticed.

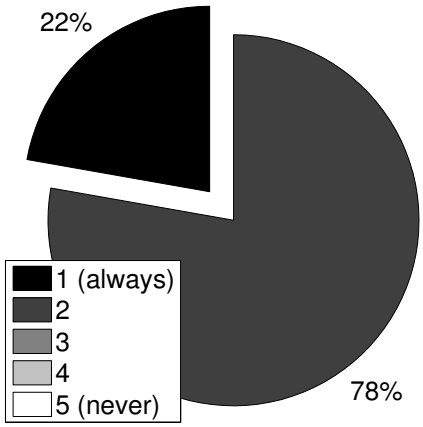
Finally, the flexibility of the system architecture becomes directly apparent, when we discuss how an insight obtained from the user studies has quickly been turned into an improvement of the system. As described before, the performance of the overall system sometimes suffered from errors in pose tracking. The "context-aware visualization" has subscribed to the active memory for object hypotheses that have a certain reliability value. As the reliability of object percepts is reduced by the "3D context analysis" when no pose is available, the

objects without 3D information did not get displayed anymore. In order to overcome this problem, only the subscription of the visualization component had to be modified, now allowing also memory elements with lower reliability to trigger the respective functionality. In consequence, also objects that are less reliably recognized by the system are displayed. But for the given scenario this turned out to be less perturbing to users than the complete absence of any annotations, although some are potentially incorrectly displayed. The information-oriented representation and the hypotheses concept in conjunction with the event-driven integration of the integration broker architecture allowed this easy adaptation of the system's behavior. This adjustment in consequence relaxed the coupling between "pose tracking" and "context-aware visualization" implicitly, as now the 3D position of objects does not affect the visualization anymore.

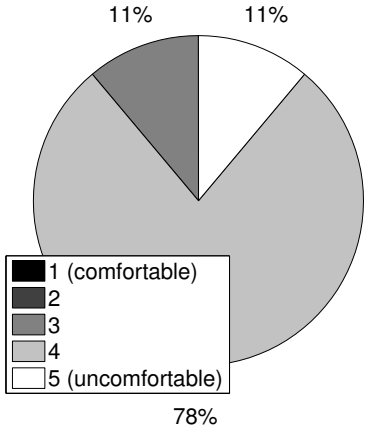
Besides these specific considerations, general attributes like collaboration with the system (Fig. 8.16(b)), usability (Fig. 8.16(d)), and overall learning capabilities (Fig. 8.18(e)) have been assessed positively in the user studies. Using the system was reported as being joyful and interesting by all subjects. Thus, the realized assistance system not only serves as a feasibility study of the concepts outlined in this thesis, but also allows to conduct further research in terms of new innovative cues in human-computer interaction in general and in assistance systems in particular.



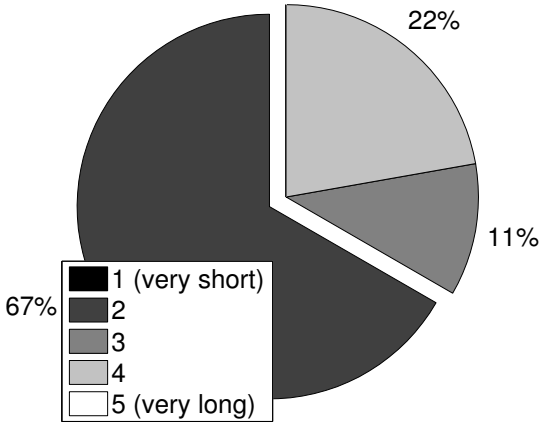
(a) User's orientation on the display



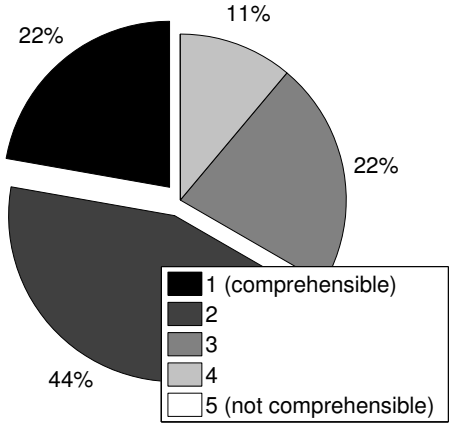
(b) Icons self-explaining



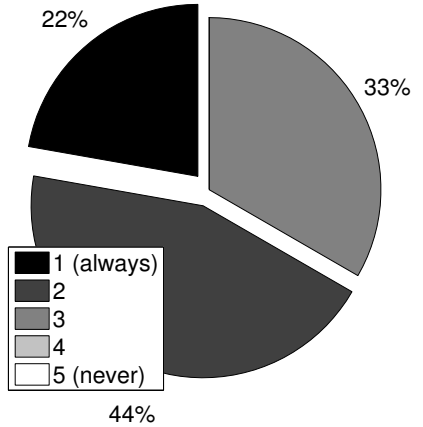
(c) Wearing of helmet



(d) Familiarization with system usage

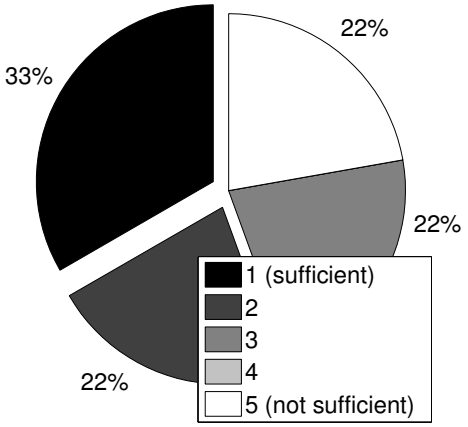


(e) Operational transparency

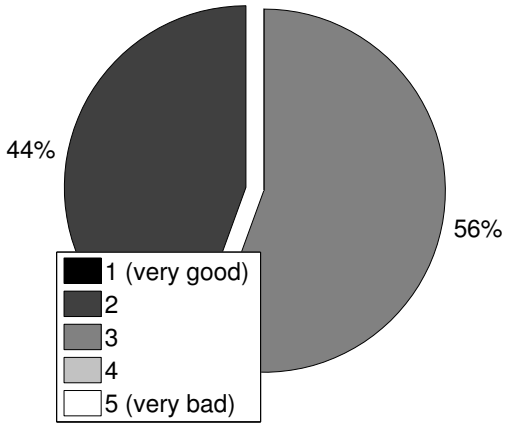


(f) Fulfils expectations

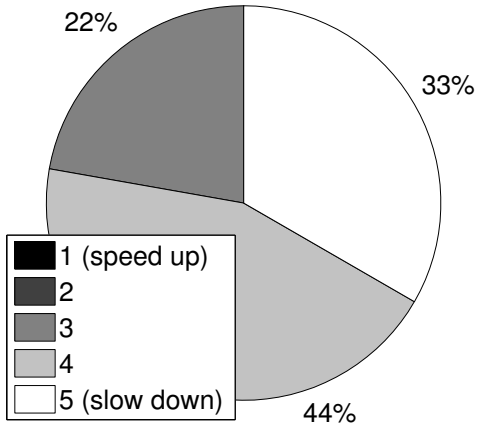
Figure 8.15: Results of questionnaire (Part 1).



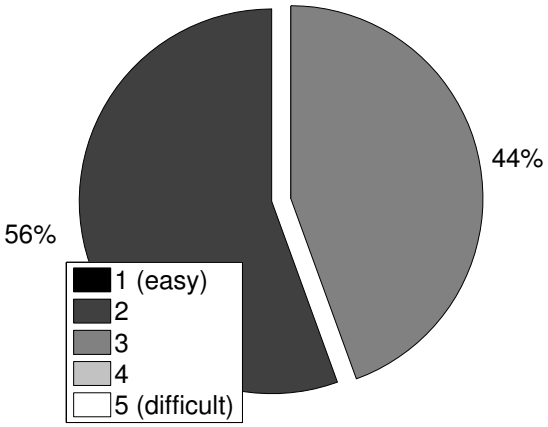
(a) Feedback during task execution



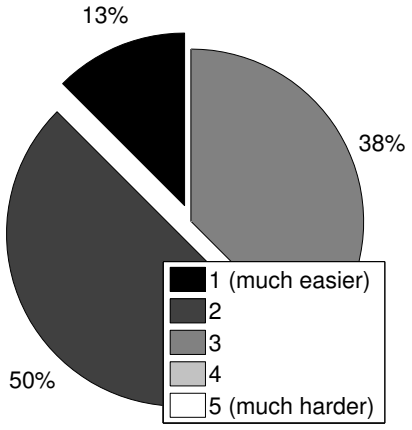
(b) Collaboration with system



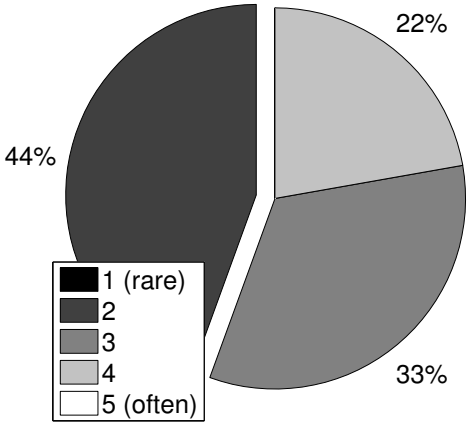
(c) Impact on user's movements



(d) System's usability

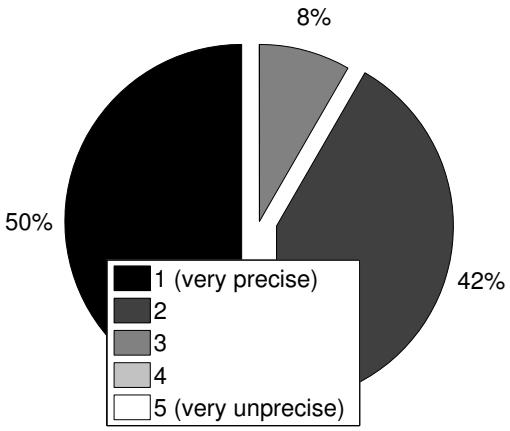


(e) Execution of second task

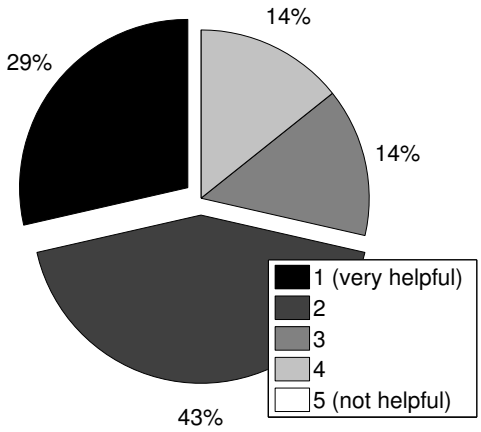


(f) Need of external assistance

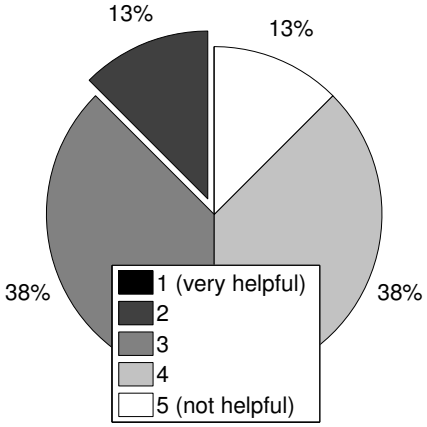
Figure 8.16: Results of questionnaire (Part 2).



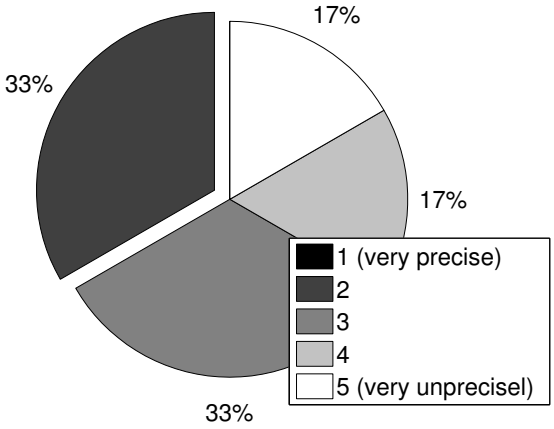
(a) Precision of object marking



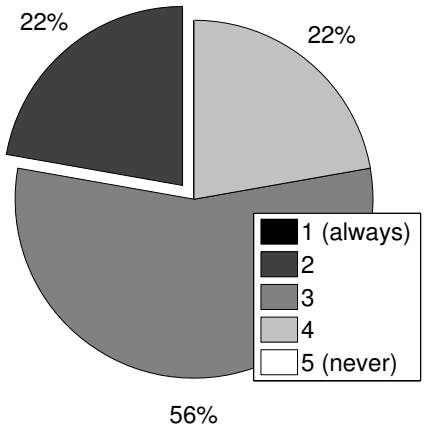
(b) Helpfulness of object marking



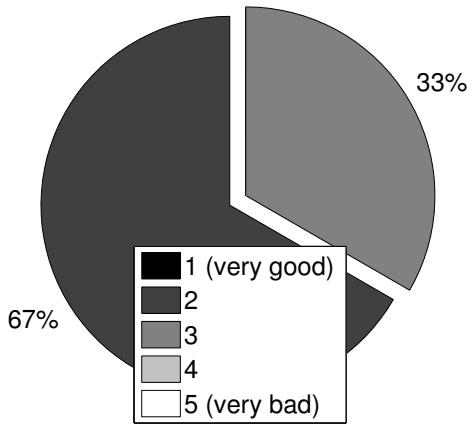
(c) Helpfulness of guide functionality



(d) Precision of guide functionality

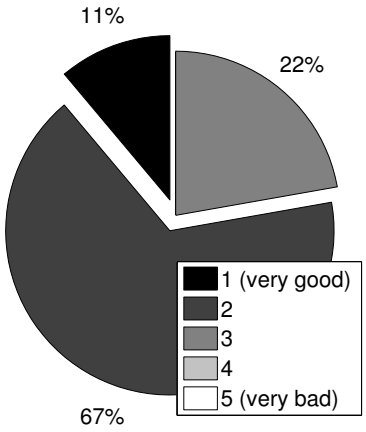


(e) Transparency of system state

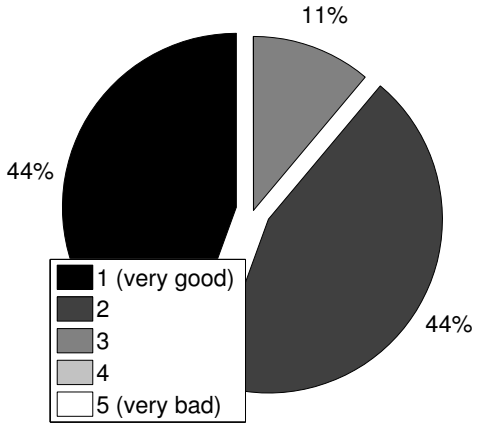


(f) Assessment of object recognition

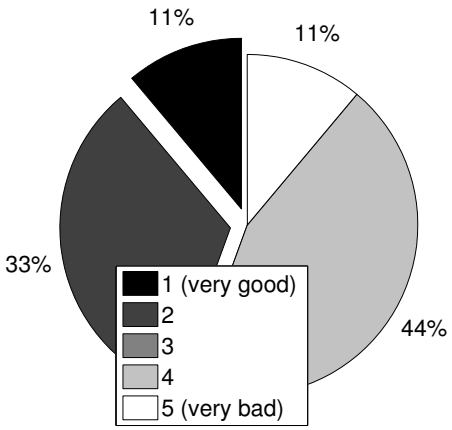
Figure 8.17: Results of questionnaire (Part 3).



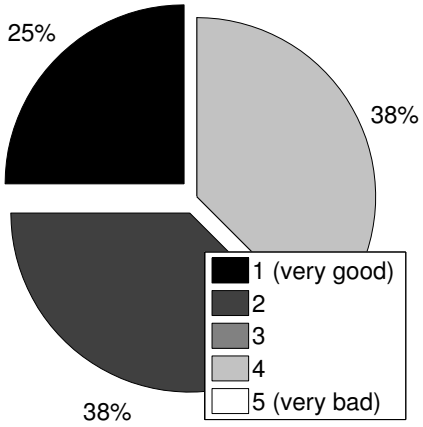
(a) System's internal learning process



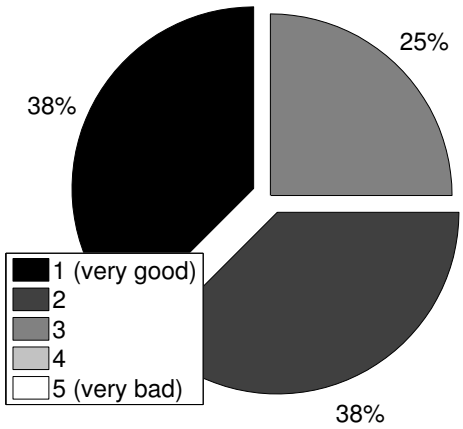
(b) Interaction with mouse



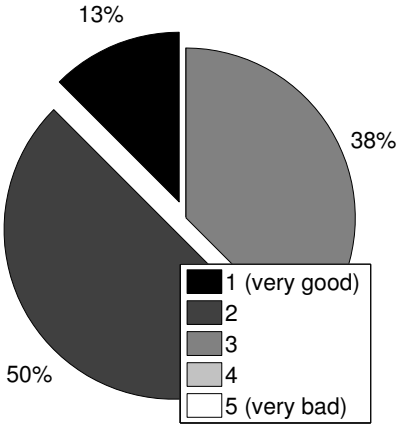
(c) Interaction via speech



(d) Textual instructions



(e) Ability of the system to learn



(f) Memory function of system

Figure 8.18: Results of questionnaire (Part 4).

9 Conclusion

Scientists build to learn; Engineers learn to build.
[Frederick P. Brooks, *1931]

Computer systems that can assist humans in their every day routine or work tasks are a long term goal of research in computer science. In order to do so, they need to develop capabilities that enable them to *perceive* and *interpret* situations, they need to *articulate* their knowledge, and have to understand and *interact* with the human user. The most valuable and flexible assistant a human nowadays can have is still another human. Hence, it is a rather natural consequence to investigate systems, that develop at least partial capabilities of human cognition. Vision is generally considered to be the most important modality for humans to perceive the world. Consequently, this work focused on vision as a primary cue for perception and interaction, too.

9.1 Summary

Starting from these basic motivations, this work set out to discuss assistance systems on the one hand as rather generic computer vision systems, that need to perceive and understand their environment and act accordingly. On the other hand, I focused on a more specific class of interactive systems, that are tightly coupled to the user. These systems have been presented under the term "*ego-vision systems (EVS)*". This novel term has been introduced as a paradigm for systems that integrate the user into the visual processing loop, usually by means of head-mounted cameras and augmented reality visualization. As a consequence, both, system and user can mutually control their perception and additionally always share the same view of the environment. This paradigm especially facilitates close collaboration and has therefore proved to be appropriate for assistance applications in particular. Ego-vision provides easy articulation of attention and enables efficient interaction strategies by context-aware prompting, to mention only a few benefits of this paradigm.

Perception Besides these benefits, ego-vision also raises several challenges to be faced when developing such systems. I have identified these challenges at the beginning of this work in chapter 2. For the perception of EVS, these primarily result from the fact that the cameras' movement cannot be directly controlled by the system. The system can only perceive the scene from head-mounted cameras, which are under inherent control of the user. Several solutions for the challenges have been proposed in the course of this study, mainly in chapter 3.

With the mosaicing of planar sub-scenes (Sec. 3.1), the limited field of view of ego-vision systems has been extended and a means to establish a spatial context on a pictorial level has been introduced. Generally, "context" played a rather crucial role in this work, as almost no

perception can be considered without its spatial, functional, or temporal context. Actions, for instance, have been recognized on the basis of the movement of a manipulated object and the contextual information about the class of this object.

As tasks are most often concerned with the manipulation of objects, their detection and classification generally play a crucial role in perceiving the environment. Two different approaches for object recognition have been discussed with respect to their appropriateness for ego-vision (Sec. 3.2). They allow to detect and recognize objects in the current image but cannot ascertain a more complete scene representation. Again, spatial context information is needed to transfer these image-based perceptions into a three-dimensional environment. Pose tracking allows to determine the position of the user's head; thereby spatial context of the current perception can be defined (Sec. 3.3).

Interaction Interaction with the user is especially important in ego-vision systems due to the mentioned close coupling and to close the loop. The system must give feedback to the user to facilitate interactive learning and adaptation and to be transparent in its operation. Therefore, basic visualization capabilities have been developed and presented. Furthermore, different input modalities have been proposed and integrated to enhance the usability of the system. Interaction using head gestures (Sec. 4.3.2) constitutes such a modality that is especially designed for the presented hardware setup of the so-called AR gear that is worn by the user as interface enabling ego-vision perception and interaction.

Visual Active Memory Providing assistance is of course more than perception and interaction. When taking the metaphor of a human, we have discussed eyes, ears, hands, and mouth so far; together with their respective cognitive functions, like seeing, hearing, and so on. Assistance functionalities however demand for a brain and a backbone, to keep the metaphor.

Technically spoken, an integration approach is needed, that on the one hand connects and coordinates functionalities, and on the other hand allows to reason, learn, memorize, and recall. In this work, the *visual active memory* (VAM) has been introduced as a conceptual architecture for cognitive visual systems. This memory has to some extend been motivated by human cognition and by further specific demands arising from the ego-vision perspective and assistance scenarios. It basically comprises four different layers that are uniformly represented in the memory: the *sensory*, *perceptual*, *episodic*, and *conceptual* memory. These layers have been comprehensively discussed. The flexible representation allows to overcome the borderlines between these layers, and facilitates learning and adaptation. In this representation, the atomic entities stored in this memory are termed *memory elements* and are basically represented as XML documents as a most flexible basis for their representation. The concept of a central memory not only as a storage but as an active integration broker facilitating coordination as proposed by Wrede et al. [172] proved to be an appropriate basis for the development of the visual active memory concept.

An interactive ego-vision system applying the outlined concepts of a central visual active memory is therefore constituted of several loosely coupled inner and outer *memory processes* that are coordinated by means of the integration broker. The functionality of the assistance system emerges from their coordinated interplay, with all information flow mediated through the memory.

Outer memory processes realize the perception and interaction capabilities, while inner memory processes mostly re-organize, assess, and fuse dedicated parts of the memory con-

tent. These inner processes basically connect the different memory processes and the different layers of the memory. Some inner memory processes have been presented that implement rather generic processes, that typically span different layers in the conceptual architecture of the VAM.

The *hypotheses anchoring* (Sec. 6.4) has been introduced to link percepts to episodes. This is a general problem as any perception is only valid at a very specific moment in time. The anchoring process allows to establish an equivalence between real world entities and the content of the episodic memory. Another generic inner memory process termed "consistency validation" allows to analyze the episodic memory content in order to detect conflicts that arose from probably erroneous perception or reasoning. It utilizes Bayesian networks to model expectations regarding the dependencies of specific episodic instances. Finally, *forgetting* has been motivated by the cognitive foundation that memory capacity is limited and only reliable information should be kept.

System architecture and evaluation A real assistance system comprised of many components has been built on the basis of the visual active memory architecture. The functionality emerged from the interplay of the different memory processes. Thus, the presented system architecture mainly mediates information and coordination flows through this active memory. The system works reliably, so that is was possible to conduct user studies in order to assess its overall performance. This evaluation studies yielded encouraging results regarding the usability, functionality, and reactivity of the whole system, but furthermore allowed to identify bottlenecks and issues to improve. Additionally, I presented an evaluation of the individual components, to which I contributed significantly.

9.2 Discussion and Future Perspectives

The visual active memory with its flexible representation concept, the coordination capabilities of the integration broker architecture, and the generic inner memory processes pave the way towards a flexible architecture picking up concepts of human cognition. Although it has currently only been evaluated in context of ego-vision scenarios, results so far are encouraging that the approach has a more general applicability. Currently, new applications in cognitive robotics are explored. Especially the inner memory process that analyzes the context of memory elements in order to assess the content of the memory, and the generic hypotheses anchoring have disclosed their effectiveness and potential only partially in the yet investigated scenarios.

The ego-vision approach appeared quite promising, especially for assistance scenarios in which the shared perception and close collaboration really facilitates effective assistance. The challenges arising from this ego-vision perspective can be solved and can to some extend be turned into advantages as has been shown in the course of this work. Although not all different concepts have converged into one integrated system at the end, they all have been developed following the basic ideas of cognitive ego-vision systems, and mate together very well. To give an example: The scene decomposition presented in Sec. 3.1 allows to define view contexts as they have been introduced in Sec. 3.3.3. This approach has also been evaluated in a less restricted environment than the user studies have finally been conducted in. So it already indicates how explicit pre-given knowledge in the memory can be reduced in favor for acquired experience and learning. The acquisition of object models is already implemented in an interactive fashion to study how learning can take place interactively.

With the visual active memory, the basis is given to further investigate on interactively learning strategies and collaborative task solving.

Nevertheless, individual challenges arising from specific application scenarios, and a significant amount of assumptions that have to be taken, still impede the development of assistance systems with a wide spectrum of different applications. It must be noted that many of the provided solutions need to be adapted or correctly parameterized depending on the dedicated scenarios. Interaction has been presented as a means to leave learning and adaptation to the user to some extend, and to optimize system performance directly in the scenario. The interactive object learning can be considered as an example here. High usability and algorithms that can learn or adapt in reasonable time on the basis of small data sets are prerequisites for this solution.

Concluding, building complex integrated systems comprised of many different components is always a challenging task. This is especially true, when all components are currently close to or beyond the state of the art in their particular field, but have not been applied in integrated settings yet. Close interaction between the system and the user can facilitate error recoverability and interactive adaptation. This close interaction is inherently provided by ego-vision systems. Therefore, EVSs are especially well suited to study interaction and collaborative learning.

Cognitive ego-vision systems will probably gain more interest in the future with hardware getting smaller and increasing computational power. With new hardware, wearable cognitive assistance systems that take the users perspective become more comfortable and realistic for many application scenarios in industrial and private domains. Progress in human-computer-interaction, cognitive systems, system integration, object and action recognition, and computer vision in general will also improve the quality and widen the range of applications for such systems in particular.

Besides hardware issues the major challenges in future research on cognitive ego-vision system are basically related to challenges of artificial cognitive vision systems in general. Although the approaches presented in this thesis have been proposed to be appropriate for more than one specific application, cognitive systems still need to develop a more general applicability to live up to their name. The role of context-awareness and adaptation will have to be strengthened and a more intertwined processing (of, e.g., object and action recognition) is needed in order to combine bottom-up and top-down schemes more effectively. I assume, that explicit models of contexts and environments do neither work individually, nor can pure data-driven learning lead to assistance systems that compete at a human level. The first is too inflexible and always specific, while the latter suffers from an unmanageable need for data and training time. Therefore, the proposal of cognitive vision systems to combine these basic patterns and successively reduce the amount of pre-given assumptions appears promising in general. Research in these directions will hopefully converge into more flexible adaptive systems, particular for assistance scenarios.

A A Brief Introduction to XML and XPath

A.1 XML

A most complete introduction on XML (eXtensible Markup Language) is given by Birbeck et al. [14], but some general concepts and their terminology are introduced shall be repeated here. XML documents are tree-like structures with different kinds of nodes. The underlying hierarchical representation is termed *Document Object Model (DOM) tree*. In such a DOM tree different types of nodes are defined: *elements* are named nodes, than can contain other child nodes. Thus, elements are used to hierarchically structure the document. As the XML representation forms a tree, exactly one *root element* must be given in a DOM. The second node type is *content*. Content can exist as a child of an element and can be comprised of data in almost any textually representable form. *Attributes* also only exist as children of elements to represent name-value pairs. The different between *attributes* which are data structures in memory elements, and XML attributes should be stressed here. Other types of nodes like comments and processing instructions are defined by the XML data set, but play no relevant role for the concepts in this thesis and are therefore not further explained. XML defines an serialized representation of DOM trees into an as well human- as machine-readable format that syntactically defines a markup language.

A.2 XPath

The active memory infrastructure (AMI) supports a flexible means to access the hierarchical XML-based representation of memory elements via *XPath* [29]. As XPath is very fundamental for the coordination and memory elements access a short introduction on this subject shall be given in the following.

XPath allows to define paths that directly address specific nodes in a DOM tree. The syntax of XPath is likewise similar to navigation in hierarchical file systems. The slash "`/`" is used as a separator between a parent node and its child. A sequence of nodes separated by slashes addresses a specific node in the DOM tree. We will term any specific node referenced by an XPath as *location*. Elements are addressed directly by its name, while attributes are prefixed with "`@`". The content of an element can be accessed by a special function `text()`. To access the `TIMESTAMPS` element in the example outlined in Fig. 5.5, for instance, the corresponding XPath axis is

```
/OBJECT/HYPOTHESIS/TIMESTAMPS .
```
(A.1)

This specific XPath is given in an *absolute* manner indicated by the leading slash. XPath statements are always given in specific *context*. For the absolute XPath presented above, this context is the XML document root node itself. But generally, the context can be any node in a DOM tree, to which an XPath is relatively defined to. Thus, in order to access the `create` attribute in the `CREATED` element relative to the `TIMESTAMPS` element, the correct

XPath will be

```
CREATED/@value .
```
(A.2)

Note, that this XPath has no leading slash as it is defined relatively to the current context – in this case, the `TIMESTAMPS` element node.

But XPath furthermore allows to define *conditions* to be checked for validity. A condition is given in square brackets "[...]". Several tests on equality, arithmetic comparison, and others more are implemented by XPath version 1.0, which are not to be discussed here in detail. Instead, an example clarifies the usage of conditions in XPath statements:

```
/OBJECT[CLASS/text()="Cup"]/REGION/RECTANGLE/@x .
```
(A.3)

This XPath addresses the $\boldsymbol{x}$-coordinate of region defined by a rectangle. But the XPath is restricted by its condition to by valid only if the `CLASS` node contains the string "Cup" as content. If the condition is not fulfilled, the evaluation of that XPath results in an empty set.

B Questionnaire

Questionnaire – Page 1

Fragebogen mit Feedback

VAMPIRE
Visual Active Memory Processes and Interactive REtrieval

User Nummer	Alter	Geschlecht m w	Studiengang; Semesteranzahl

1. Die Darstellung

Wie gut konnten Sie sich auf dem Bildschirm orientieren? sehr gut ☐☐☐☐☐ sehr schlecht

Begründung:.....

Sind die Symbole auf dem Bildschirm selbsterklärend? ja, immer ☐☐☐☐☐ nein, nie

Begründung:.....

2. Der Tragekomfort

Wie empfanden Sie das Tragen des Helms? angenehm ☐☐☐☐☐ unangenehm

Wurden Sie durch das Tragen des Helms am Ausführen der Aufgaben behindert? ja nein

Wenn ja, warum? Begründung:.....

3. Die Bedienerfreundlichkeit

Empfanden Sie Ihre Eingewöhnungszeit als lang oder kurz? sehr lang ☐☐☐☐☐ sehr kurz

Sind die Abläufe während der Benutzung klar verständlich? verständlich ☐☐☐☐☐ unverständlich

Entspricht die Zusammenarbeit mit dem System zur Bewältigung der Aufgabe Ihren Erwartungen? ja, immer ☐☐☐☐☐ nein, nie

Begründung:.....

Ist das vom System gegebene Feedback ausreichend, um die Aufgaben sicher bewältigen zu können? ausreichend ☐☐☐☐☐ unzureichend

Wie bewerten Sie die Zusammenarbeit mit dem System? sehr gut ☐☐☐☐☐ sehr schlecht

Begründung:.....

Questionnaire – Page 2

Fragebogen mit Feedback

VAMPIRE
Visual Active Memory Processes and Interactive REtrieval

Wie wirkt die Benutzung des Systems auf Ihre Bewegungen?	hetzend	☐☐☐☐☐	verlangsamend
	Begründung:.....		
Ist Ihnen die Bedienung des Systems schwer gefallen?	sehr leicht	☐☐☐☐☐	sehr schwer
	Begründung:.....		
Ist Ihnen die Bewältigung der zweiten Aufgabe leichter als die der ersten gefallen?	viel leichter	☐☐☐☐☐	viel schwerer
	Begründung:.....		

4. Die Hilfe-Funktionen

Wie oft bestand nach dem Film und der Eingewöhnungsphase der Bedarf nach Hilfe?	nie	☐☐☐☐☐	häufig
	Begründung:.....		
Wie präzise empfanden Sie die Kennzeichnung der benutzten Objekte? (Kästen um die Objekte)	sehr präzise	☐☐☐☐☐	sehr unpräzise
Wie hilfreich empfanden Sie die Unterstützung durch die Kennzeichnung der benutzten Elemente?	sehr hilfreich	☐☐☐☐☐	gar nicht hilfreich
Wie hilfreich empfanden Sie die Funktion zum Wiederfinden von Objekten? (roter Pfeil)	sehr hilfreich	☐☐☐☐☐	gar nicht hilfreich
Wie präzise empfanden Sie die Funktion zum Wiederfinden von Objekten?	sehr präzise	☐☐☐☐☐	sehr unpräzise
Waren Sie sich immer im Klaren darüber, was das System gerade tat?	ja, immer	☐☐☐☐☐	nein, nie

5. Die Komponenten

Wie beurteilen Sie die Objekterkennung des Systems?	sehr gut	☐☐☐☐☐	sehr schlecht
Wie beurteilen Sie den Vorgang des internen Lernens?	sehr gut	☐☐☐☐☐	sehr schlecht
Wie beurteilen Sie die Zusammenarbeit mit dem System über die Maus?	sehr gut	☐☐☐☐☐	sehr schlecht
Wie beurteilen Sie die Zusammenarbeit mit dem System über Sprache?	sehr gut	☐☐☐☐☐	sehr schlecht
Wie beurteilen Sie die textuellen Anweisungen durch das System?	sehr gut	☐☐☐☐☐	sehr schlecht

Questionnaire – Page 3

Fragebogen mit Feedback

VAMPIRE
Visual Active Memory Processes and Interactive REtrieval

6. Persönlicher Kommentar

Wie beurteilen Sie die Lernfähigkeit des Systems?	sehr gut	☐ ☐ ☐ ☐ ☐	sehr schlecht
Wie beurteilen Sie die Gedächtnisfunktion des Systems?	sehr gut	☐ ☐ ☐ ☐ ☐	sehr schlecht

Was hat Ihnen nicht an dem System gefallen?
.....
.....
.....

Was hat Ihnen besonders an dem System gefallen?
.....
.....
.....

Wie möchten Sie zur Beantwortung der folgenden Frage bitten, lediglich das System zu beurteilen. Der Helm könnte wie im Bild auf der linken Seite gezeigt wird ebenfalls durch eine Brille >dieser< Art ersetzt werden.

Wo sehen Sie alltägliche Anwendungsmöglichkeiten für dieses System?
.....
.....
.....

7. Sonstige Angaben

Haben Sie bereits Vorwissen über dieses System?	ja nein
Haben Sie bereits Erfahrungen mit Systemen dieser Art?	ja nein

Vielen Dank für die Beantwortung unserer Fragen!

C Questionnaire – translations

1. Die Darstellung [The depiction of the screen]
   - ▷ Wie gut konnten Sie sich auf dem Bildschirm orientieren? sehr gut ... sehr schlecht
     [How well was your orientation on the display? very good ... very bad]
   - ▷ Sind die Symbole auf dem Bildschirm selbsterklärend? ja, immer ... nein, nie
     [Are the icons on the screen comprehensible? always ... never]
2. Der Tragekomfort [The wearing comfort]
   - ▷ Wie empfanden Sie das Tragen des Helms? angenehm ... unangenehm
     [What was your impression of wearing the helmet? comfortable ... not comfortable]
3. Die Bedienerfreundlichkeit [The usability]
   - ▷ Empfanden Sie ihre Eingewöhnungszeit als lang oder kurz? sehr lang ... sehr kurz
     [What was your impression of your familiarization, long or short? very long ... very short]
   - ▷ Sind die Abläufe während der Benutzung klar verständlich? verständlich ... unverständlich
     [Is the operation transparent during system usage? comprehensible ... not comprehensible]
   - ▷ Entspricht die Zusammenarbeit mit dem System zur Bewältigung der Aufgabe Ihren Erwartungen? ja, immer ... nein, nie
     [Does the collaboration with the system meet your expectations regarding the task to be fulfilled? always ... never]
   - ▷ Ist das vom System gegebene Feedback ausreichend, um die Aufgabe sicher zu bewältigen? ausreichend ... unzureichend
     [Is the feedback that is given by the system sufficient for the successful execution of the task? sufficient ... not sufficient]
   - ▷ Wie bewerten Sie die Zusammenarbeit mit dem System? sehr gut ... sehr schlecht
     [How would you assess the collaboration with the system? very good ... very bad]
   - ▷ Wie wirkt die Benutzung des Systems auf Ihre Bewegung? hetzend ... verlangsamend
     [How does the system usage affect your own movements? speed up ... slow down]
   - ▷ Ist Ihnen die Bedienung des Systems schwergefallen? sehr leicht ... sehr schwer
     [How difficult has it been to handle the system? easy ... difficult]

- ▷ Ist Ihnen die Bew"altigung der zweiten Aufgabe leichter als die der ersten gefallen? viel leichter ... viel schwerer
  [Was the successful execution of the second task much easier than the first one? much easier ... much harder]

4. Die Hilfe-Funktion [The help function]

   - ▷ Wie oft bestand nach dem Film und der Eingewöhnungsphase der Bedarf nach Hilfe? nie ... häufig
     [How frequent did you need help after the video and the accustom period? rare ... often]
   - ▷ Wie präzise empfanden Sie die Kennzeichnung der benutzten Objekte? (Kästen um die Objekte) sehr präzise ... sehr unpräzise
     [What was your impression regarding the precision of the marking of objects used? (boxed objects) very precise ... very unprecise]
   - ▷ Wie hilfreich empfanden Sie die Unterstützung durch die Kennzeichnung der benutzten Elemente? sehr hilfreich ... gar nicht hilfreich
     [What was your impression regarding the helpfulness of the marking of items used? very helpful ... not helpful]
   - ▷ Wie hilfreich empfanden Sie die Funktion zum Wiederfinden von Objekten? (roter Pfeil) sehr hilfreich ... gar nicht hilfreich
     [What was your impression regarding the helpfulness of the function for object finding (red arrow) very helpful ... not helpful]
   - ▷ Wie präzise empfanden Sie die Funktion zum Wiederfinden von Objekten? sehr präzise ... sehr unpräzise
     [What was you impression regarding the precision of the function for object finding? very precise ... very unprecise]
   - ▷ Ware Sie sich immer im Klaren darüber, was das System gerade tat? ja, immer ... nein, nie
     [Have you always been aware of what the system currently did? always ... never]

5. Die Komponenten [The components]

   - ▷ Wie beurteilen Sie die Objekterkennung des Systems? sehr gut ... sehr schlecht
     [How would you assess the object recognition of the system? very good ... very bad]
   - ▷ Wie beurteilen Sie den Vorgang des internen Lernens? sehr gut ... sehr schlecht
     [How would you assess the process of internal learning? very good ... very bad]
   - ▷ Wie beurteilen Sie die Zusammenarbeit mit dem System über die Maus? sehr gut ... sehr schlecht
     [How would you assess the iteraction with the system via mouse? very good ... very bad]
   - ▷ Wie beurteilen Sie die Zusammenarbeit mit dem System über Sprache? sehr gut ... sehr schlecht
     [How would you assess the iteraction with the system via speech? very good ... very bad]
   - ▷ Wie beurteilen Sie die textuellen Anweisungen durch das System? sehr gut ... sehr schlecht
     [How would you assess the textual instructions by the system? very good ... very bad]

6. Persönlicher Kommentar [personal comment]

   ▷ Wie beurteilen Sie die Lernfähigkeit des Systems? sehr gut ... sehr schlecht
   [How would you assess the ability of the system to learn? very good ... very bad]

   ▷ Wie beurteilen Sie die Gedächtnisfunktion des Systems? sehr gut ... sehr schlecht
   [How would you assess the memory function of the system? very good ... very bad]

Bibliography

[1] A. Aamodt and M. Nygard. Different roles and mutual dependencies of data, information and knowledge - an ai perspective on their integration. *Data and Knowledge Engineering*, 16:191–222, 1995.

[2] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In M. Landy and J. A. Movshon, editors, *Computational Models of Visual Processing*, pages 3–20. MIT Press, Cambridge, 1991.

[3] A. Ahmadyfard and J. Kittler. A comparative study of two object recognition methods. In *Proc. British Machine Vision Conference*, pages 363–372, 2002.

[4] Y. Aloimonos. Active vision revisited. In Y. Aloimonos, editor, *Active Perception*, pages 1–18. Lawrence Erlbaum, 1993.

[5] H. Anton. *Elementary Linear Algebra*. John Wiley and Son, 1994.

[6] M. Arens and H.-H. Nagel. Representation of behavioral knowledge for planning and plan-recognition in a cognitive vision system. In *Lecture Notes in Computer Science, Volume 2479, Jan 2002, Page 268*, volume 2479, pages 268–282. Springer, Jan. 2002.

[7] R. T. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385., 1997.

[8] A. Baddeley. *Human Memory: Theory and Practice*. Psychology Press, Hove, East Sussex, 2002.

[9] F. Bajramovic, C. Gräßl, and J. Denzler. Efficient combination of histograms for real-time tracking using mean-shift and trust-region optimization. In *Proc. Pattern Recognition Symposium (DAGM)*, pages 254–261, Heidelberg, 2005. Springer.

[10] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *Proc. CVPR*, pages 434–441, June 1998.

[11] C. Bauckhage and T. Käster. Benefits of separable, multilinear classification. In *Proc. Int. Conf. on Pattern Recognition*, volume 3, pages 1240–1243. IEEE, 2006.

[12] C. Bauckhage, J. Fritsch, K. 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, Pittsburgh, PA, 2002.

[13] H. Bekel, I. Bax, G. Heidemann, and H. Ritter. Adaptive computer vision: Online learning for object recognition. In C. E. Rasmussen, H. H. Bülthoff, B. Schölkopf, and M. A. Giese, editors, *Proc. Pattern Recognition Symposium (DAGM)*, volume 3175 of *Lecture Notes in Computer Science*, pages 447–454. Springer, 2004.

[14] M. Birbeck, J. Duckett, O. G. Gudmundsson, B. Loesgen, P. Kobak, E. Lenz, S. Livingstone, D. Marcus, S. Mohr, N. Ozu, J. Pinnock, K. Visco, A. Watt, K. Williams, and Z. Zaev. *Professional XML*. Wrox Press Inc., 2nd edition, 2001.

[15] M. J. Black and A. D. Jepson. A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. In *Proc. European Conf. on Computer Vision*, volume 1, pages 909–924, 1998.

[16] A. Bobick. Movement, activity, and action: The role of knowledge in the perception of motion. *Proceedings of the Royal Society of London*, 352:1257–1265, 1997.

[17] C. Breazeal and B. Scassellati. A context-dependent attention system for a social robot. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1146–1153, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-613-0.

[18] C. Breazeal, D. Buchsbaum, J. Gray, D. Gatenby, and B. Blumberg. Learning from and about others: Towards using imitation to bootstrap the social understanding of others by robots. *Artificial Life*, 11(1):31–62, 2005. ISSN 1064-5462. doi: http://dx.doi.org/10.1162/1064546053278955.

[19] R. Brooks. Symbolic reasoning around 3-d models and 2-d images. *Artificial Intelligence*, 17:285–348, 1981.

[20] Carnegie Mellon University's Software Engineering Institute. Software engineering glossary. WWW, Oct. 2006. URL `http://www.sei.cmu.edu/opensystems/glossary.html#s`.

[21] C. Castelfranchi. Cognitive systems: Towards an integration of symbolic and sensormotor intelligence? *ERCIM News*, 53:10–11, Apr. 2003.

[22] M. Chandraker, C. Stock, and A. Pinz. Real time camera pose in a room. In *Proc. Int. Conf. on Computer Vision Systems*, volume 2626 of *LNCS*, pages 98–110, Graz, Austria, April 2003.

[23] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000. URL `http://citeseer.ist.psu.edu/chen00survey.html`.

[24] H. D. Cheng, X. H. Jiang, Y. Sun, and J. Wang. Color image segmentation: Advances and prospects. *Pattern Recognition*, 34(12):2259–2281, Dec. 2001.

[25] H. Christensen. Cognitive (vision) systems. *ERCIM News*, 53:17–18, Apr. 2003.

[26] H. I. Christensen. CogVis: Cognitive vision systems. [online], July 2006. URL `http://cogvis.nada.kth.se/techannex.html`.

[27] H. I. Christensen. CoSy: Cognitive systems for cognitive assistants. [online], July 2006. URL `http://www.cognitivesystems.org/abstract.asp`.

[28] W. Christmas, A. Kostin, F. Yan, I. Kolonias, and J. Kittler. A system for the automatic annotation of tennis matches. In *Fourth International Workshop on Content-Based Multimedia Indexing*, 2005. Invited paper.

[29] J. Clark and S. DeRose. XML Path Language, W3C Recommendation. Technical Report REC-xpath-19991116, World Wide Web Consortium, Nov. 1999. URL `http://www.w3.org/TR/1999/REC-xpath-19991116`.

[30] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[31] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–575, May 2003.

[32] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, June 2001.

[33] S. Coradeschi and A. Saffiotti. Perceptual anchoring of symbols for action. In *Proc. Intl. Conf. on Artificial Intelligence*, pages 407–416, 2001.

[34] S. Coradeschi and A. Saffiotti. Anchoring symbols to sensor data: preliminary report. In *Proc. of the 17th Conf. on Artificial Intelligence (AAAI-00)*, pages 129–135. AAAI/MIT Press, July 2000.

[35] S. Coradeschi and A. Saffiotti. An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43(2–3):85–96, May 2003.

[36] COSPAL consortium. COSPAL: Cognitive systems using perception-action learning. [online], July 2006. URL `http://www.cospal.org/`.

[37] J. Crowley and H. Christensen, editors. *Vision as Process*. Springer, 1995.

[38] H. Cruse. The evolution of cognition – a hypothesis. *Cognitive Science*, 27(1):135–155, 2003.

[39] R. Cutler and M. Turk. View-based interpretation of real-time optical flow for gesture recognition. In *Proceedings International Conference on Automatic Face and Gesture Recognition*, pages 416–421, Apr. 1998. doi: 10.1109/AFGR.1998.670984.

[40] J. Davis and A. Bobick. The representation and recognition of human movement using temporal templates. In *Proc. Conf. on Computer Vision and Pattern Recognition*, pages 928–934, 17-19 June 1997. doi: 10.1109/CVPR.1997.609439.

[41] J. Davis and S. Vaks. A Perceptual User Interface for Recognizing Head Gesture Acknowledgements. In *Proc. Workshop on Perceptive User Interfaces*, 2001.

[42] A. J. Davison and D. W. Murray. Mobile robot localisation using active vision. In *Proc 5th European Conference on Computer Vision*, volume 1407 of *LNCS*, pages 809–825, Freiburg, June 1998. Springer.

[43] Defense Advanced Research Projects Agency. The darpa urban grand challenge. WWW, Oct. 2006. URL `http://www.darpa.mil/grandchallenge`.

[44] B. Deutsch, C. Gräßl, F. Bajramovic, and J. Denzler. A comparative evaluation of template and histogram based 2d tracking algorithms. In *Proc. Pattern Recognition Symposium (DAGM)*, Heidelberg, 2005. Springer.

[45] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[46] S. J. Dickinson, H. I. Christensen, J. K. Tsotsos, and G. Olofsson. Active object recognition integrating attention and viewpoint control. *Computer Vision and Image Understanding*, 67(3):239–260, 1997.

[47] A. Dix, J. Finlay, G. Abowd, and R. Beale. *Human-computer interaction*. Prentice Hall Europe, 1998.

[48] B. Draper, A. Hanson, and E. Riseman. Knowledge-directed vision: control, learning and integration. *Proc. of the IEEE*, 84(11):1625–1681, Nov. 1996.

[49] B. Duffy and G. Joue. Intelligent robots: The question of embodiment. In *Proc. of the Brain-Machine Workshop*, 2000.

[50] M. M. Ellenrieder, L. Krüger, D. Stößel, and M. Hanheide. A versatile model-based visibility measure for geometric primitives. In *Proceedings of Scandinavian Conference on Image Analysis*, volume 3540 of *LNCS*, pages 669–678, Heidelberg, Germany, 2005.

[51] O. Faugeras. *Three-Dimensional Computer Vision*. MIT Press, Cambridge, 1993.

[52] G. Fink. Developing HMM-based Recognizers with ESMERALDA. In *Text, Speech and Dialogue*, volume 1692 of *LNAI*, pages 229–234. Springer, 1999.

[53] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[54] T. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(3–4):143–166, 2003.

[55] J. Fritsch, N. Hofemann, and G. Sagerer. Combining sensory and symbolic data for manipulative gesture recognition. In *Proc. Int. Conf. on Pattern Recognition*, number 3, pages 930–933, Cambridge, United Kingdom, 2004. IEEE.

[56] K. Fu and J. Mui. A survey on image segmentation. *Pattern Recognition*, 13:3–16, 1981.

[57] N. Gorges. An Approach to a Pictorial Memory – Representing a Scene as a Collection of Planar Mosaics. Master's thesis, Bielefeld University, 2004.

[58] N. Gorges, M. Hanheide, W. Christmas, C. Bauckhage, G. Sagerer, and J. Kittler. Mosaics from arbitrary stereo video sequences. In C. E. Rasmussen, H. H. Bülthoff, M. A. Giese, and B. Schölkopf, editors, *Proc. Pattern Recognition Symposium (DAGM)*, volume 3175, pages 342–349, Heidelberg, Germany, 2004. Springer-Verlag.

[59] G. Granlund. Organization of architectures for cognitive vision systems. In *Proceedings of Workshop on Cognitive Vision*, Schloss Dagstuhl, Germany, 2003.

[60] G. Granlund. A Cognitive Vision Architecture Integrating Neural Networks with Symbolic Processing. *Künstliche Intelligenz*, 2:18–24, 2005.

[61] G. Granlund. Organization of architectures for cognitive vision systems. In H. H. Nagel and H. I. Christensen, editors, *Cognitive Vision Systems*, pages 39–58. Springer, Heidelberg, 2005.

[62] G. H. Granlund. The complexity of vision. *Signal Processing*, 74(1):101–126, Apr. 1999.

[63] C. Gräßl, T. Zinßer, and H. Niemann. Efficient hyperplane tracking by intelligent region selection. In *Proc. IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 51–55, Lake Tahoe, USA, 2004.

[64] A. Haasch, S. Hohenner, S. Hüwel, M. Kleinehagenbrock, S. Lang, I. Toptsis, G. A. Fink, J. Fritsch, B. Wrede, and G. Sagerer. BIRON – The Bielefeld Robot Companion. In *Proc. Int. Workshop on Advances in Service Robotics*, pages 27–32, 2004.

[65] D. L. Hall and J. Llinas, editors. *Handbook on Multisensor Data Fusion*. CRC Press, 2001.

[66] M. Hanheide, C. Bauckhage, and G. Sagerer. Combining environmental cues & head gestures to interact with wearable devices. In *Proceedings of the 7th International Conference on Multimodal Interfaces*, pages 25–31. ACM, Oct. 2005. ISBN 1-59593-190-2.

[67] M. Hanheide, N. Hofemann, and G. Sagerer. Action recognition in a wearable assistance system. In *Proc. Int. Conf. on Pattern Recognition*. IEEE, 2006.

[68] A. R. Hanson and E. M. Riseman. Visions: a computer system for interpreting scenes. In R. Hanson and E. M. Riseman, editors, *Computer Vision Systems*, pages 303–333. Academic Press, 1978.

[69] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. Alvey Vision Conference*, pages 147–151, 1988.

[70] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2001.

[71] G. Heidemann, H. Bekel, I. Bax, and H. Ritter. Interactive online learning. *Pattern Recognition and Image Analysis*, 15(1):55–58, 2005.

[72] M. Holcombe and F. Ipate. *Correct Systems: Building a Business Process Solution*. Springer-Verlag, London, 1998.

[73] B. Hu and C. Brown. Interactive indoor scene reconstruction from image mosaics using cuboid structure. In *Proc. Motion & Video Computing*, pages 208–213, Aug. 2002.

[74] M. Irani, P. Anandan, and S. Hsu. Mosaic based representations of video sequences ans their applications. In *Proceedings Int. Conference on Computer Vision*, 1995.

[75] M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *Proc 6th Int. Conf. Computer Vision*, pages 107–112, 1998.

[76] JAST consortium. JAST: Joint action science and technology. [online], July 2006. URL `http://www.jast-net.gr/`.

[77] T. Jebara, C. Eyster, J. Weaver, T. Starner, and A. Pentland. Stochasticks: Augmenting the billiards experience with probabilistic vision and wearable computers. In *ISWC '97: Proceedings of the 1st IEEE International Symposium on Wearable Computers*, pages 138–145, Washington, DC, USA, Oct. 1997. IEEE Computer Society. ISBN 0-8186-8192-6.

[78] F. V. Jensen. *An introduction to Bayesian networks*. UCL Press, London, 1996.

[79] F. V. Jensen. *Bayesian networks and decision graphs*. Statistics for engineering and information science. Springer, New York, Berlin, Heidelberg, 2001.

[80] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME Journal of Basic Engineering*, 82:35–45, 1960.

[81] F. Kaplan and V. V. Hafner. The challenges of joint attention. In *Proceedings of the Fourth International Workshop on Epigenetic Robotics*, pages 67–74, 2004.

[82] A. Kapoor and R. Picard. A real-time head nod and shake detector. In *Proc. Workshop on Perceptive User Interfaces*, 2001.

[83] T. Käster, M. Pfeiffer, and C. Bauckhage. Usability Evaluation of Image Retrieval Beyond Desktop Applications. In *Proc. Int. Conf. on Multimedia and Expo*, pages 385–388. IEEE, 2006.

[84] M. Kipp. Anvil – a generic annotation tool for multimodal dialogue. In *Proc. European Conference on Speech Communication and Technology (Eurospeech)*, pages 1367–1370, Sept. 2001.

[85] J. Kittler, J. Matas, M. Bober, and L. Nguyen. Image interpretation: Exploiting multiple cues. In *Proc. Conference on Image Processing and Applications*, pages 1–5, 1995.

[86] J. Kittler, W. J. Christmas, A. Kostin, F. Yan, I. Kolonias, and D. Windridge. A memory architecture and contextual reasoning framework for cognitive vision. In *Image Analysis*, number 3540 in Lecture Notes in Computer Science, pages 343–358. Springer, 2005.

[87] R. Kjeldsen. Head Gestures for Computer Control. In *IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 61–68, 2001.

[88] G. Klinker, K. Ahlers, D. Breen, P.-Y. Chevalier, C. Crampton, D. Greer, D. Koller, A. Kramer, E. Rose, M. Tuceryan, and R. Whitaker. Confluence of computer vision and interactive graphics for augmented reality. *Presence: Teleoperations and Virtual Environments*, 6(4):433–451, August 1997.

[89] M. Kölsch and M. Turk. Fast 2d hand tracking with flocks of features and multi-cue integration. In *IEEE Workshop on Real-Time Vision for Human-Computer Interaction (at CVPR)*, volume 10, page 158, 2004.

[90] T. Kölzow. *Wissensbasierte Entwicklungsumgebung für Bildanalysesysteme aus dem industriellen Bereich*. PhD thesis, Universität Bielefeld, 2003.

[91] R. Kumar, P. Anandan, M. Irani, J. Bergen, and K. Hanna. Representation of scenes from collections of images. In *Proc. IEEE Workshop on Representations of Visual Scenes*, pages 10–17, 1995.

[92] S. Lauritzen. The EM-algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19(2):191–201, 1995.

[93] M. Lhuillier and L. Quan. Robust dense matching using local and global geometric constraints. In *Proc. ICPR*, pages 968–972, 2000.

[94] H. Lieberman and T. Selker. Out of context: Computer systems that adapt to, and learn from, context. *IBM Systems Journal*, 39(3 & 4):617–632, 2000.

[95] F. Lömker, S. Wrede, M. Hanheide, and J. Fritsch. Building modular vision systems with a graphical plugin environment. In *Proc. of International Conference on Vision Systems*, St. Johns University, Manhattan, New York City, USA, January 2006. IEEE.

[96] H. López Romero. Inertial-based Head Gesture Recognition for Intuitive User Interaction. Master's thesis, Bielefeld University, 2005. in german.

[97] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision*, pages 1150–1157, Sept. 1999.

[98] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. L. Cam and J. Neyman, editors, *Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–296, 1967.

[99] MACS consortium. MACS: Multi-sensory autonomous cognitive systems. [online], July 2006. URL `http://www.macs-eu.org/`.

[100] N. Maillot, M. Thonnat, and A. Boucher. Towards ontology-based cognitive vision. *Machine Vision and Applications*, 16(1):33–40, Dec. 2004.

[101] S. Mann. Wearcam (the wearable camera). In *Proceedings of the International Symposium on Wearable Computers*, pages 124–131, Oct. 1998.

[102] D. Marr. *Vision: a computational investigation into the human representation and processing of visual information.* Freeman, San Francisco, 1982.

[103] J. Matas, P. Remnagnino, J. Kittler, and J. Illingworth. Control of scene interpretation. In J. Crowley and H. Christensen, editors, *Vision as Process*, pages 347–371. Springer, 1995.

[104] K. Messer, W. Christmas, E. Jaser, J. Kittler, B. Levienaise-Obadia, and D. Koubaroulis. A unified approach to the generation of semantic cues for sports video annotation. *Signal Processing*, 83:357–383, 2005. Special issue on Content Based Image and Video Retrieval.

[105] G. Metta, D. Vernon, and G. Sandini. The robotcub approach to the development of cognition: Implications of emergent systems for a common research agenda in epigenetic robotics. In *Proceddings Epigenetic Robotics Workshop*, July 2005.

[106] K. Mikolajczyk, B. Leibe, and B. Schiele. Local features for object class recognition. In *Proc. Int. Conference on Computer Vision*, volume 2, pages 1792–1799, 2005. doi: http://doi.ieeecomputersociety.org/10.1109/ICCV.2005.146.

[107] B. Möller, D. Williams, and S. Posch. Robust image sequence mosaicing. In *Proc. of 25th DAGM Symposium*, LNCS 2781, pages 386–393, 2003.

[108] D. Moore, I. Essa, and I. Hayes, M.H. Exploiting human actions and object context for recognition tasks. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 80–86, Sept. 1999. doi: 10.1109/ICCV.1999.791201.

[109] U. Muehlmann, M. Ribo, P. Lang, and A. Pinz. A new high speed cmos camera for real-time tracking applications. In *IEEE International Conference on Robotics and Automation*, pages 5195–5200, 2004.

[110] K. Murphy, A. Torralba, and W. T. Freeman. Using the forest to see the trees: A graphical model relating features, objects, and scenes. In *Proc. Conf. on Neural Information Processing Systems*, 2003.

[111] K. P. Murphy. The bayes net toolbox for matlab. *Computing Science and Statistics*, 33, 2001.

[112] Y. Nagai. Learning to comprehend deictic gestures in robots and human infants. In *Proc. of the 2005 IEEE Int. Workshop on Robot and Human Interactive Communication*, pages 217–222, Kyoto, Japan, Aug. 2005.

[113] Y. Nagai, K. Hosoda, A. Morita, and M. Asada. A constructive model for the development of joint attention. *Connection Science*, 15(4):211–229, Dec. 2003.

[114] Y. Nakamura, J. Ohde, and Y. Otha. Structuring personal activity records based on attention: Analysing videos from a head-mounted camera. In *Proc. of International Conference on Pattern Recognition*, pages 4222–4225, Sept. 2000.

[115] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-20). Technical report, Columbia University, 1996.

[116] H. Niemann, G. Sagerer, S. Schröder, and F. Kummert. Ernest: A semantic network system for pattern understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):883–905, 1990.

[117] R. F. Office:. The official robocup site. WWW, Oct. 2006. URL `http://www.robocup.org`.

[118] K. Pahlavan, T. Uhlin, and J.-O. Eklundh. Active vision as a methodology. In Y. Aloimonos, editor, *Active Perception*. Lawrence Erlbaum, 1993.

[119] 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, July 1997.

[120] S. Peleg and J. Herman. Panoramic mosaics by manifold projection. In *Proc. ICPR*, pages 338–343, 1997.

[121] S. Peleg, B. Rousso, A. Rav-Acha, and A. Zomet. Mosaicing on adaptive manifolds. *IEEE PAMI*, 22(10):1144–1154, Oct. 2000.

[122] A. Pentland. Perceptual user interfaces: perceptual intelligence. *Commun. ACM*, 43 (3):35–44, 2000. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/330534.330536.

[123] V. Peters. Effizientes Training ansichtsbasierter Gesichtsdetektoren. Master's thesis, Bielefeld University, 2006. in german.

[124] J. L. Peterson. *Petri Net Theory and The Modeling of Systems*. Prentice Hall, Inc., Englewood Cliffs, Massachusetts, 1981.

[125] J. Piater and R. Grupen. Feature learning for recognition with bayesian networks. In *Proc. International Conference on Pattern Recognition*, volume 1, pages 17–20, Sept. 2000. doi: 10.1109/ICPR.2000.905267.

[126] M. Pilu. On the use of attention clues for an autonomous wearable camera. Technical report, HP Laboratories Bristol, 2003. URL `http://www.hpl.hp.com/personal/Maurizio_Pilu/publcs/publicat.htm`.

[127] J. Preece, D. Benyon, G. Davies, and L. Keller. *A guide to usability*. Addison Wesley, 1993.

[128] R. P. N. Rao and D. H. Ballard. An active vision architecture based on iconic representations. *Artificial Intelligence Journal*, 78(1–2):461–505, 1995.

[129] J. Rehg, K. Murphy, and P. Fieguth. Vision-based speaker detection using bayesian networks. In *Proc. Int.Conf. on Computer Vision and Pattern Recognition*, volume 2, June 1999. doi: 10.1109/CVPR.1999.784617.

[130] G. Reitmayr and D. Schmalstieg. Collaborative augmented reality for outdoor navigation and information browsing. In *Proc. Symposium Location Based Services and TeleCartography*, 2004.

[131] P. Ribeiro and J. Santos-Victor. Human activities recognition from video: modeling, feature selection and classification architecture. In *Proc. Workshop on Human Activity Recognition and Modelling*, pages 61–70, Oxford, Sept. 2005. Workshop on Human Activity Recognition and Modelling pp 61-70, Oxford, Sept 2005.

[132] M. Ribo, H. Ganster, M. Brandner, P. Lang, C. Stock, and A. Pinz. Hybrid tracking for outdoor ar applications. *IEEE Computer Graphics and Applications Magazine*, 22 (6):54–63, Nov. 2002.

[133] M. Ribo, M. Brandner, and A. Pinz. A flexible software architecture for hybrid tracking. In *Int. Conf. on Advanced Robotics*, volume 3, pages 1899–1906, Coimbra, 2003.

[134] M. Richter. Sichtgestützte Objektverfolgung und Aktionserkennung mit einer nichtstatischen Kamera. Master's thesis, Bielefeld University, 2005. in german.

[135] K. Rohlfing, J. Fritsch, and B. Wrede. Learning to manipulate objects: A quantitative evaluation of motionese. In *Third International Conference on Development and Learning (ICDL 2004)*, page 27, La Jolla, CA, Oct. 2004. ISBN 0-615-12704-5.

[136] J. P. Rolland, Y. Baillot, and A. A. Goon. A survey of tracking technologies for virtual environments. In W. Barfield and T. Caudell, editors, *Fundamentals of wearable computers and augmented reality*, pages 67–112. Lawrence Erlbaum, 2001.

[137] P. Rosas, J. Wagemans, M. Ernst, and F. Wichmann. Texture and haptic cues in slant discrimination: Reliability-based cue weighting without statistically optimal cue combination. *Optical Society of America*, 22(5):801–809, 2005.

[138] P. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79:871–880, 1984.

[139] M. Sadeghi and J. Kittler. A Comparative Study of Data Fusion Strategies in Face Verification. In *Proc. 12th European Signal Processing Conference*, page to appear, 2004.

[140] G. Sandini, G. Metta, and D. Vernon. Robotcub: An open framework for research in embodied cognition. In *Procedings International Conference on Humanoid Robots*, Nov. 2004.

[141] R. E. Schapire. A brief introduction to boosting. In *Proc. of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1405, 1999.

[142] B. Schiele, N. Oliver, T. Jebara, and A. Pentland. An interactive computer vision system dypers: Dynamic personal enhanced reality system. In *ICVS*, volume 1542 of *LNCS*, pages 51–65, 1999.

[143] J. Shi and C. Tomasi. Good features to track. In *Proc. Comuter Vision and Pattern Recognition*, pages 593–600, Seattle, June 1994.

[144] H. Siegl and A. Pinz. A Mobile AR kit as a Human Computer Interface for Cognitive Vision. In *Int. Workshop on Image Analysis for Multimedia Interactive Services*, page to appear, Lissabon, 2004.

[145] H. Siegl, M. Brandner, H. Ganster, P. Lang, A. Pinz, M. Ribo, and C. Stock. A mobile augmented reality system. In *Proc. Int. Conf. on Computer Vision Systems*, pages 13–14, Graz, 2003.

[146] H. Siegl, G. Schweighofer, and A. Pinz. An ar human computer interface for object localization in a cognitive vision framework. In N. Sebe, M. S. Lew, and T. S. Huang, editors, *Int. Workshop on Computer Vision in Human-Computer Interaction, ECCV 2004*, volume 3058 of *Lecture Notes in Computer Science*, pages 176–186, Prague, Czech Republic, May 2004. Springer.

[147] H. Siegl, M. Hanheide, S. Wrede, and A. Pinz. Augmented reality as human computer interface in a cognitive vision system. *Image and Vision Computing*, 2006. to appear.

[148] T. Starner, B. Schiele, and A. Pentland. Visual contextual awareness in wearable computing. In *Proceedings of the International Symposium on Wearable Computers*, pages 50–57, Pittsburgh, Pennsylvania, USA, Oct. 1998.

[149] R. Stiefelhagen, J. Yang, and A. Waibel. Estimating focus of attention based on gaze and sound. In *Proceedings of the Workshop on Perceptive User Interfaces*, Orlando, Florida, 2001.

[150] R. Stiefelhagen, C. Fuegen, P. Giesemann, H. Holzapfel, K. Nickel, and A. Waibel. Natural Human-Robot Interaction using Speech, Gaze and Gestures. In *EEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2422–2427, 2004.

[151] C. Stock and A. Pinz. Tracking of natural landmarks for augmented-reality purpose. In *Joint Hungarian-Austrian Conference on Image Processing and Pattern Recognition*, pages 343–349, Veszprém, Hungary, May 2005.

[152] C. Stock and A. Pinz. Vision-based tracking-framework using natural landmarks for indoor augmented reality. In *submitted to ISMAR*, 2005. submitted to ISMAR.

[153] S. Thad, S. Mann, B. Rhodes, J. Levine, J. Healey, D. Kirsch, R. Picard, and A. Pentland. Augmented reality through wearable computing. *Presence*, 6(4):386–398, 1997.

[154] The VAMPIRE Consortium. Visual Active Memory Processes and Interactive Retrieval. www, 2002–2005. URL `http://www.vampire-project.org`. IST-2001-34401.

[155] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, Sept. 2006. doi: 10.1002/rob.20147.

[156] T. Tian, C. Tomasi, and D. Heeger. Comparison of approaches to egomotion computation. In *Proc. Int.Conf. on Computer Vision and Pattern Recognition*, pages 315–320, 1996.

[157] E. Tulving. *Elements of Episodic Memory*. Clarendon Press, Oxford, 1983.

[158] E. Tulving. Organization of memory: quo vadis? In M. Gazzaniga, editor, *The Cognitive Neurosciences*, pages 839–847. MIT Press, Cambridge, 1995.

[159] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuro Science*, 3(1):71–86, 1991.

[160] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, volume 1, pages 511–518, Kauai, Hawaii, Dec. 2001.

[161] S. Wachsmuth and G. Sagerer. Bayesian networks for speech and image integration. In *Proc. of 18th National Conf. on Artificial Intelligence (AAAI-2002)*, pages 300–306, Edmonton, Alberta, Canada, 2002.

[162] S. Wachsmuth, S. Wrede, M. Hanheide, and C. Bauckhage. An active memory model for cognitive computer vision systems. *KI-Journal, Special Issue on Cognitive Systems*, 19(2):25–31, 2005.

[163] S. Wachsmuth, S. Wrede, and M. Hanheide. Coordinating interactive vision behaviors for cognitive assistance. *Computer Vision and Image Understanding*, 2006. to appear.

[164] M. Wagner and G. Klinker. An architecture for distributed spatial configuration of context aware applications. In *2nd Int. Conf. on Mobile and Ubiquitous Multimedia*, Norrköping, Sweden, Dec. 2003.

[165] E. Waltz. Information understanding: integrating data fusion and data mining processes. In *Proceedings IEEE Symbosium Circuits and Systems*, volume 6, pages 553–556, May 1998. doi: 10.1109/ISCAS.1998.705333.

[166] C. Wisneski, H. Ishii, A. Dahley, M. Gobert, S. Brave, B. Ullmer, and P. Yarin. Ambient displays: Turning architectural space into an interface between people and digital information. In N. Streitz, S. Konomi, and H. Burkhardt, editors, *Proc. Int. Workshop on Cooperative Buildings*, pages 22–32, Darmstadt, Germany, Feb. 1998.

[167] F. Woelk and R. Koch. Robust monocular detection of independent motion by a moving observer. In *Proc. International Workshop on Complex Motion*, Günzburg, Germany, Oct. 2004.

[168] S. Wrede. *An Information-Driven Integration Framework for Cognitive Systems*. PhD thesis, Universität Bielefeld, 2007. to appear.

[169] S. Wrede, J. Fritsch, C. Bauckhage, and G. Sagerer. An xml based framework for cognitive vision architectures. In *Proc. Int. Conf. on Pattern Recognition*, number 1, pages 757–760, 2004.

[170] S. Wrede, M. Hanheide, C. Bauckhage, and G. Sagerer. An active memory as a model for information fusion. In *Int. Conf. on Information Fusion*, number 1, pages 198–205, Stockholm, Sweden, 2004.

[171] S. Wrede, W. Ponweiser, C. Bauckhage, G. Sagerer, and M. Vincze. Integration frameworks for large scale cognitive vision systems - an evaluative study. In *Proc. Int. Conf. on Pattern Recognition*, number 1, pages 761–764, 2004.

[172] S. Wrede, M. Hanheide, S. Wachsmuth, and G. Sagerer. Integration and coordination in a cognitive vision system. In *Proc. of International Conference on Computer Vision Systems*, St. Johns University, Manhattan, New York City, USA, 2006. IEEE. IEEE ICVS'06 Best Paper Award.

[173] C. Yu and D. Ballard. Learning to recognize human action sequences. In *Development and Learning, 2002. Proceedings. The 2nd International Conference on*, pages 28–33, 12-15 June 2002. doi: 10.1109/DEVLRN.2002.1011726.

[174] T. Ziemke. What's that thing called embodiment? In *Proceedings of the 25th Meeting of the Cognitive Society*, July/Aug. 2003.

[175] T. Zinßer, C. Gräßl, and H. Niemann. Efficient feature tracking for long video sequences. In C. E. Rasmussen, H. H. Bülthoff, M. A. Giese, and B. Schölkopf, editors, *Pattern Recognition, 26th DAGM Symposium*, volume 3175 of *LNCS*, pages 326–333, Tübingen, Germany, September 2004. Springer-Verlag, Berlin, Heidelberg, New York.

[176] I. Zoghlami, O. Faugeras, and R. Deriche. Using geometric corners to build a 2D mosaic from a set of images. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 420–425, June 1997.