Sven Rebhan

# Task- and Knowledge-Driven Scene Representation

# Task- and Knowledge-Driven Scene Representation

A Flexible On-Demand System Architecture for Vision

Von Sven Rebhan

Universitätsverlag Ilmenau
2011

## Impressum

**Bibliografische Information der Deutschen Nationalbibliothek**
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über http://dnb.d-nb.de abrufbar.

Diese Arbeit hat der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau als Dissertation vorgelegen.

| | |
|---|---|
| Tag der Einreichung: | 18. Dezember 2009 |
| 1. Gutachter: | Prof. Dr.-Ing. Horst-Michael Groß (Technische Universität Ilmenau) |
| 2. Gutachter: | Dr. Julian Eggert (Honda Research Institute Europe GmbH, Offenbach/Main) |
| 3. Gutachter: | Prof. Dr.-Ing. Bärbel Mertsching (Universität Paderborn) |
| Tag der Verteidigung: | 11. August 2010 |

Technische Universität Ilmenau/Universitätsbibliothek
**Universitätsverlag Ilmenau**
Postfach 10 05 65
98684 Ilmenau
www.tu-ilmenau.de/universitaetsverlag

**Herstellung und Auslieferung**
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
www.mv-verlag.de

ISBN 978-3-939473-93-0 (Druckausgabe)
URN urn:nbn:de:gbv:ilm1-2010000477

Titelfoto: photocase.com

# Acknowledgements

First of all I am grateful to Prof. Edgar Körner for giving me the unique opportunity to carry out my research at the Honda Research Institute. The friendly and open atmosphere at this institute is really inimitable.

I also want to express my sincere thanks to Prof. Horst-Michael Groß for his continuous, valuable constructive hints and his guidance throughout this thesis despite my geographical remoteness.

Warm thanks also to Prof. Bärbel Mertsching for her willingness, time and effort to review this thesis.

My work would not have developed to its current form without the supervision of Dr. Julian Eggert. I am very grateful for his patience and enthusiasm in the numerous scientific discussions and his support during my work. With his immense knowledge in the field he steered my research into a promising direction.

Furthermore, I want to thank all my colleagues for the great time I had during this thesis. Especially I am deeply indebted to Daniel Weiler, Nils Einecke, Volker Willert, Jochen Eppler and Andreas Richter for their help and endless fruitful discussions on both scientific and personal topics. With their humor and cooperativeness they enriched my everyday life. It was really a pleasure to work with all of them.

This thank is extended to Nevriye Memet for her friendly help on administrative tasks.

Computer simulations are of little value without a stable hard- and software infrastructure. Therefore I also wish to thank the administration team Burkhard Zittel, David Luttropp and Elsah Hrnjic for providing fast and uncomplicated technical assistance even though I am not always easy to please.

Last but not least I want to thank my parents, my sister and my better half Dorit for never giving up on me and motivating me to eventually finish this work. This would not have been possible without their support.

# Kurzfassung

Die Umgebung des Menschen ist voller visueller Details. Diese immense Menge an Information kann, unter der Annahme von begrenzten Verarbeitungs- und Speicherresourcen, nur teilweise aufgenommen und gespeichert werden. Daraus ergibt sich die Notwendigkeit einer selektiven Verarbeitung, die, je nach Aufgabenstellung, zu einer unterschiedlichen Repräsentation der visuellen Szene führt. Psychophysische Experimente zeigen, dass dabei die erfasste Umgebung nicht nur örtlich, sondern auch im Merkmalsraum selektiv bearbeitet wird, dass heißt es wird nur die visuelle Information aufgenommen, die für das Lösen der jeweiligen Aufgabe erforderlich ist.

Im Rahmen dieser Arbeit werden eine flexible Systemarchitektur und eine Kontrollstruktur zur aufgabenbezogenen Szenenrepräsentation vorgestellt. Im Gegensatz zu existierenden Arbeiten ermöglicht dieser Ansatz eine selektive Informationsaufnahme. Die vorgeschlagene Architektur enthält neben einem Lang- und Kurzzeitgedächtnis sowie einer Aufmerksamkeitskarte auch mehrere Verarbeitungsmodule zur Merkmalsextraktion. Diese Verarbeitungsmodule sind spezialisiert auf die Extraktion eines Merkmals und arbeiten unabhängig voneinander. Sie können jedoch je nach Aufgabenstellung dynamisch miteinander gekoppelt werden um gezielt die benötigte Information aus der Szene zu extrahieren. Die Entscheidung, welche Information benötigt wird und welche Module zur Extraktion dieser Merkmale gekoppelt werden müssen, trifft die im Rahmen der Arbeit entwickelte Kontrollstruktur, welche das gespeicherte Wissen des Systems und die gestellte Aufgabe berücksichtigt. Weiterhin stellt die Kontrollstruktur sicher, dass algo-

rithmische Abhängigkeiten zwischen den Verarbeitungsmodulen unter Zuhilfenahme von systemimmanentem Prozesswissen automatisch aufgelöst werden.

Die hier vorgestellte Systemarchitektur und die ebenfalls vorgeschlagene Kontrollstruktur werden experimentell anhand einer realen Tischszene evaluiert. Bei den durchgeführten Experimenten zeigt sich, dass bei Lösung einer gestellten Aufgabe die Menge der vom System verarbeiteten und gespeicherten Informationen deutlich reduziert wird. In der Folge werden die Anforderungen an die Verarbeitungs- und Speicherressourcen ebenfalls deutlich reduziert. Diese Arbeit leistet damit einen Beitrag zur aufgabenbezogenen Repräsentation von visuellen Szenen, da nur noch die Information verarbeitet und gespeichert wird, die tatsächlich zur Lösung der Aufgabe erforderlich ist.

# Abstract

The visual environment of humans is full of details. This incredible amount of data can neither be processed nor stored when assuming a limited computational power and memory capacity. Consequently, a selective processing is necessary, which leads to different representations of the same scene depending on the given task. Psychophysical experiments show that both the spatial domain as well as the feature domain are parsed selectively. In doing so, only those information are extracted from the visual scene that are required to solve a given task.

This thesis proposes a flexible system architecture along with a control mechanism that allows for a task-dependent representation of a visual scene. Contrary to existing approaches, the resulting system is able to acquire information selectively according to the demands of the given task. This system comprises both a short-term and a long-term memory, a spatial saliency algorithm and multiple visual processing modules used to extract visual properties of a focused object. At this, the different visual processing modules operate independently and are specialized in extracting only a single visual property. However, the dynamic coupling of multiple processing modules allows for the extraction of specific more complex features that are relevant for solving the given task. Here, the proposed control mechanism decides which properties need to be extracted and which processing modules should be coupled. This decision is based on the knowledge stored in the long-term memory of the system. Additionally, the control mechanism ensures that algorithmic dependencies between processing modules are resolved au-

tomatically, utilizing procedural knowledge which is also stored in the long-term memory.

A proof-of-concept system is implemented according to the system architecture and the control mechanism presented in this thesis. The experimental evaluation using a real-world table scene shows that, while solving the given task, the amount of data processed and stored by the system is considerably lower compared to processing regimes used in state-of-the-art systems. This in turn leads to a noticeable reduction of the computational load and memory demand. In doing so, the present thesis contributes to a task-dependent representation of visual scenes, because only those information are acquired and stored that are relevant for solving the given task.

# Contents

# 1. Introduction

The environment we are able to see, hear, smell, feel and taste is very complex and full of details. Consequently, the amount of information one could concentrate on is incredibly high. Neither organisms nor technical systems are provided with enough processing resources to handle such an amount of data, let alone the memory capacity large enough to store it. Thus it is crucial to separate the important information to process and store from those that is not important. According to the literature reviewed in section 2.1, this selection process is influenced by different factors like the available processing resources, the current need or task and the knowledge about the current scene and the world in general. Depending on those factors we are able to determine which parts, regions and details of the scene need to be processed and stored. Let us take a look at a fictitious scenario to illustrate this.

## 1.1. Fictitious Scenario

Assume we are in an indoor scene as shown in Fig. 1.1. How would the internal representation of this scene look like? As pointed out above, this strongly depends on the current task of the system. The following example illustrates the different resulting internal representations for two different tasks.

**Figure 1.1.:** An indoor scene full of potentially processable and memorizable details.

## A Cup of Coffee

Imagine the task is to get yourself a cup of coffee. As this task is too complex to solve it at once, it needs to be divided into simpler subtasks. First, the cup has to be found, followed by the coffee can and finally the coffee needs to be poured into the cup. Note that many of those subtasks involve *visual search processes*. In the initial scanning of the scene as shown in Fig. 1.2 a) there is no cup visible. However, the *long-term knowledge* tells us that cups can be found in cupboards. The first fixation is on the rightmost cupboard, after opening it, it turns out that no cups can be found. In the second cupboard glasses and cups are stored as shown in Fig. 1.2 b). We now grasp the first cup

we can reach and put it onto the table (see Fig. 1.2 c)). The next subtask is to find the coffee can and fill the coffee into the cup and so on. Apart from the long-term knowledge, we only need very limited

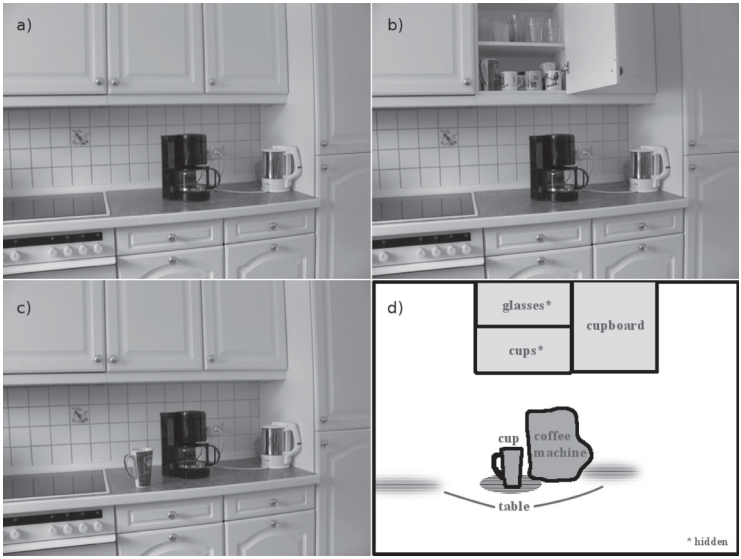


**Figure 1.2.:** The images a), b) and c) show a hypothetical image sequence involved in the cup task, leading to the scene representation in d). Note that only a few properties of the objects are saved like the coarse form and the location.

information about the scene and the objects contained in it to solve the given task (see Fig. 1.2 d)). Information like the form of the cup (for grasping), the location of the table and the location of the coffee can is relevant. Properties like color, texture or the exact classification of the different cups in the cupboard are irrelevant for the task and would only bloat the representation as they increase the memory demand and require additional computing resources. Keeping this in mind, we look at another task in the current scene.

## The "Linux cup"

Imagine that after finishing the previous task someone tells us to also bring the "Linux cup". From our long-term knowledge we know that this special cup is white and has the popular Tux penguin on it. Starting from the already acquired scene representation we check if the requested cup is already represented. To do so, we look at the cup we have taken out of the cupboard (see Fig. 1.3 a)). There is no need to search for this cup again, as our *short-term knowledge* about the current scene already contains the location of that cup. An examination of the cup's color reveals that this is not the requested one, so we once again open the cupboard in the middle (we again use our short-term knowledge). Given Fig. 1.3 a) it is the third cup from the left in the cupboard. We can now grasp the identified cup to solve the given task. As shown in Fig. 1.3 b) the scene representation has to contain many

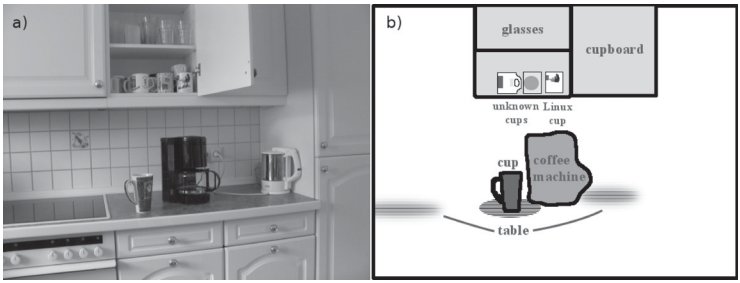


**Figure 1.3.:** Image a) shows the hypothetical image during the search for the "Linux cup" starting from the previous task. The resulting scene representation is now more detailed.

more details to solve the given task, because a special cup has to be identified. This requires further processing like precise segmentation, classification and so on. However, the system can still avoid to process *all* objects in the scene in such a great detail, as only cups are relevant for solving the task. It is sufficient to represent the table, the coffee can

and the cupboards very coarsely. Again, the system can save some processing time and memory capacity by selectively acquiring information about the scene compared to an extensive processing of all information potentially available in the scene.

Both examples show that only those properties of the scene need to be extracted that are relevant for the current task. By incorporating the given task in the process of building-up and maintaining the internal representation of a visual scene both the computational load and the memory demand can be reduced, as not all details of the scene and objects need to be stored. This way, the scene and the objects in this scene are represented as coarse as possible, but as fine-grained as required to solve the task. The outlined arguments that humans also process their environments selectively depending on the task will be supported by psychophysical evidence in chapter 2.

## 1.2. Claims

Current state-of-the-art systems, reviewed in chapter 2, only address the spatial aspect of selective processing. That is, they incorporate attention subsystems to perform visual search tasks, which select salient regions in the visual scene that probably contain good candidates for the object to be found. However, those state-of-the-art systems then statically process the attended object candidates by usually applying an attention-segmentation-classification cycle. There is no adaption of the processing to the given task and its needs, leading to a waste of processing time and memory capacity as also object properties irrelevant for the given task are computed and stored.

Contrary to this, the goal of this thesis is to model a selective processing regime in both the spatial *and* feature domain. To do so, the static attention-segmentation-classification cycle has to be split up to allow for a selective processing of object features. In this thesis a flexible and

dynamic system architecture is proposed, able to adapt to the level of detail required to solve the given task. This system is able to dynamically rearrange its processing pathways dependent on the current needs. It must be able to only process the color of an object in one case, while performing the whole segmentation-classification processing in another case dependent on the given task. It will be shown that even the processing of a rather simple static scene involves a lot of dynamical processing when it comes to selecting information from that visual scene.

To decide which information is relevant in the current scene a control mechanism is required that determines the objects, properties and locations that should be processed. Ideally the system should store as few information as possible, but still enough to solve the task. In this thesis, the usage of the system's long-term memory is proposed to identify the locations, the kind of objects and the properties of those objects that are potentially relevant with respect to a given task. It will be shown that, by using the proposed selective processing regime, both the computational effort and the memory load for the system's short-term memory are reduced.

As there is currently no similar approach a few fundamental questions need to be addressed during this thesis:

1. *What system architecture allows for flexible processing pathways?* A system architecture has to be found that allows for a dynamical combination of processing modules according to the task requirements. This involves the pipelining of processing steps and the selection of the level of detail for the processing modules themselves.

2. *How does the system account for its algorithmic needs?* The system has to make sure that during the combination and pipelining of processing modules all information and features for the follow-

ing module are available. As an example, the system needs to ensure that an object is detected *before* it can be segmented.

3. *How can a consistent representation of the system's knowledge about the world and the algorithmic needs be achieved?* The system must be able to represent knowledge about objects in the scene as well as procedural knowledge. This way the system knows how to measure properties of an object and how to modulate processing pathways consistently.

4. *How is the flexible processing controlled?* The acquisition process needs to be organized by a control mechanism that selects relevant objects, properties and locations which should be attended by the system in the current scene.

The main contributions of this thesis are the flexible and demand-driven system architecture as proposed in chapter 3 and its control mechanism and the consistent modeling of procedural and semantic knowledge in one relational long-term memory as proposed in chapter 4. It will be shown that contrary to state-of-the-art architectures, the proposed system architecture allows for a dynamic rearranging of its processing pathways according to the current demands of the system, while accounting for dependencies between the different processing modules. Furthermore, it will be shown that the control mechanism proposed in chapter 4 is able to select properties and locations of the visual scene according to the given task. Based on the information acquisition demand created by these properties and locations, the control mechanism infers which processing modules to use and organizes the acquisition process by combining these processing modules in a meaningful way. Here, the decisions of the control mechanism are based on the system's long-term memory. The architecture of such a long-term memory comprising the procedural and semantic knowledge of the system is given in chapter 4. To prove the concept of the proposed architecture and control mecha-

nism, an implementation of such a system is evaluated in the realistic scenario described in the next section.

## 1.3. Realistic Scenario

As the work done in this thesis concentrates on the architecture of the system and the control aspects of selective information processing, some simplifications on the scenario described in section 1.1 are required. This foremost concerns the complexity of the used visual algorithms and the scene settings. The following assumptions and constraints are made for the scenario used throughout this thesis:

- A visual scene is constructed by an arrangement of individual objects. Here, only the presence and the properties of an object are regarded, the spatial arrangement or parts of the objects are ignored.
- Very basic algorithms are used to extract visual properties of objects. However, whenever possible references to alternative and more elaborated algorithms are given. The simplification made on the visual algorithms also restricts the possible scenarios that can be dealt with. Therefore a new, more realistic scenario will be presented at the end of this section.
- To relax the constraint on the processing speed of the proof-of-concept implementation, only static scenes are evaluated. Nevertheless, some preliminary work has been done to also be able to process dynamic scenes. An outlook on this work and planned extensions to the system is given in chapter 6.
- The content of the system's long-term memory is taken for granted. It is not learned while the system is running, but is rather filled beforehand using a preprocessed clustering of the dataset. Even

though the online learning of objects and environments is an interesting aspect of such a system, the implementation of learning, consolidation and forgetting of memory contents would have exceeded the time frame of this thesis. They are therefore not part of this work.

- To avoid the problem of separating complex tasks into meaningful subtasks, only atomic visual search tasks are given that can be solved purely visually without the need of physical interaction with the scene. However, the given tasks involve different levels of detail like searching for an object with a certain color or searching for a specific object in the scene. It will be shown in chapter 5 that the different requirements of these tasks lead to different levels of detail in the scene representation.

To account for the restrictions made, a reduced scenario is used to evaluate the system proposed here. The visual scene that is used throughout this thesis is shown in Fig. 1.4. According to the restrictions and simplifications mentioned above, simple search tasks with different levels of details are given to the system, like "find a red object" or "find the Asimo" in the scene above. For each of these tasks different levels of detail in the scene representation are required. State-of-the-art systems would basically solve a given task statically and independent of the required level of detail, by searching for an object candidate, segmenting the object and measuring *all* available visual properties of a candidate, including computationally demanding calculations like the identification of the object. In the example of the task "find a red object" this would mean that state-of-the-art systems calculate *all* properties of the object, while only the color of an object candidate is relevant for the task. Contrary to this static processing, the system proposed in this thesis would perform the following steps for the task "find a red object":

1. The control mechanism determines which properties of objects in the scene need to be measured by activating the requested prop-

**Figure 1.4.:** This real-world scene is used throughout the thesis in a reduced scenario to evaluate the system proposed here.

erties in the long-term memory. In this example, the requested property "red" would activate the color measurement pathway. This activation would cause a *demand* in the system for measuring the color of an object candidate.

2. To measure the color of an object candidate, first a suitable candidate needs to be located in the scene. This is an example of a dependency between measurement processes, as the color measurement depends on the location of an object among others. In the proposed system, the control mechanism recursively resolves such dependencies utilizing the procedural knowledge the system has about its own sensory processes. These processes are not

hard-coded, like in state-of-the-art systems, but are rather *dynamically* allocated and combined by the system on demand.

3. After determining the order in which the dependencies need to be resolved, the control mechanism infers which visual routine it has to trigger to, e.g. locate an object, segment it or measure its color. This information is also stored in the procedural memory of the system. Here, the system itself creates a demand by in fact splitting up the measurement process into atomic steps.

4. Knowing the exact sequence of the measurements and the visual routines to use in each step, the control mechanism triggers the requested visual routines sequentially and modulates them if applicable. In the given example of finding a red object, an object candidate will first be located by modulating and triggering the saliency routine. In a second step a coarse mask of the located object candidate will be calculated by triggering the segmentation routine. The calculated mask will be used in a final step to determine the initially requested color of the object candidate by triggering the color extraction routine.

5. In each step, the acquired information about the properties of the object candidate is stored in the system's memory. As soon as there is enough evidence that the attended object candidate actually matches the searched object, the process is stopped and the task is regarded as solved. If there is not enough evidence, the next property of the searched object is measured starting with step 2 and so on. In this example, measuring the color of an object candidate is sufficient to decide if the candidate matches the searched object. However, for more complicated tasks like "find the Asimo" properties need to be determined that identify the searched object among other objects. This step of determining discriminative properties is described in chapter 4.

The processing steps described here will construct a representation of the current scene by sequentially acquiring more information about the objects in the scene as long as a demand exists. Contrary to state-of-the-art systems, the sequence of measurements required to solve a given task is constructed dynamically by the system on demand. This demand is initially triggered by the task given to the system, but is also generated by the system itself when splitting the given task into measurement sequences. To be able to perform this sequential construction of the scene, the flexible system architecture proposed in chapter 3 is required. The organization of the measurement processes is then done by the control mechanism under heavy use of the system's long-term memory. This long-term memory not only contains the semantic knowledge of the system, but also represents procedural information like dependencies between visual routines. It is shown in chapter 5 that the proposed overall system is able to construct such task-related representations in a demand-driven way.

## 1.4. Summary

The review of psychophysical literature in chapter 2 suggests that information contained in a visual scene is selectively processed both in the spatial and feature domain. That is, only the information (regions and object properties) necessary to solve the current task is selectively extracted. Other details of the scene are ignored, leading to a change blindness in these regions and features. Inspired by the biological evidence, state-of-the-art systems are reviewed in chapter 2. Most of these state-of-the-art systems implement an attention mechanism that selects regions in the scene which are processed afterwards. While implementing this spatial selection mechanisms, all those state-of-the-art systems lack a selective processing of information in the feature domain. They rather apply a fixed and static processing regime like the attention-segmentation-classification cycle. This leads to a waste of processing

power and memory capacity, as information irrelevant for the given task is processed.

To overcome this limitation and to allow a more efficient processing in terms of time and memory capacity, a selective, task-oriented and demand driven system-architecture is proposed in chapter 3 of this thesis. To do so, the processing pathway is split into multiple visual routines that are specialized to extract one property of an object each. Furthermore, a novel control mechanism is proposed in chapter 4 utilizing the (predefined) long-term knowledge of the system about known objects to identify object properties that are relevant for the given task. This long-term knowledge not only comprises the knowledge about objects, but also represents procedural knowledge about the system itself like which visual routines need to be triggered for measuring certain properties of objects and which visual routines depend on each other. By incorporating this knowledge, the different visual routines are combined to extract the selected properties, while handling possible dependencies between the visual routines. Basic memory structures and assumptions made on the memory content are also discussed in chapter 4. The proposed system architecture together with the scheduling mechanism are the main contributions of this thesis. To evaluate the performance of such a system, experiments are conducted using a proof-of-concept implementation. The results presented in chapter 5 show that the proposed dynamical and flexibly organized processing regime leads to a more efficient system with respect to computational and storage demand. The experiments show that by selecting the information that has to be extracted from the scene, the number of processing steps and the number of stored object properties are dramatically reduced in most of the cases. Eventually, important questions that provide the basis for further research on the topic of task-oriented processing and representation of scenes are given in chapter 6.

# 2. Related Work

As described in the previous chapter one goal of this work is to create a vision system architecture that provides the means to build-up a sparse, flexible and demand-driven scene representation. Much can be learned from the most advanced vision system available, the human. Therefore, the psychophysical literature will be reviewed in the first part of this chapter, trying to identify important properties a flexible and demand-driven scene representation must have. In the second part of this chapter, state-of-the-art vision systems will be reviewed.

## 2.1. Experimental Evidence

Humans feature the probably most advanced vision system available. To get a glimpse on how this system represents its detailed and rich visual environment, some psychophysical experiments found in literature should be reviewed.

The question of what we represent internally is closely related to the question of what we look at. With his direct and non-invasive experiments on eye-movements Buswell [Buswell 1935] laid the foundation to one of the most extensively used methods in psychophysical experiments today. Since then, numerous eye-tracking experiments have been performed, trying to shed light on the question "where do we look". With the data from such experiments a first model was proposed, suggesting a visual buffer that integrates different views of a scene [Mc-

Conkie and Rayner 1976, Jonides et al. 1982] into a detailed internal representation [Marr 1982]. In [O'Regan and Lévy-Schoen 1983] first doubts on the integration idea emerged, which were amplified by the so called "change blindness" experiments [Bridgeman et al. 1975, Pashler 1988, Simons and Levin 1997, Rensink et al. 2000, Simons 2000] (see [Simons and Ambinder 2005] for a recent review). In these experiments, changes in the scene made during saccades went unnoticed by subjects. A schematic illustration can be found in Fig. 2.1. To explain this effect,


**Figure 2.1.:** Here a schematic illustration of a typical "change blindness" experiment is shown. While the subjects (possibly provided with a certain task) perform a saccade, parts of the scene are modified. These modifications may comprise deletions, displacements, modification of features and others.

a new model was created where the scene representation is thought to be abstract and schematic [Irwin 1992, Intraub 1997]. Furthermore, the representation is not dense but rather sparse [O'Regan 1992, Irwin and Andrews 1996, Hayhoe et al. 1998], which seems counter-intuitive at a first glance, because to us our representation of the world seems detailed and dense. However, to get and store a detailed view of an object it is necessary to focus attention on this object [Treisman and Gormican 1988, Intraub 1997]. Representations of unattended objects are simply overwritten [Rensink et al. 1997]. This new view on the nature of internal representations poses new important questions. Which locations do we attend and what guides our attention there? Which information about the attended location is stored? Which mechanisms are involved in attention? Even though some aspects might overlap, the literature is grouped into two categories. The first category deals

with the spatial aspects of attention (where do we look), the second with the question what details are stored about the locations attended (what do we look at).

## Spatial Aspects of Attention

When it comes to the question which locations in a scene are attended, one of the most cited experiments is the one of Yarbus [Yarbus 1967]. In his experiments he monitored the eye movements of humans for different tasks. The photo used and the scan paths for two different tasks are shown in Fig. 2.2. The scan path of the subjects are clearly differ-

**Figure 2.2.:** Yarbus used the image shown on the left and provided the subjects with different tasks. One task was to estimate the material circumstances of the shown family (scan path in the middle). Another was to estimate the age of the different people (scan path on the right). All pictures are taken from [Yarbus 1967].

ent for the two selected tasks. As Yarbus wrote *"... the distribution of the points of fixation on an object, the order in which the observer's attention moves from one point of fixation to another, the duration of the fixations, the distinctive cyclic pattern of examination, and so on are determined by the nature of the object and the problem facing the observer at the moment of perception."* [Yarbus 1967, p. 196]. The observation that the current task influences the way the subject looks at a scene was confirmed by other experimenters like [Just and Carpenter

1976, Rensink et al. 1997, Rothkopf et al. 2007]. This suggests that the build-up of the internal representation is to a high degree task-driven [Hayhoe et al. 1998, Rothkopf et al. 2007].

Beside the influence of the current task on controlling attention, there is evidence that also past experiences influence attention. In his experiment Just concluded that there are *"...two possible sources of such information [guiding the eye-movement], namely, the task structure and information computed during the trial."* [Just and Carpenter 1976, p. 475]. This modulatory influence of the short-term memory was confirmed in further experiments like [Chun and Nakayama 2000, Soto et al. 2008]. Additional to the short-term memory semantic (or long-term) memory also plays a role in attention control [Henderson et al. 1999]. At this, objects that do not fit semantically into the scene are not salient per se [Henderson et al. 1999], but rather get fixated more often and prolonged. However, *"search paths to a specified object tend to be shorter to objects that are consistent with the scene ..."* [Henderson et al. 1999, p. 226].

To summarize, both short- and long-term memory bias the attention on objects in the current scene. Furthermore, regions we attend in a visual scene are influenced by the task we currently have. But what is stored about the attended location?

## Featural Aspects of Attention

More recent experiments give hints on what details are actually stored about the visual scene. There seems to be a consensus in literature that not only the spatial representation of the scene, but also the representation of object features is sparse. However, different views do exist on how detailed and comprising this representation is. A very extreme view is that the world itself serves as an external memory and that there is no internal representation at all [Gibson 1966, MacKay

1973, O'Regan 1992]. It would be hard to explain recalls from the short-term memory using this theory. A more moderate theory proposes that around three to four "object files", indexing objects in the world, are stored internally [Kahneman et al. 1992, Irwin 1992]. Rensink proposes an additional gist to be stored [Rensink 2000]. However, contradictory to those views Henderson found a rather detailed encoding of saccade targets [Henderson and Hollingworth 2003]. Additionally, he reported in a later experiment that subjects could even remember the orientation of objects in a photo over 24 hours [Hollingworth 2005]. This suggests that a more detailed internal representation exists.

The work of Hayhoe, Ballard and colleagues provide a deeper insight into the question of what is actually represented internally. In an ex-

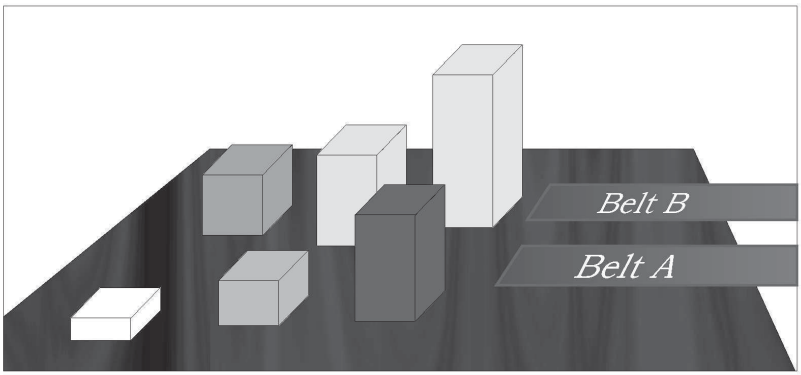


**Figure 2.3.:** Subjects were asked to perform three different tasks in such a virtual environment, while the size of the blocks was changed. All tasks involved moving the blocks to the belts. However, only when the size was relevant for the task, the changes were noticed. Illustration after [Triesch et al. 2003].

periment by Triesch [Triesch et al. 2003] subjects were asked to perform different tasks, while the size of objects in the scene was changed. The experimental setup is depicted in Fig. 2.3. The subjects were asked to

1. pick up the blocks in front to back order and put them on belt A
2. first pick up the tall blocks, then the small blocks and put them on belt A
3. first pick up the tall blocks and put them on belt A, then pick up the small blocks and put them on belt B

and report any suspicious events they noticed during the experiment. Even though the conditions for changing the block size were equal for all three tasks, subjects noticed changes more often only in the third case. This third task was the only one where the size was task-relevant before *and* after the change. Some changes even went unnoticed while the subject was tracking the block in question with his/her eyes. Triesch concluded that *"most information may be computed 'on demand' by engaging specialized functional routines at just the right times"* [Triesch et al. 2003, p. 92]. Furthermore, this experiment suggests that object properties are only stored and maintained as long as they are relevant for the current task. This finding is also supported by another

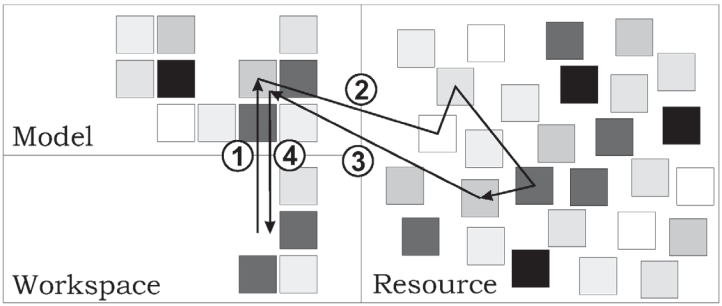


**Figure 2.4.:** In this pattern-copy-task, subjects were asked to replicate the pattern of colored blocks shown on a display. While doing so, the eye and hand movement were recorded. The most prominent eye movement pattern is shown here. Illustration after [Ballard et al. 1995].

experiment [Ballard et al. 1995]. The most prominent eye movement pattern of this experiment is shown in Fig. 2.4. As you can see, the eyes first move from the workspace to the monitor displaying the model, probably acquiring the color of the next block, then to the resource area for finding a block of the required color. Afterwards, the eyes move back to the monitor to probably acquire the location of the block picked up before. This movement is surprising, as the position could have been stored in the first fixation. Ballard concluded that *... the information required for the task is acquired just prior to its use."* [Ballard et al. 1995, p. 76]. This result was confirmed in [Hayhoe et al. 1998].

Starting from these experiments on what is represented (both in terms of location and properties) internally, computational state-of-the-art models are reviewed trying to explain one or more of the properties found. In the next section, the focus is on systems dealing with the internal representation of a visual scene using attention mechanisms.

## 2.2. State-of-the-art in Architectures

One of the first real-time applications using a visual scene representation is VITRA [Herzog and Wazinski 1994]. This system visually processes an ongoing soccer game and generates verbal comments in real-time. Even though the results of this system are very impressive, it is restricted to its special task and domain. That is, because VITRA monitors all objects and actions simultaneously, making the approach unusable in less constraint environments. As the psychophysical experiments suggest, a fast and efficient way of finding objects in a visual input is a key aspect for representing scenes. However, unbound and data-driven visual search approaches are NP-Complete [Tsotsos 1992], rendering them infeasible. Therefore, bottom-up approaches are not subject of this chapter, rather those state-of-the-art system architectures subscribing to a top-down guided schema will be reviewed. These

approaches can be grouped by their processing principles into four basic architectures as shown in Fig. 2.5. In the following, the active vision

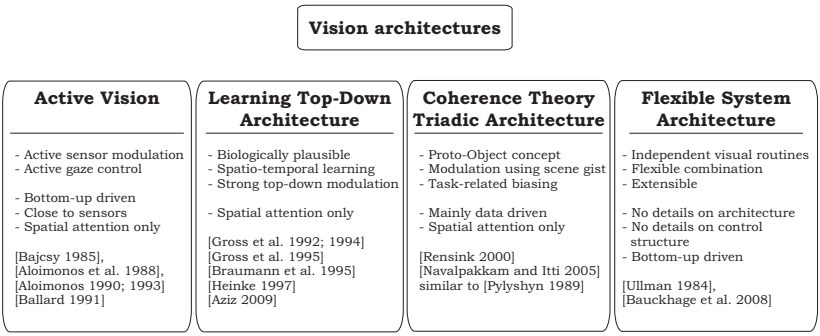


**Figure 2.5.:** State-of-the-art system architectures can be grouped into four different categories. Important characteristic facts are listed for each of the categories.

approaches, learning top-down approaches, the coherence theory approaches and flexible system architectures will be described in more detail. Their relation to this thesis will be discussed in section 2.3.

## Active Vision Architecture

Contrary to the idea of vision as a passive reconstructive process, active vision systems try to model vision as an active process. That is, they actively modify parameters of the (possibly passive) sensors at hand, in order to explore and disambiguate their visual input. The term *active vision* was first coined by Aloimonos [Aloimonos et al. 1988], but the basic idea ranges back to the work of Bajcsy on *active perception* [Bajcsy 1985]. She described active vision (or in this case active perception) as *"... an application of intelligent control theory which includes reasoning, decision making, and control."* [Bajcsy 1988, p. 996]. Subsequent works on *purposive vision* [Aloimonos 1990; 1993] and *animate*

*vision* [Ballard 1991] constitute additional variants of active vision architectures. A schematic illustration of an active vision architecture can be found in Fig. 2.6. As can be seen, the processing modules are

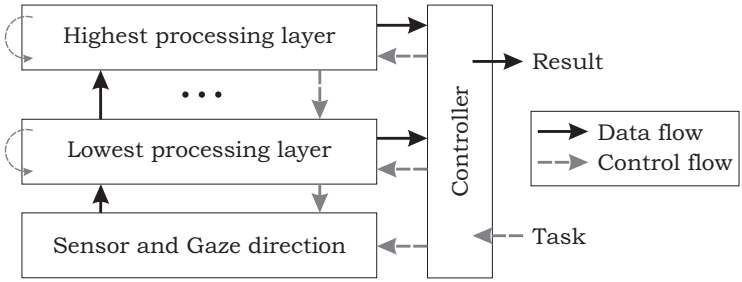


**Figure 2.6.:** Active vision is thought of as a data-driven architecture, where an intelligent controller uses information about the input and task to modulate parameters of the processing modules. More importantly, the controller also guides attention spatially in later approaches.

modulated by a controller in a top-down manner. This controller uses information about the current scene and a given task to determine feedback parameters like focus, zoom, aperture, gain, gaze direction and others. While Bajcsy mainly focused on the control of sensor parameters, in later work the ability to control the gaze direction and thus spatial attention is emphasized [Ballard 1991]. This led to the idea of representing the visual scene [Aloimonos 1993] only partially, using a sparse spatial memory [Ballard 1991]. It was shown that using this kind of representations together with the active sensing of the input, the complexity of some tasks involving visual ambiguities can be reduced [Ballard 1991].

To summarize it can be said that active vision systems modulate sensor parameters in a top-down manner, influenced by the visual input and the current task. The most important parameter is the gaze direction which can be interpreted as spatial attention. This spatial attention

is then used to create a partial representation of the scene. All active vision approaches exhibit a strong relation of the control process to physical actions like changing the position to ease the segmentation of an object.

## Learning Top-Down Architecture

The neural network architecture proposed in [Gross et al. 1992; 1994] aims at presenting a biologically plausible system architecture to incorporate knowledge into the process of visual attention. Figure 2.7

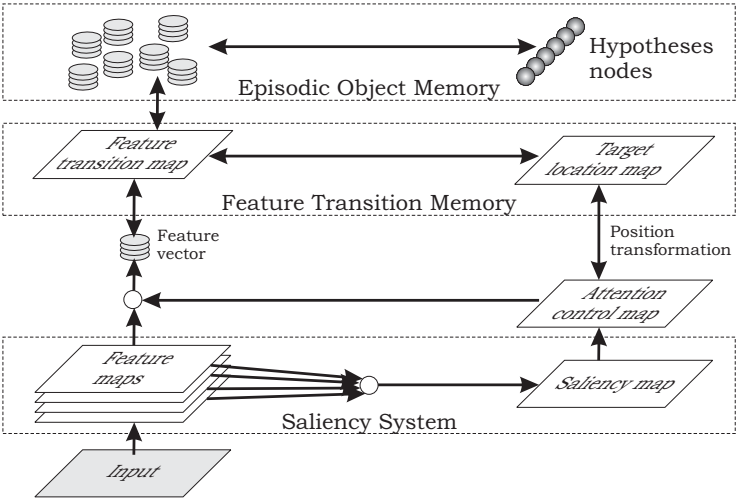


**Figure 2.7.:** This system architecture is biologically inspired and models top-down and bottom-up processes of spatial visual attention. Illustration after [Gross et al. 1995].

shows an illustration of the architecture. The basic processing principle is to modulate the underlying spatial saliency computation using

learned spatio-temporal combinations. First, visual features like color, texture and luminance are extracted. Using these features a saliency map is computed. The resulting saliency map then modulates the Attentional Control Map together with top-down provided and learned spatio-temporal information. By doing so, positions in the input image can be preferred or neglected according to the internal hypothesis of the system. In the next processing step, the Attentional Focus Identifier selects a location based on the Attentional Control Map and compares the features measured at this location with the expected features provided in a top-down fashion. The result of this comparison then serves as a reinforcement signal to improve the spatio-temporal modulation as well as the feature prediction for a given object. As a result of the learning process, the spatio-temporal pattern of the scanning path changes dramatically and shows a more goal directed behavior [Braumann et al. 1995, Gross et al. 1995]. A more detailed description of the learning of the feature-space-time relations is given in [Heinke and Gross 1993, Heinke 1997]. The work presented in [Aziz 2009] and [Hamker 2007] pursue a similar approach, trying to model the task influence on the saliency map. To do so, a combination of top-down influences and bottom-up saliency maps is used.

To summarize, the architecture proposed here incorporates learned spatio-temporal patterns as well as feature combinations to guide the spatial attention control process. Furthermore, the difference between the expected features and measured features is used to learn and improve the feature-space-time relations. However, the prediction and measurement processes comprise *all* features available to the system regardless of the given task.

## Coherence Theory and Triadic Architecture

Another approach on representing visual scenes is based on Rensink's work on dynamic scenes [Rensink 2000]. In this work, he proposes

the so called coherence theory and presents a tripartite system architecture. This system architecture comprises a non-attentional scene schema, an attention control part and a low-level vision part as illustrated in Fig. 2.8. The scene schema is computed by combining the gist

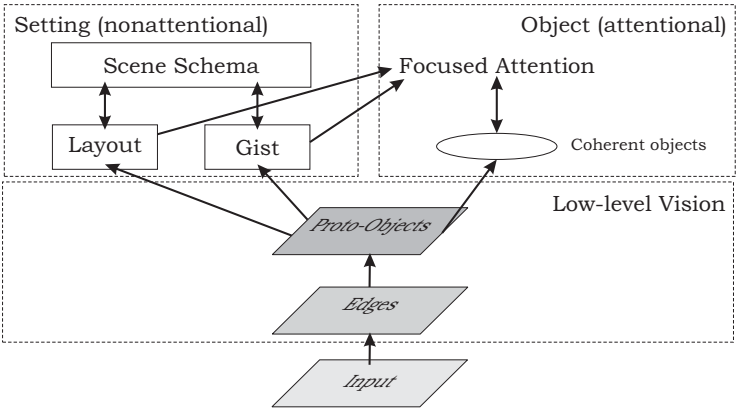


**Figure 2.8.:** In the triadic system architecture as shown above, the scene schema influences the attention guidance. The schema itself is composed of a coarse scene layout and a scene context or gist, based on low-level statistical processes. Illustration after [Rensink 2000].

and a coarse layout of the scene retrieved from a long-term memory. Using this scene schema, the visual attention is then guided spatially. The attentional part of the architecture is based on the coherence theory which is in parts similar to FINST [Pylyshyn 1989]. Here, volatile prototypical objects, called proto-objects, are formed in the low-level vision part in parallel and are then potentially instantiated by the attention process. When a proto-object is attended, a coherent object forms, combining *all* visual properties of the proto-object. As soon as attention is removed from a coherent object, it vanishes and degenerates to a proto-object. Additionally, a two-level hierarchy is proposed in [Rensink 2000] to represent coherent objects (upper level) as a com-

position of parts (lower level). A more elaborated and detailed model based on the triadic architecture is proposed in [Navalpakkam and Itti 2005]. In this model, a detailed view on how the scene context could influence the attention process is provided. There, the modulation of the low-level features combined with a so called task-relevance map plays an important role. This task-relevance map is computed by integrating the spatial bias of the gist and the coarse layout of a scene. The coarse scene layout itself might also be influenced by the working memory. This way locations relevant for a given task are preferred. Furthermore, the task relevance map is used to bias low-level features. Together with a learned feature weighting retrieved from the long-term memory, a saliency map is constructed using the method presented in [Itti and Koch 2000; 2001]. This biasing of features and locations enhances the probability that the system's attention is focused on a task-relevant location in the scene.

To summarize, the triadic architecture is based on the coherence theory, stating that volatile proto-objects are formed rapidly and in parallel. As soon as attention is focused on one of the proto-objects a coherent object is formed that combines all visual features. The attention process itself is guided by a spatial and feature bias incorporating the task, the scene context and a coarse spatial layout of the scene. A combination of short- and long-term memory is used to store a few objects as well as saliency weights learned for an object. Tasks are then processed by retrieving and using those weights to bias the low-level visual processing. Attended objects in the scene are then stored together with *all* their features in the short-term memory.

## Flexible System Architecture

In [Ullman 1984] the usage of elementary processing modules, called visual routines, is proposed to solve visual tasks. Even though there is no explicit system architecture contrary to this thesis, the author

implies a flexible system architecture that allows for a combination of multiple elementary visual routines to solve more complex tasks. A solution on how such a flexible combination of the visual routines can be controlled is not given. Nevertheless, the idea of flexibly concatenating elementary operations is appealing and provides the basis for the system architecture proposed in this thesis.

Recently, a novel system architecture was proposed that also contains elements of a flexible organization of the system's processing pathways. The architecture is called visual active memory and is proposed in [Bauckhage et al. 2008]. An illustration of the system architecture is shown in Fig. 2.9. The system architecture is centered around the vi-

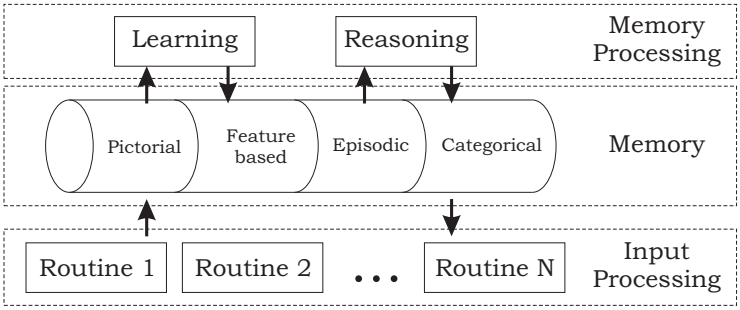


**Figure 2.9.:** This system architecture is centered around the visual active memory. Different visual processing algorithms are loosely coupled to the memory to maintain a flexible and modular structure. Illustration after [Bauckhage et al. 2008].

sual active memory component shown in the middle row of Fig. 2.9. Different processing modules are flexibly and loosely coupled to this memory to ease the integration of additional modules. According to [Bauckhage et al. 2008], the processing in this framework is driven in a bottom-up fashion without top-down control. However, the bottom-up information triggers the different processing modules working on the memory. These processing modules then consolidate memory content

by fusing bottom-up information with contextual results. To do so, all information is represented in memory as hypotheses with an uncertainty. Unfortunately, the attention mechanism is not described in detail. Nevertheless, the paper states that two different approaches are used. First, there is a bottom-up driven saliency map to extract interesting regions in the image and second, interesting regions can be defined by using human-machine interaction like pointing. Although the approach has many interesting properties like the human-machine interaction, an extensible interface and memory consolidation processes, the information is not selectively processed in this framework. That is, the system has to handle *all* sensory information passed from lower vision processes, no matter if it is relevant for the current task or not. In the example system shown in [Bauckhage et al. 2008] classifiers were used to reduce the amount of bottom-up data. However, one has to investigate if the proposed principle still holds when a larger amount of the bottom-up data is passed to the system.

## 2.3. Discussion

In the first part of this chapter psychophysical literature concerned with visual attention was reviewed. It has been shown that the perception of the visual vicinity is, contrary to our subjective experience, both sparse in the location *and* sparse in the feature space. The selection of the attended locations and features is influenced by the task, the information already acquired about the current scene and the knowledge about the world and its structure. As Triesch put it: *"What you see is what you need."* [Triesch et al. 2003]. The experiments discussed here suggest that attention is guided sequentially and is strongly influenced by top-down processes. Furthermore, vision is thought to be an active process [Tsotsos 1992, Hayhoe 2000] that aims at acquiring those information necessary to solve a given task on demand.

In the second part of this chapter state-of-the-art systems for scene representation were reviewed. While the active vision approaches model top-down processes to ease the perception of objects or the scene, they mostly focus on adaptation of sensor parameters. In later models like [Ballard 1991] attention is used to select regions in the scene to focus on. However, in most cases these systems are data-driven (bottom-up) and only use top-down modulation to guide spatial attention faster. To do so, in [Navalpakkam and Itti 2005] properties stored in a long-term memory are utilized to bias location with properties similar to the searched ones. Furthermore, Navalpakkam et al. model the influence of the current task on spatial attention using a task-relevance map. A more advanced spatio-temporal modulation of the attention process can be found e.g. in [Gross et al. 1995]. While this incorporation of knowledge about the objects, spatio-temporal relations and the task leads to a huge improvement in the search speed, the influence of the task on attended features is completely neglected. That is, once all those state-of-the-art models select a certain location to focus on, the whole feature vector of that location is stored in memory and processed further regardless of the given task. By doing so, these systems neglect potential savings introduced by the task in both the amount of processing and the used memory capacity. For example, if the task only requires to determine the color of an object, these models nevertheless will run a classifier and store a full-fledged representation of the object as they are built on static processing pathways. In such a processing paradigm, the information are processed in pipes from the image pixel information up to e.g. an object ID.

To model aspects of attention also in the feature domain, a more flexible processing schema is required. Very early ideas on such a flexible architecture can be found in [Ullman 1984]. However, a concrete architecture or a control mechanism required in such a system are not proposed there. In the next chapter a system architecture will be presented that is based on such a flexible processing paradigm and is capable of creating processing pathways on demand. In the following it will

be shown that the proposed system architecture can be used to implement a system which actively focuses its attention on certain properties of an object to solve given search tasks. By doing so, the fundamental question posed by [Hayhoe 2000] on how to schedule those information acquisition processes is tackled.

# 3. System Architecture and Components



The goal of this thesis is to model the selection of information in both the spatial *and* feature domain, dependent on the current task and knowledge the system has. To achieve this goal, a very flexible and dynamic processing paradigm is required, which aims at processing and storing only those information required to solve the system's task. In chapter 1 a number of fundamental questions were formulated. This chapter deals with the first of those questions by proposing a novel system architecture. This system architecture allows for the flexible and dynamic processing paradigm mentioned above and satisfies the special requirements such a paradigm poses. In the first section, these requirements are discussed and the system architecture itself is presented. Afterwards, the processing flow is illustrated using the realistic scenario given in chapter 1. Important components of the system are discussed in detail. Finally, references to related work concerning the different components of the system are given and concrete realizations of the components used in the proof-of-concept implementation of the system are presented. Some aspects of the system architecture proposed in this chapter were already presented in [Eggert et al. 2007] and [Rebhan et al. 2008].

## 3.1. Requirements

A dynamic and flexible processing paradigm puts special constraints on the system's architecture. This ranges from the selective and specialized processing of information in lower levels up to the way of storing the acquired information. Those requirements with respect to the system architecture are mentioned in this section. A more comprehensive overview and discussion about the general challenges that cognitive vision systems face can be found in [Eggert and Wersing 2008].

As stated before, one goal of this thesis is to provide a system architecture that lays the foundation for a selective acquisition of information in both the spatial and feature domain. Contrary to state-of-the-art systems, the selective acquisition and storage of information requires the dynamical and flexible combination of different processing pathways on demand. The goal is to process and store only those information necessary to solve the current task. To account for that requirement, it must be possible to run different visual algorithms independently. Nevertheless, the algorithms should provide the means to combine their information whenever necessary. Early, in [Ullman 1984], the usage of so called *visual routines* was proposed. These visual routines are highly specialized processing modules that only extract elemental visual features of the given image. A combination of multiple visual routines can be used to solve more complex tasks. In [Ullman 1984] also the combination or assembly of those routines is briefly mentioned and the storage of a set of useful combinations is proposed. The system architecture presented here also subscribes to the specialized and independent visual routine concept. However, contrary to the fixed set of stored visual routine combinations, the goal of this thesis is to create visual routine combinations on-the-fly according to the task and the processing resources the system has.

Based on the ability of flexibly combining different visual routines, an algorithm is required that selects the visual routines to combine and

control the temporal orders of their execution. This *control mechanism* selects the regions and the feature information that should be attended. In addition, it should incorporate the given task and the knowledge the system already has about the current scene and the world. A lot of work has been done on the selection of image locations using *saliency maps* (see e.g. [Hamker 1998, Itti and Koch 2000, Frintrop et al. 2005, Navalpakkam and Itti 2007, Michalke et al. 2008, Aziz and Mertsching 2008]). However, the question of what feature information to attend is not well researched. In chapter 4, a control mechanism will be presented that takes both aspects of attention into account. In the following this control mechanism will be called *attention control*.

The attention control needs access to the knowledge of the system. Hence, a memory storing that knowledge has to exist in the system. This memory must be able to represent the information in a way that eases the work of the attention control. To adhere to the psychophysical studies presented in the last chapter, the memory is divided into a *short-term memory* and a *long-term memory*. The short-term memory stores objects seen in the current scene, whereas the long-term memory stores the knowledge gathered about all objects seen in the world. One can interpret this separation as a separation into prototypical objects and instances of objects. This topic will be discussed in chapter 4.

Another very important requirement for system architectures is the extensibility. That is the ability to start with a relatively simple system and to make it more and more complex later. By doing so, the system stays manageable and can incrementally address problems, rather than face all of them from the beginning. To create an extensible system, defined interfaces are necessary as discussed for example in [Bauckhage et al. 2008]. In the system presented here, the main interfaces reside between the image features, the visual routines and the memory. By keeping the set of image features static, the definition of interfaces between the features and the visual routines is trivial. However, to combine information from different visual routines, an interface between

the system's memory and the routines needs to be defined. This interface must allow both to deliver information from the visual routines to the memory and to *top-down feed back* information from the memory to the visual routines.

To summarize, the system architecture must fulfill the following requirements:

- Different specialized visual routines can be combined on demand. These routines compute a certain property in the visual input and deliver the result to the memory. They are able to incorporate top-down feedback.
- An attention control mechanism exists that mediates the order and execution of the different visual routines to acquire those information necessary to solve the given task. Thereby, the task itself and the knowledge of the system are used to select regions and properties attended in the visual scene.
- A memory architecture is required that provides the knowledge of the system in a suitable way to the attention control mechanism. The memory itself is split into a short-term and a long-term part to adhere to psychophysical findings.

In the next section a system architecture is proposed that fulfills the requirements discussed here.

## 3.2. Architecture

The system architecture proposed in this thesis comprises three major sections as shown in Fig. 3.1. The relational memory includes both the short-term and long-term memory (see (1) in Fig. 3.1). The short-term

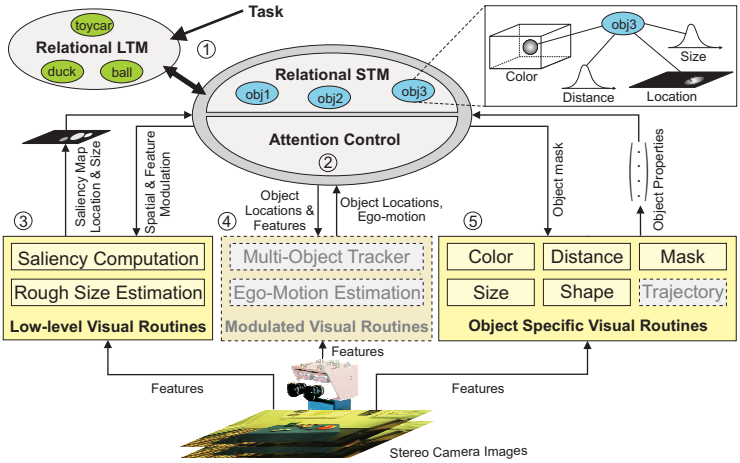


**Figure 3.1.:** The system architecture comprises 1) a relational short- and long-term memory, 2) an attention control mechanism coordinating the processing and information flow between visual routines (3–5), 3) object unspecific visual routines like saliency computation, 4) multi-object visual routines like multi-object tracking and 5) object specific visual routines like segmentation and property measurements.

memory stores information about the current scene, whereas the long-term memory represents the system's knowledge about the world. Both use a relational graph structure to represent relations between objects and properties. Furthermore, the long-term memory stored information about the measurement processes of object properties. More details can be found in chapter 4.

Based on the memory and the given task, the attention control mechanism (see (2) in Fig. 3.1) determines which regions and features will be attended in the scene. To do so, the attention control can modulate features in a saliency map or in visual routines to find new object candidates or measure object properties respectively. Furthermore, it is responsible for scheduling the internal measurement processes and thus

the visual routines of the system. Please refer to chapter 4 for more details. Once a property is selected for measurement, the corresponding visual routine is triggered.

The visual routines and the saliency map are located in the middle layer of the architecture. The saliency map (see (3) in Fig. 3.1) is dedicated to find object candidates in the visual scene that best match the top-down expectations raised by the memory and the attention control mechanism. The visual routines marked with (4) and (5) in Fig. 3.1 are specialized to compute *one* selected visual property of an object each, similar to the concept proposed in [Ullman 1984]. Separating the processing into different highly specialized modules allows an easy extension of the system and a free combination of those modules in later stages. Furthermore, it becomes possible to run only those visual routines required to gather the information relevant for the task. One can arrange the visual routines into three different (partly overlapping) groups. On the left-hand side there are object unspecific routines like the saliency computation (see (3) in Fig. 3.1). Those routines are bottom-up driven and only influenced by modulatory top-down inputs. The unspecific processing allows parallel computation with respect to the image location. Routines of the second group (see (4) in Fig. 3.1), work in parallel on a few objects [Luck and Vogel 1997]. However, they are tuned to those objects by top-down information like object features or locations. An example for such a routine is a multi-object tracker. Currently, this class of visual routines is not implemented (grayed out box in Fig. 3.1) in the system. However, in [Eggert et al. 2007] that kind of visual routines have been tested separately in a proof-of-concept system. On the right-hand side of the middle layer (see (5) in Fig. 3.1), object specific visual routines are grouped. Those algorithms work sequentially on only one object at a time. The processing itself is very selective and triggered in a top-down manner, including a strong top-down bias. Examples are classification or segmentation algorithms that include for example object biasing and form priors.

The raw camera images are preprocessed in the lower layer to provide different feature channels (see bottom of Fig. 3.1). In the next section, we take a closer look at the preprocessing and different components of the lower and middle layer. References to alternative state-of-the-art algorithms for the different components are given. Furthermore, details for the algorithms are provided which are used in the proof-of-concept system to generate the results shown in chapter 5. The memory and attention control mechanism of the upper layer are presented in chapter 4.



## 3.3. Processing Flow Example

In this section the processing flow in the system is shown using the exemplary task "find a red object" as described in section 1.3. Figure 3.2 shows again the realistic scenario used throughout this thesis that was already presented in chapter 1. To initiate the search of the requested red object, the task-related properties are activated in the system's long-term memory (see (1) in Fig. 3.1). The attention control mechanism (see (2) in Fig. 3.1) now tries to determine which properties have to be measured in the current visual scene to solve the task. In this case, the attention control mechanism needs to trigger the measurement of the object color to solve the task. Furthermore, the attention control mechanism needs to resolve the elemental processing steps required to measure the color which comprise finding a suitable object candidate, extracting an object mask for the found candidate and finally measuring the color of the object candidate. In each step, the acquired information is compared to the properties known for the searched object. Details on this part of the processing are given in the next chapter (chapter 4). The elemental processing steps

1. Search a suitable object candidate

**Figure 3.2.:** This reduced scenario is used to evaluate the proposed system architecture.

2. Extract a mask for the object candidate found
3. Extract the color of the object candidate using its mask

will result in the processing flow shown in Fig. 3.3. To locate a suitable object candidate the attention control mechanism triggers the saliency computation (see (1) in Fig. 3.3). As indicated by the red arrow in Fig. 3.3, the attention control mechanism provides modulatory inputs for the saliency processing. These inputs are constructed using the knowledge about the searched object. In the example of finding a red object, the system knows the color to look for and can provide this color as a modulatory input for the underlying processing. Furthermore, spatial information can be provided, e.g. the object is on the table or

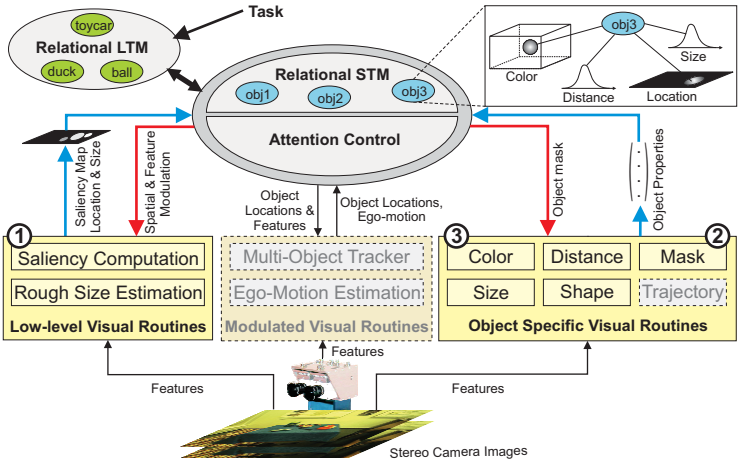


**Figure 3.3.:** The processing flow for the simple task "find a red object". First, a suitable object candidate is located (1), then the object candidate is segmented (2) and finally the color of the candidate is measured (3).

the object is on the left. Additional to the modulatory color input, a weight for each saliency feature is provided by the attention control mechanism. Now the saliency map is computed using a color map tuned to enhance red objects, while suppressing objects with other colors. This way, the saliency map is tuned for the given task. The resulting saliency map is propagated to the attention control mechanism (blue arrow in Fig. 3.3) together with a map containing the roughly estimated retinal object sizes at each location (see Appendix B for details). The attention control now selects the most promising object candidate by performing a maximum selection. The selected candidate is attended and stored in the short-term memory, together with its location and rough size.

To make sure the attended object candidate is actually the searched object, the system has to measure its color. As discussed previously,

the second processing step is to extract a mask of the selected object candidate. To do so, the attention control mechanism triggers a segmentation routine (see (2) in Fig. 3.3). Again, the attention control provides modulatory information about the object candidate to the visual routine (red arrow in Fig. 3.3), namely the location and the rough size previously stored in the short-term memory. The segmentation routine uses the information provided as a starting condition to segment the object candidate. The resulting mask is then returned to the attention control mechanism (blue arrow in Fig. 3.3) and stored in the short-term memory together with the already acquired information.

At this point the location, the rough size and the mask for the attended object candidate are known. This is still not enough information to solve the task of finding a red object. However, it enables the system to finally measure the color of the object candidate to make sure it is red. To do so, the attention control mechanism triggers the visual routine for extracting the color of the object candidate (see (3) in Fig. 3.3). The visual routine for extracting the color requires location and mask of the object candidate, so the attention control mechanism provides these information as modulatory inputs (red arrow in Fig. 3.3). The visual routine extracts and returns the measured color of the object candidate (blue arrow in Fig. 3.3). The attention control mechanism stores the color in the short-term memory along with the other properties of the candidate and finally compares the measured color with the requested one. If they match (at least to a degree specified before), the task is solved. Otherwise a new object candidate needs to be found using the saliency map, while the previous candidate is suppressed (inhibition of return). The whole processing loop will repeat until the object is found or all locations in the visual scene have been visited.

Keep in mind that the combination of the visual routines is done dynamically and on-the-fly. During the measurement process the attention control mechanism evaluates each newly acquired information by storing the information in the short-term memory and comparing it

to the properties of the searched object. If a mismatch between the predicted and measured property is detected, the measurement process will be aborted and a new object candidate will be localized.

## 3.4. Components



This section deals with details about the different processing components of the system architecture. First, the low-level preprocessing routines will be discussed briefly. Then the saliency computation is presented in more detail, as it is a central element in spatial attention guiding determining the time it takes until a suitable object candidate is focused. Eventually, selected algorithms used to extract properties of objects are presented.

### Preprocessing

Cameras deliver raw image data that needs to be preprocessed to extract useful features for higher layers. The selection of features that are useful strongly depends on the algorithms used in the concrete system. Therefore, in this section the feature set corresponds to the current implementation of the proof-of-concept system. The system used for the experiments of chapter 5 is equipped with a stereo camera system. Based on the left and right camera image four different feature channels are extracted:

1. RGB color image $\mathbf{I}^L$ of the left camera
2. saturation $\mathbf{s}$ of the left RGB color image
3. lighting $\mathbf{l}$ of the left RGB color image
4. disparity $\mathbf{d}$ and thus the distance $\mathbf{z}$ with respect to the left camera image.

Here, a vector like $\mathbf{z}$ denotes a map that contains one feature for all pixels $x, y$. That is, a map $\mathbf{m}$ has the form

$$\mathbf{m} = \begin{pmatrix} m_{0,0} \\ m_{1,0} \\ \dots \\ m_{x,y} \\ \dots \\ m_{X,Y} \end{pmatrix} , \tag{3.1}$$

for all pixels $x, y$. For multichannel images or maps like the RGB color image multiple single channel vectors form a matrix. That is, an $N$-channel map $\mathbf{M}$ has the form

$$\mathbf{M} = \begin{pmatrix} \mathbf{m}_1^T \\ \mathbf{m}_2^T \\ \dots \\ \mathbf{m}_N^T \end{pmatrix} , \tag{3.2}$$

where $\mathbf{m}_1^T$ to $\mathbf{m}_N^T$ are single channel maps with the notation mentioned in Eq. 3.1. If only a single pixel $x, y$ of the map above is considered, one gets the following vector $\mathbf{m}(x, y)$, defined as

$$\mathbf{m}(x, y) = \begin{pmatrix} m_1(x, y) \\ m_2(x, y) \\ \dots \\ m_N(x, y) \end{pmatrix} . \tag{3.3}$$

The feature channels of all maps are anchored in the coordinate system of the left camera for an easier fusion of information. The used stereo camera system delivers two RGB color images $\mathbf{I}^R$ and $\mathbf{I}^L$ for the right and the left camera respectively. The HLS color conversion [Gonzalez and Woods 2002] is used to calculate the saturation $\mathbf{s}$ and lighting $\mathbf{l}$ of the left camera image. In the current system implementation,

the stereo processing is done using a correlation based algorithm [Fua 1993]. Alternatively, more elaborated patch based stereo algorithms like [Hirschmüller 2005] or model-based stereo algorithms like [Einecke et al. 2008] could be used instead. See [Scharstein and Szeliski 2002] for an overview on state-of-the art stereo algorithms. The calculated disparity map $\mathbf{d}$ is also anchored in the coordinate system of the left camera. The disparity map $\mathbf{d}$ is then converted to a distance map $\mathbf{z}$ using the known parameters of the stereo camera system, like the distance between the two cameras, the focal point of the cameras, etc. The resulting map $\mathbf{z}$ contains the distance $z_{x,y}$ for each pixel $x, y$ of the camera. Finally, all preprocessed maps are combined to a single feature matrix $\mathbf{F}$ for all pixels $x, y$ that has the form

$$\mathbf{F} = \begin{pmatrix} (\mathbf{i}_r^L)^T \\ (\mathbf{i}_g^L)^T \\ (\mathbf{i}_b^L)^T \\ \mathbf{l}^T \\ \mathbf{s}^T \\ \mathbf{z}^T \end{pmatrix} . \tag{3.4}$$

and can be accessed by all higher layers of the system.

## Saliency Computation

The saliency computation provides a map of possible object candidates and their estimated size. This initial location of object candidates is very important for the system, as the quality of the saliency map determines if and how fast a search task can be completed successfully. There are a lot of different algorithms to compute a spatial modulation map. Some of them are purely bottom-up driven like [Itti and Koch 2000], but the vast majority incorporate top-down modulation to guide the spatial attention. This top-down modulation is normally done by modulation and weighting of the input features using knowledge about

the object to search. Examples for spatial attention algorithms using such top-down modulations are [Wolfe 1994, Hamker 1998, Frintrop et al. 2005, Navalpakkam and Itti 2007] and [Michalke et al. 2008]. While these algorithms mainly focus on the modulation of features to enhance the signal-to-noise ratio between the searched object and the background, [Gross et al. 1995] and [Heinke 1997] also propose a spatial guidance of attention. This spatial guidance is achieved by learning relations between object features and their spatial arrangement. By doing so, object-specific informative locations and features can be predicted in the input image, leading to a shorter overall scanning path.

Compared to the very advanced spatial attention algorithms mentioned above, the saliency computation used in this thesis is kept relatively simple and is heavily inspired by the work presented in [Itti and Koch 2000] and [Navalpakkam and Itti 2007]. Nevertheless, it incorporates a basic top-down modulation scheme on the color feature and the weighting of visual routines. However, the simple computation can be replaced by the more advanced algorithms above at any time.

A novel algorithm to coarsely estimate the pixel size of objects is proposed as part of this thesis. The schematic illustration of the saliency processing is shown in Fig. 3.4. Three different processing steps can be identified: tuning of one or more channels of the input features $\mathbf{F}$ (blue in Fig. 3.4), calculation of the center-surround contrast (yellow in Fig. 3.4) and computation of the lateral dynamics (green in Fig. 3.4). First, the input features are modulated to enhance the contrast of a searched object against other regions in the scene. Here, a method similar to the one proposed in [Navalpakkam and Itti 2007] is used. To modulate a certain pixel in the input image, the distribution of an object property is assumed to be Gaussian. Hence, the similarity $\mathbf{t}$ between a selected feature $\mathbf{f}_i$ and the Gaussian distribution $(\mu, \sigma)$ of the searched object property can be calculated for each pixel $x, y$. In the current system implementation, this kind of feature biasing is used for the depth map $\mathbf{z}$ resulting in a biased map $\mathbf{t}_z$ with the top-down

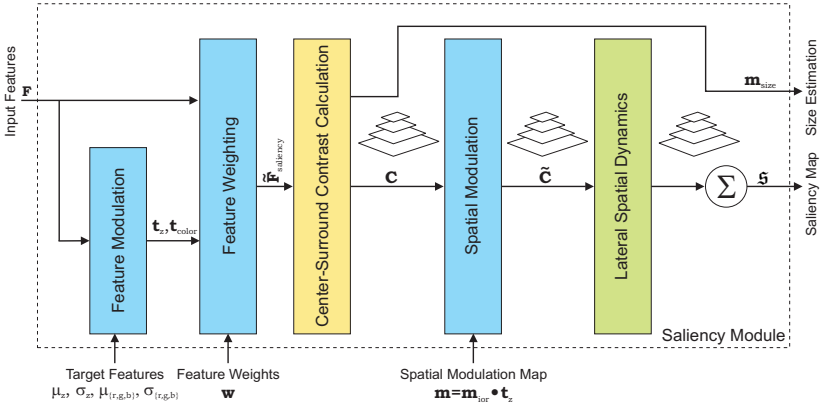


**Figure 3.4.:** The input features **F** are modulated and weighted according to the top-down modulatory inputs (bottom). After computing the contrast, spatial modulation and lateral dynamics, the result is integrated into a final saliency map $\mathfrak{s}$. All these computations work on spatial resolution pyramids. Additionally to the saliency map, the center-surround contrast is used to roughly estimate object sizes.

provided Gaussian distribution $\mu_z, \sigma_z$, where $\mu_z$ denotes the mean and $\sigma_z$ the standard deviation

$$t_z(x,y) = e^{-\frac{1}{2\sigma_z^2}(z(x,y)-\mu_z)^2} \quad . \tag{3.5}$$

For the RGB image $\mathbf{I}^L$ consisting of the channels $\mathbf{i}_r^L$, $\mathbf{i}_g^L$ and $\mathbf{i}_b^L$, the equation above is extended to the multichannel case. With the Gaussian distribution $(\mu_r, \mu_g, \mu_b), (\sigma_r, \sigma_g, \sigma_b)$ the biased color map $\mathbf{t}_{color}$ is calculated as

$$t_{color}(x,y) = e^{-\frac{1}{2}\sum\limits_n \frac{(i_n^L(x,y)-\mu_n)^2}{\sigma_n^2}} \quad . \tag{3.6}$$

Here $n$ is the $n$-th channel of the input features $n = \{r, g, b\}$, $\mu_n$ represents the mean values of the Gaussian distribution and $\sigma_n$ is the standard deviations of the different channels. By applying this kind

of modulation, locations with features similar to the searched ones are enhanced, while regions with different features are suppressed. The Gaussian distribution describing features of the searched object is provided by the attention control mechanism based on object knowledge and the given task. The values for the means $\mu$ and standard deviations $\sigma$ are stored together with the objects as described in chapter 4.

Out of the biased and unbiased features the input feature matrix $\mathbf{F}_{saliency}$ for the saliency computation is constructed

$$\mathbf{F}_{saliency} = \begin{pmatrix} (\mathbf{i}_r^L)^T \\ (\mathbf{i}_g^L)^T \\ (\mathbf{i}_b^L)^T \\ \mathbf{s}^T \\ (\mathbf{t}_{color})^T \end{pmatrix} . \tag{3.7}$$

The resulting feature matrix $\mathbf{F}_{saliency}$ is weighted using a top-down weight vector $\mathbf{w}$ giving the weighted feature matrix $\tilde{\mathbf{F}}_{saliency}$

$$\tilde{\mathbf{F}}_{saliency} = \begin{pmatrix} w_1(\mathbf{i}_r^L)^T \\ w_2(\mathbf{i}_g^L)^T \\ w_3(\mathbf{i}_b^L)^T \\ w_4\mathbf{s}^T \\ w_5(\mathbf{t}_{color})^T \end{pmatrix} . \tag{3.8}$$

As mentioned before, more elaborated algorithms to modulate and weight the input features of the saliency computation exist (see e.g. [Frintrop et al. 2005]). The incorporation of the principles presented there are subject to future work (see chapter 6).

After biasing and weighting the input features, the center-surround contrast $c_i(x, y)$ for each pixel $x, y$ and center-surround combination $i$ must be computed. Here, $i$ denotes a certain combination of the Gaussian filters $\mathbf{g}_i^{center}$ and $\mathbf{g}_i^{surround}$ with $\sigma_i^{center}$ and $\sigma_i^{surround}$ respectively. Contrary to other saliency implementations like [Itti and Koch 2000],

the Euclidean distance between the center and surround feature *vector* is used in this implementation. The contrast $\mathbf{c}_i$ is calculated within one combination $i$ as

$$\mathbf{c}_i^2 = \sum_n \left( \left( \mathbf{g}_i^{center} - \mathbf{g}_i^{surround} \right) * \tilde{\mathbf{f}}_n \right)^2 . \tag{3.9}$$

The $*$ operator in the equation above denotes a convolution of the filter $\mathbf{g}_i^{center} - \mathbf{g}_i^{surround}$ with a single feature channel $\tilde{\mathbf{f}}_n$. For choosing the center and surround combination $i$ for the Gaussian filters the schema proposed in [Itti and Koch 2000] is followed. An efficient implementation of the contrast calculation is based on Gaussian resolution pyramids. By doing so, the calculations $\mathbf{g}_i^{center} * \tilde{\mathbf{f}}_n$ and $\mathbf{g}_i^{surround} * \tilde{\mathbf{f}}_n$ boil down to calculating different pyramid levels of the feature vector.

Now, the contrast maps $\mathbf{c}_i$ are biased using a spatial modulation map $\mathbf{m}$, provided in a top-down manner

$$\tilde{c}_i(x, y) = c_i(x, y) \cdot m(x, y) . \tag{3.10}$$

The top-down spatial modulation map integrates multiple information. First, the modulation map allows a three dimensional biasing by using the biased distance map $\mathbf{t}_z$ described in Eq. 3.5. Second, the attention control mechanism can inhibit locations of known objects based on the memory content using the modulation map $\mathbf{m}_{ior}$. This process is known as inhibition of return (IOR) [Itti and Koch 2000]. The multiplicative combination of both maps leads to

$$m(x, y) = m_{ior}(x, y) \cdot t_z(x, y) . \tag{3.11}$$

Another important difference to other saliency implementations is the fact that in the approach presented here, the spatial modulation is done *before* the lateral dynamics. The implications of this slight change become clear when looking at the effect of the lateral dynamics, which act as strong non-linearities in the spatial domain. That is, while different

peaks compete with each other, their positions get shifted across the map. When the spatial modulation is applied after that non-linearity, one has to take these shifts into account, which is impossible. By applying the spatial modulation map before the lateral dynamics, the problem can be circumvented.

The next stage of the processing is to perform a lateral spatial competition on the contrast maps $\tilde{\mathbf{c}}_i$ (see Fig. 3.4). This is done to enhance the signal to noise ratio (SNR) in those maps. In the current implementation, an Amari field dynamics [Amari 1977] is applied, which leads to similar results as shown in [Itti and Koch 2000]. For details on the Amari field dynamic see Appendix A. Finally, the maps of different scales $i$ are combined into one single saliency map $\mathfrak{s}$, containing locations of object candidates. Figure 3.5 shows some results for an unbiased, a feature biased and a feature and location biased saliency map. As can be seen, less object candidates can be found in the map,

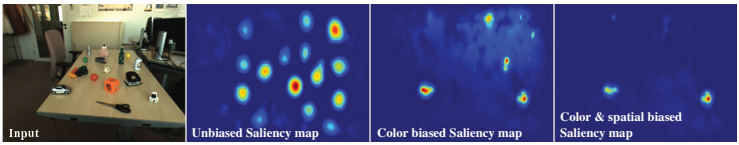


**Figure 3.5.:** This shows an example of an unbiased, color biased (white) and a color and spatially biased (distance bias at $70cm$) saliency map.

by incorporating knowledge about the object. Less object candidates means less search time. Using more advanced biasing strategies, further speedups can be achieved [Navalpakkam and Itti 2007].

Additional to the saliency map, a rough size estimation is calculated by the saliency module. Here, the filter responses of the Gaussian pyramid are used to estimate this size at all locations. An example for such a distribution of filter responses at one position is shown in Fig. 3.6. For homogeneous objects the Gaussian filter responses stay nearly constant until the filter exceeds the object. When the filter size exceeds the

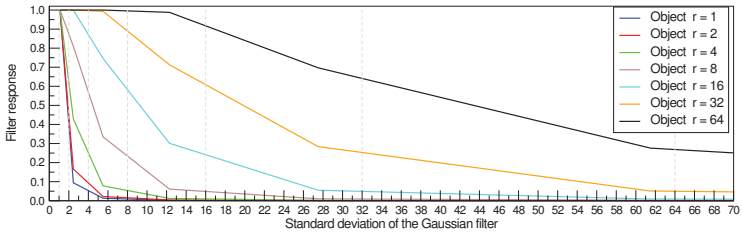


**Figure 3.6.:** The Gaussian filter response stays constant for a star object until the border is reached. At this points the filter response drops dramatically, which can be used to estimate the object size.



object boundaries, the covered structure changes and so does the filter response. By determining the standard deviation of the filter where this change occurs, the rough size of the underlying object can be estimated, as the standard deviation is proportional to the object size [Rebhan et al. 2008]. For more details on the calculation of the rough size estimation see Appendix B. Figure 3.7 shows the calculated object size for a given input scene. The assumption about a homogeneous

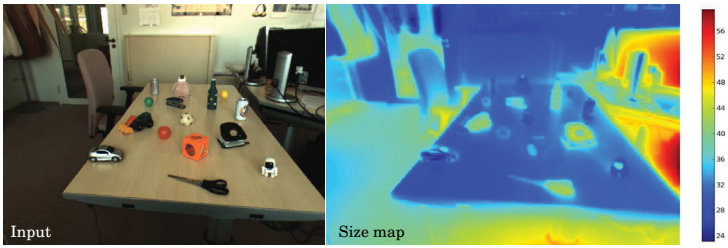


**Figure 3.7.:** On the left hand side the input image is shown. On the right hand side the corresponding size estimation in pixels is shown using a color-code.

object structure barely holds for real world images. Nevertheless, the estimated size has proven to be a good initial hypothesis for algorithms

applied later on [Rebhan et al. 2008]. The final saliency map and the size estimation map provide the location and rough size of an object candidate. This information can now be accessed by the attention control mechanism to select a certain region to focus on.

## Feature Extraction

In normal operation, the attention control mechanism has to measure one or more properties of an object after localizing an object candidate. This feature or property extraction is done by using elementary visual routines that are similar to the ones presented in [Ullman 1984] in complexity. Those visual routines are specialized to extract one property for a given object while using information provided in a top-down manner like an object mask, feature biases, form biases or similar. As the focus of this thesis is not on the visual routines themselves, they are kept rather simple. Especially invariances in rotation and lighting were not considered here for the sake of simplicity. However, also in this part of the system more elaborated visual routines could be used.

As stated earlier, a common interface is required to make the system extensible. Two interfaces have to be defined for the visual routines, one to the low-level features $\mathbf{F}$ and one to the memory and attention control mechanism. The number and kind of preprocessed features $\mathbf{F}$ is fixed for a given system and the information only flows from the lower-layer to the visual routines. Contrary to this, the interface to the memory is more complex, as information needs to be exchanged between the visual routines and the memory. To pass distribution information to the memory, the cluster centers of a k-means clustering [Lloyd 1982] is used as it can approximate both unimodal and multimodal distributions easily. The required precision of the approximation and thus the potentially required number of cluster centers are defined beforehand. The information passed from the memory to the visual routines consists of an object mask and optional biasing information.

The object mask is a binary image with the same size as the feature maps. The optional biasing information consists of the cluster centers $(\mu_1, \mu_2, ..., \mu_N)$. For visual routines delivering spatial information like segmentation algorithms, the information is passed as maps. By approximating the feature distributions in a given mask with a k-means algorithm, information about the variance or exact characteristic of the distribution is lost. That means if the description of an object strictly depends on such an exact representation of its feature distribution, it will be impossible to represent or discriminate such an object. In the conducted experiments, no such a case was found. Also for this part of the system multiple alternative algorithms exist. The feature distribution could be stored as is, excluding the possibility of information loss, or as an approximation like histograms, Gaussian Mixture Models [Bishop 2006a], Self-Organizing Feature Maps [Kohonen 1990], Growing Neural Gas [Firtzke 1995], Radial Basis Functions [Powell 1985] to only name a few. The only requirement for the approximation is that a distance measure between two of those approximations can be defined. For now, the k-means algorithm is used due to its speed and simplicity. For details on the k-means algorithm see Appendix C.

Currently, the system comprises visual routines for extracting the RGB color property $\mathbf{p}_{color}$, the distance $p_z$, the pixel size $p_{size}$, the physical size $p_{physicalsize}$, a rudimentary shape $p_{shape}$ and the object mask $\mathbf{p}_{mask}$. To calculate the object mask, segmentation algorithms like region-based segmentation [Sonka et al. 1998], Graph-cut [Boykov and Jolly 2001] or Level-Set methods [Mumford and Shah 1989] can be used. The mask itself is a function of the retinal object location $\mathbf{x} = (x, y)^T$, the low-level feature vector $\mathbf{F}$ and potential feature modulations $\mathbf{F}_{mod}$

$$\mathbf{p}_{mask} = f(\mathbf{x}, \mathbf{F}, \mathbf{F}_{mod}) \ . \tag{3.12}$$

Because the object mask is required to calculate all object properties it is curcial for the system. Current system the multi-cue Level-Set method presented in [Weiler and Eggert 2007] is used.

The color and distance properties, $\mathbf{p}_{color}$ and $p_z$ are approximated using a k-means clustering as mentioned earlier. The approximation estimates the cluster centers $\mu$ in the feature space of a property $\mathbf{p}$ within the given mask $\mathbf{p}_{mask}$. In the following the k-means clustering method is expressed as $kmeans(\cdot, \mathbf{p}_{mask})$, where $\cdot$ is replaced by one or more feature maps $\mathbf{F}$ (see Appendix C for details). The second argument is always the binary object mask $\mathbf{p}_{mask}$

$$\mathbf{p}_{color} = \mu_{color} = kmeans(\mathbf{I}^L, \mathbf{p}_{mask}) \tag{3.13}$$

$$p_z = \mu_z = kmeans(\mathbf{z}, \mathbf{p}_{mask}) \ . \tag{3.14}$$

Here, $\mathbf{I}^L$ is the left RGB camera image and $\mathbf{z}$ is the computed depth map. In the current implementation the k-means algorithm is started with multiple randomly placed cluster centers. However, currently only the dominating cluster, containing the most elements, is passed on to the system's memory. This leads to an estimation of the dominating color or depth while at the same time outliers and locations in the mask, that potentially belong to the background, are ignored. For the color approximation it would be desirable to pass multiple clusters to the memory as an object might have multiple colors. The testing of this setting and the evaluation of other approximation and cluster algorithms are subject to future investigations (see chapter 6).

To determine the size and shape of an object in the current implementation, the minimal bounding box $\mathbf{p}_{BB}$ is calculated for the object mask. To extract this minimal bounding box the algorithm described in [Toussaint 1983] is used, returning a width $w$ and a height $h$. In the following the estimation method is denoted with $boundingbox(\cdot)$ and only gets one argument, the object mask $\mathbf{p}_{mask}$

$$(w, h)^T = boundingbox(\mathbf{p}_{mask}) \ . \tag{3.15}$$

For details on the bounding box estimation algorithm see Appendix D. As an assignment of the width $w$ and the height $h$ is arbitrary, the

orientation of the bounding box is normalized in a way that $w \geq h$. Using the normalized bounding box parameters $w$ and $h$, the size $p_{size}$ and shape $p_{shape}$ properties are calculated as

$$p_{size} = h \cdot w \tag{3.16}$$

$$p_{shape} = h/w \ . \tag{3.17}$$

In this case, the size is the area and the shape is the aspect ratio of the bounding box, identifying "compact" objects with $p_{shape} \approx 1$ and "elongated" objects with $p_{shape} \ll 1$. Using the pixel size $p_{size}$ and the depth $p_z$ of an object, the physical size can be approximated by

$$p_{physicalsize} \propto p_{size} \cdot p_z^2 \ . \tag{3.18}$$

All processed properties are then passed to the system's short-term memory if requested.



## 3.5. Discussion

In this chapter, a flexible system architecture was proposed that provides the foundation for a demand driven information acquisition. One can of course find similarities to other state-of-the-art architectures like [Rensink 2000] here. However, the processing paradigm differs dramatically. In this thesis, a top-down driven processing is proposed which is essential to process and store only those information necessary to solve the given task. This flexibility can only be achieved by dynamically combining different visual routines which provide elemental processing steps as proposed in [Ullman 1984]. The coordination of these visual routines is done in the attention control part of the system.

An exemplary processing flow was presented using the scenario shown in Fig. 1.4. The combination of the visual routines working together is coordinated by the attention control mechanism. Here, the saliency

computation plays a major role for locating initial object candidates relevant for the given task. The saliency implementation used in this thesis is heavily inspired by the state-of-the-art in this field. It allows for top-down modulation in both the spatial and feature domain. Furthermore, it comprises an algorithm for estimating the coarse pixel size of an object candidate that was developed as part of this thesis. Beside the saliency processing, visual routines for extracting different properties of object candidates like color, shape and size were presented. Those routines are kept very simple to be able to focus on the system architecture and the attention control mechanism. The extraction of the object properties is done by applying a k-means clustering on the feature vector. This form of representation has been chosen because it allows for an information reduction in the input data, while being able to estimate the dominating properties in the feature vector for a given object mask. Once calculated, the processed information is passed to the system's memory, which stores this information in a graph-based way. Both the memory architecture and the attention control mechanism are subject to the next chapter.

# 4. Memory Architecture and Attention Control



In the previous chapter, a flexible system architecture was proposed. Essential parts of this system architecture are the short-term and long-term memory together with the attention control mechanism. Those parts are subject of this chapter. The questions how the system accounts for its algorithmic needs and how those needs can be represented consistently (compare chapter 1) are covered here as well. Furthermore, an attention control mechanism is proposed.

To be able to decide which properties of an object should be measured, the system needs to incorporate both the current task and the knowledge about the world. For the visual search tasks aimed at here, this world knowledge not only comprises information about the object to be found, but also about other objects in the scene. In the first part of this chapter, the focus is on the memory itself and the way it stores and represents information. At the end of the first part, a way to deal with functional dependencies between object properties will be presented. This procedural knowledge is important for the system, as the highly specialized visual routines need to be combined in a certain order to solve more complex problems. Afterwards, the memory elements presented before are combined into a memory architecture used in the system. Furthermore, it will be briefly discussed how this architecture can be efficiently used by the attention control mechanism later. In the third section of this chapter the attention control algorithm itself is described. It implements the combination and scheduling of the visual

routines based on the procedural and semantic memory of the system. Important aspects of the memory architecture and the control mechanism proposed in this chapter were already published in [Rebhan et al. 2008; 2009a] and [Rebhan et al. 2009b].

## 4.1. Relational Memory

In this thesis, the system's memory is more than just a "data store" for the world knowledge. More importantly, it constitutes a suitable representation for a control mechanism deciding which properties need to be measured for different objects in order to fulfill a given task. A suited flexible and general memory was proposed in [Röhrbein et al. 2007]. This memory is capable of representing and operating on large object ontologies. Thus, the work done here is based on this graph-based structure. However, major improvements like representation of procedural knowledge and the memory architecture itself have been developed as a part of this thesis.

The basic structure of the memory is a graph. Figure 4.1 shows an exemplary representation of an object in the graph structure of the memory. In Fig. 4.1, the graph structure on the right hand side is the system's internal representation of the object on the left. Nodes in the graph represent both properties and the object itself. Edges connecting the nodes specify a "hasProperty" relation between the object and its properties. Importantly, the property nodes are anchored in the sensory representations as shown in Fig. 4.1. That is, each property node stores the sensory value or distribution it stands for. As an example, the object `obj1` in Fig. 4.1 has a property `color1`. The color node itself stores a Gaussian distribution of the object's color, in this case the color red (see Fig. 4.1). In the following, more relations between nodes are defined, necessary to represent both the semantic and procedural knowledge of the system.

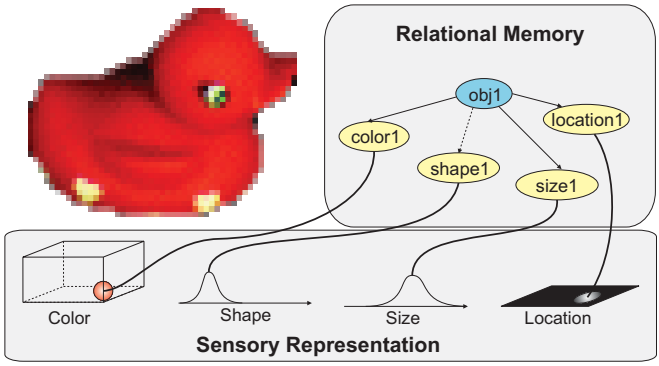


**Figure 4.1.:** The duck is represented in the memory graph of the system. Internally its properties are described by nodes, connected by edges that express a "hasProperty" relation. Each node stores a link to the sensory representation.



## Representing Knowledge

In a more formal way, the memory is a graph $G = (V, E)$ where $V$ is the set of vertices or nodes and $E$ is the set of edges connecting these nodes. The nodes $V$ can be further separated into object nodes $O$ and property nodes $P$ where $V = O \cup P$. An illustration of this structure can be found in Fig. 4.2. The sensory information is stored in the property nodes. Object nodes then act as combining elements that group different properties. In this interpretation, the nodes carry the information and the edges represent relations between the different nodes and thus between the different information. It is not only possible to represent a single relation between nodes, but the memory architecture is capable of representing an arbitrary number of relations. That is, one can express relations between objects, between an object and property nodes and between property nodes. The role of a node is determined by the incoming and outgoing edges, i.e. its graph structure, rather than by an artificial definition. In the same way, the meaning of an edge is not

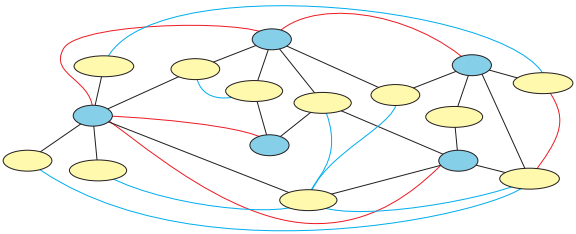

**Figure 4.2.:** The memory graph consists of nodes and edges. Objects and properties are represented by nodes, whereas edges represent relations between objects and properties. Different edge colors denote different relations that can be established between nodes.

determined by definition, but rather by the pattern in the memory and the algorithmic interpretation of this structure. Here, the edges shown in Fig. 4.2 are bidirectional, implemented as two oppositely directed edges. When applying this schema consequently, the memory structure can be interpreted as a bunch of interconnected graphs (one for each relation or edge type). This interpretation allows the application of well-known graph algorithms as used later in this chapter.

As mentioned above, three classes of relations can be modeled with bilateral edges in memory: object-property, object-object and property-property relations (also see Fig. 4.2). A more concrete illustration of relations used in the system can be seen in Fig. 4.3. The probably most common one is the object-property relation. A typical relation of this kind is the "hasProperty" relation. As edges are bidirectional, there also exists an edge into the other direction called "isPropertyOf". Using this relation, properties can be assigned to an object. If for example one wants to express that an object $o$ has a certain color $p_{color}$, one first would generate an object node $o \in O$ and a property node $p_{color} \in P$. Then one defines an edge between $o$ and $p_{color}$ as

$$o \xrightarrow{\text{hasProperty}} p_{color} \ . \tag{4.1}$$

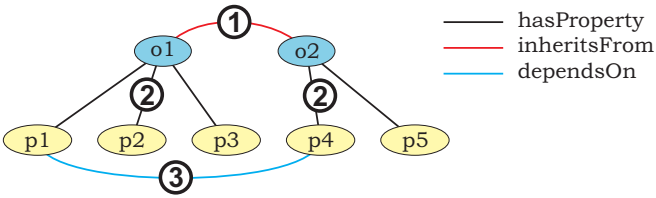


**Figure 4.3.:** In the memory, relations between 1) the objects $o_1$ and $o_2$, 2) between object $o_1$ and the properties $p_1$ .. $p_3$ as well as between object $o_2$ and the properties $p_4, p_5$ and 3) between the property $p_1$ and $p_4$ are represented. For interpretations of these relations, please refer to the text.



At the same time also a relation in the inverse direction called

$$p_{color} \xrightarrow{\text{isPropertyOf}} o \tag{4.2}$$

will be generated. In the following examples only the definition in one direction will be mentioned for readability. However, an edge for the inverse direction is always created simultaneously. As it is common in graphs, one object can have arbitrarily many properties and a property can belong to arbitrarily many objects.

The second type of relation that should be described in more detail here is the object-object relation. An example for this kind of relation is the "contains" relation with the counterpart "isContainedIn". When establishing a "contains" edge, a spatial relation between two object nodes $o_1$ and $o_2$ is expressed. That is,

$$o_1 \xrightarrow{\text{contains}} o_2 \tag{4.3}$$

expresses that object $o_2$ is located completely inside $o_1$. Of course, one now can make use of the ability to define arbitrarily many relations between objects such as "touches", "isNextTo", etc. Currently this type of spatial relations is not yet used in the system, however, research on how to use them has been done by [Riad et al. 2009].

Another important relation, the "inheritsFrom" and the corresponding inverse "specializesTo" relation, can be established both between two objects or between two properties. This means that for the relation $o_1 \xrightarrow{\text{specializesTo}} o_2$, object $o_2$ will have exactly the same structure and properties as $o_1$. The same is true for a pair of properties $p_1, p_2$, which then share the same values and relations. However, based on these inherited properties or values, specializations can take place. That means, inherited properties can be removed or overwritten, new properties can be added or values can be modified. Here is an example: imagine an object $o_1$ with the color property $p_{color}$ attached ($o_1 \xrightarrow{\text{hasProperty}} p_{color}$) and an object $o_2$ without property nodes attached. When writing

$$o_1 \xrightarrow{\text{specializesTo}} o_2 \ , \tag{4.4}$$

$o_2$ inherits the color of $o_1$ and is also assumed to have the color $p_{color}$. When attaching another property node like the size $p_{size}$ to $o_2$

$$o_2 \xrightarrow{\text{hasProperty}} p_{size} \ , \tag{4.5}$$

the second object is specialized, that is, it inherits all properties from $o_1$ but defines additional or modified properties. For property nodes, usually the values are modified and the edge is used to express for example a similarity.

The last relation type that should only briefly be mentioned here is a property-property relation. To account for technical aspects of the system and the sequential processing, the relation "dependsOn" and its inverse relation "influencedBy" are proposed in this thesis. This relation represents knowledge about how to measure properties and is described in the next section in more detail.

The relations defined in this section construct a small subset of the edges that may be found in a memory graph. To summarize, we have a memory graph

$$G = (V, E) \tag{4.6}$$

with the set of nodes $V$ split into property nodes $P$ and object nodes $O$

$$V = P \cup O \ . \tag{4.7}$$

Furthermore, the edges $E$ split into the different relations $E_{hasProperty}$, $E_{specializesTo}$ and $E_{dependsOn}$ as well as their counterparts $E_{isPropertyOf}$, $E_{inheritsFrom}$ and $E_{influencedBy}$

$$\begin{aligned} E &= E_{hasProperty} \cup E_{specializesTo} \cup E_{dependsOn} \\ &\cup E_{isPropertyOf} \cup E_{inheritsFrom} \cup E_{influencedBy} \ . \end{aligned} \tag{4.8}$$

Classical AI approaches face the so called "grounding" problem. Here the difficulty is to relate an abstract sensory node like "red" to a sensory experience of the system, as this information is not stored in the node itself. Contrary to this, the memory framework used here explicitly links the sensory representation to each property node as illustrated in Fig. 4.4. By keeping a link to the sensory representation in the

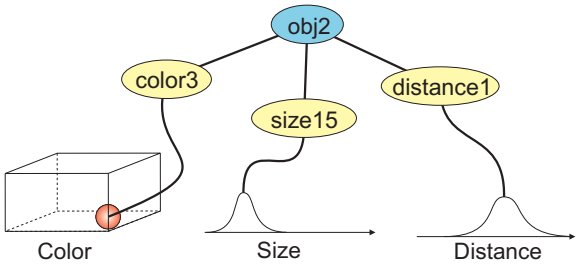


**Figure 4.4.:** All property nodes storing sensory information explicitly store a link to the sensory representation of this property. In the case shown here, color, size and distance internally store a Gaussian approximation of the real sensor data.

property nodes, an artificial abstraction between the sensory side and the memory is avoided. Furthermore, the resulting anchoring of the node allows for an easy conversion of the node information back to the

sensory representation. This is very important when modulating input features of visual routines as described in chapter 3. To avoid an excessive usage of memory by storing very detailed sensory representations in the property nodes, those representations are approximated using a k-means clustering. To summarize, by storing a link to the sensory representation, the property nodes are anchored in these representations and thus avoid the "grounding" problem classical AI approaches face. Furthermore, the measurement of an object property can be triggered using the connection of the property node to the actual visual routine (through the sensory representation).

## Representing Functional Aspects

When relying on sequential processing of information such as in technical systems, the correct ordering of information acquisition processes is crucial to keep the system functional. If for example a visual routine for acquiring the color $p_{color}$ of an object requires information about the retinal location $p_{x,y}$ of the object, it is important to acquire the location *before* the color routine is called. In current state-of-the-art-systems this is ensured by manually constructing processing pipelines in a static processing regime. This means, that the whole processing apparatus including e.g. saliency, segmentation and classification is executed even though for example only the color of an object needs to be determined. The ordering of the visual routines becomes even more important when breaking this static processing as done in this thesis. In a flexible processing regime as proposed in this work, highly specialized visual routines are called on demand without executing unrelated ones. Nevertheless, one must ensure that all information required by the called visual routine is available. For doing so, a new type of property-property relation called "dependsOn" and its inverse relation "influencedBy" have been introduced. Using this relation, functional dependencies between sensory pathways can be expressed. That is, if

one has a property $p_1$ and a property $p_2$ and writes

$$p_1 \xrightarrow{\text{dependsOn}} p_2 \ , \tag{4.9}$$

this means that the processing of $p_1$ can only take place if $p_2$ has already been processed. The definition of some of these relations is shown in Fig. 4.5. As can be seen, only direct dependencies between the different property nodes are defined. We will see later that by using these definitions a canonical dependency graph can be created automatically. The problem here is very similar to problems found in compiler construction. Even though Ballance et al. state in their paper that "neither switches nor control dependence are required for a demand driven interpretation" [Ballance et al. 1990, p. 261], some modifiers for dependency relations are required to account for the algorithmic needs of visual routines. Figure 4.6 shows those required modifiers. They cover the cases:

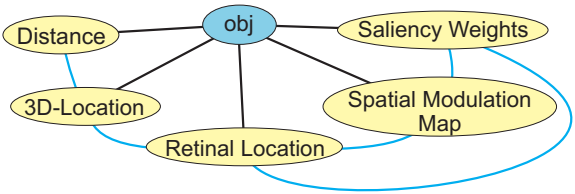


**Figure 4.5.:** Exemplary dependencies of different property nodes. To measure the three dimensional location of an object, the system needs to know the retinal location and the distance of the object. However, theses properties themselves depend on other properties like saliency cue weights and a spatial modulation map.

a) The operation of node $C$ is optional and not absolutely required for measuring node $A$, but would e.g. improve the result of the measurement. For example a spatial modulation map could constrain the search space for an object, but is not mandatory. If the map is not available, the whole space has to be searched for

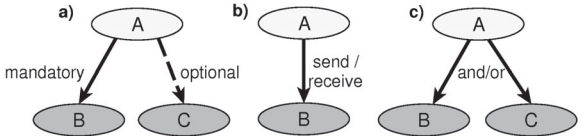


**Figure 4.6.:** Different modifiers are required for dependency relations: a) dependencies can be optional or mandatory, b) send and receive operations can be executed by the target node and c) all dependencies can be required to be fulfilled or only one dependency needs to be fulfilled.

the object (see Fig. 4.6 a). The type of a dependency is then $type(e \in E_{dependsOn}) = \{\text{"optional"}, \text{"mandatory"}\}$.

b) Property nodes can execute two operations. Each property node can potentially send its information to another node / sensory representation or receive information from another node / sensory representation. The modifier shown in Fig. 4.6 b specifies which operation node $B$ has to execute in order to fulfill the dependency of node $A$. The operation requested by a dependency is then $operation(e \in E_{dependsOn}) = \{\text{"send"}, \text{"receive"}\}$, corresponding to the modulation and measurement of sensory representation by visual routines respectively.

c) There might be alternative ways to measure a certain property, so the system needs to fulfill only one of several dependencies. Think of different segmentation algorithms for estimating the shape of an object, where only one of those algorithms is required to get a shape (see Fig. 4.6 c). A dependency can take one of the following logical modes $mode(e \in E_{dependsOn}) = \{\text{"and"}, \text{"or"}\}$. If no mode is given, the "and" mode is assumed by default.

Based on the dependency relations along with the modifiers, it is possible to model functional dependencies between property nodes and thus also between the visual routines bound to those nodes. This distin-

guishes the memory used here from other architectures that do not explicitly model relations between processing modules.

## 4.2. Memory Architecture

Based on the relational memory described in the previous section, a memory architecture for the system is defined in this part. The memory architecture is split into four parts as shown in Fig. 4.7. At the bottom

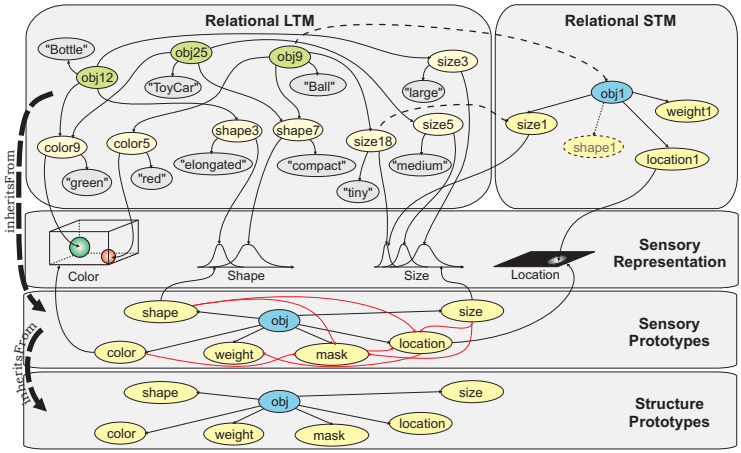


**Figure 4.7.:** Top row: The long-term and short-term memory store world knowledge and scene information respectively. Second row: Sensory information computed by the visual routines. Third row: Sensory Prototypes that define the binding of the object property nodes to the corresponding sensor. Furthermore, dependencies between property computations are defined (red edges). Objects in the long- and short-term memory inherit information from this layer. Last row: The structure prototype definitions express the relations between an object and its properties.

of Fig. 4.7, the prototypical memory structure is shown. There, possible patterns of relations between property nodes $P$ and object nodes $O$

and thus the object structure are defined. The relations used here are mainly "hasProperty" relations. In the example shown in Fig. 4.7 an object can have the properties shape, color, mask, weight, etc.

Based on the structure prototype layer, the sensory prototype layer is constructed (see Fig. 4.7). This layer *inherits* the structural definition of the layer below explicitly denoted by "inheritsFrom" edges in the graph. By doing so, the relations between the property nodes $P$ and the object nodes $O$ are transferred to the sensory prototype layer without the need to redefine them. Additionally, relations between nodes and the sensory representations are defined here. The most important point is the "binding" of the sensory representations to their corresponding property nodes. The sensory representations themselves are calculated by the visual routines described in chapter 3. That way, the property nodes are also bound to the visual routines of the system. Furthermore, dependency relations between property nodes $P$ are defined in this layer (shown as red edges in Fig. 4.7) to represent the measurement process as described in section 4.1. By binding the property nodes to the sensory representations and thus the visual routines together with the dependency definitions between property nodes, the system can use the knowledge about its own measurement apparatus to estimate costs and information gain of different visual actions.

The long-term memory *inherits* both the sensory binding along with the dependency structure and the structural definition of sensory prototypes. Note that the structural definition of the sensory prototypes is itself inherited from the structural prototype layer. Contrary to the layers described before, multiple instances of a prototypical object are created in the long-term memory part. Each of these objects is a specialization of the prototypical one and thus has individual properties bound to the object. These individual property nodes represent for example the color or the shape of an object found in the visual world. When looking at Fig. 4.7, not all possible properties are attached to the node in the long-term memory. Here, only properties that are not

volatile but persistent for a certain object are stored. That is, properties that are constant over time are bound to the object node, whereas volatile properties like the distance or location of an object are not instantiated. Furthermore, you can see that the nodes in the long-term memory do not contain "names". This illustrates that property nodes and object nodes are only anchored in the sensory representations and do not require a label. However, labels (gray nodes in Fig. 4.7) may be attached to allow for a more intuitive use of the system by humans. In the following illustrations, these labels are merged into the nodes to increase the lucidity. However, please keep in mind that the labels are *not* necessary for the system to function and do not provide any information to the system.

The fourth and last part of the memory architecture is the short-term memory. This part of the system stores knowledge about the current scene. While doing so, information can be *inherited* both from the sensory prototype layer and the long-term memory. If an object is not yet known or identified, the object inherits its information from the sensory prototype object. As soon as the object is identified, the object in the short-term memory inherits information from the long-term memory. By doing so, properties that are not yet measured can be predicted. Here, the identical structure of an object in both the long-term and short-term memory eases the transfer of information between these two memory instances. In Fig. 4.7 an example is shown, where `obj1` in the short-term memory is identified to correspond to `obj9` of the long-term memory (dashed line in Fig. 4.7). This means, that an object (`obj1`) was found in the current scene, having similar properties to a known object (`obj9`) in the long-term memory. As illustrated by the solid outlines of the nodes, size, location and saliency weights of `obj1` are measured or sent respectively. However, the shape of `obj1` was not yet measured, but can be predicted from the knowledge about `obj9`. This mechanism can be used to establish a prediction-confirmation loop, where the long-term memory provides the predictions and the measurement apparatus tries to confirm the properties using the visual routines.

In the architecture proposed here, multiple layers are used to separate the structural, procedural, semantic and sensory definition of an object. The separation of procedural (skills) and semantic knowledge is also proposed in [Langley et al. 2009]. However, a rule-based representation of the knowledge is used there, with very high-level procedures (skills) and without an anchoring of the object properties in the sensory representation. By splitting the memory architecture, it can be easily adapted to different systems or underlying algorithms by only changing the sensory layer. Here, the structural definition accounts for the sensor pathways and the variety of information available in a system. The sensory definition accounts for the concrete implementation of a sensory pathway and the underlying algorithms. Additionally, the dependencies between different visual routines and thus the dependencies between the underlying algorithms is consistently defined in the sensory prototype layer. Both the long-term and the short-term memory inherit from the sensory layer, which leads to an identical object structure in both parts. This identical structure later eases the transfer of information between the two memory instances. By doing so, a prediction-confirmation loop can be established where the long-term memory predicts properties of objects in the scene and the system tries to confirm these predictions using its visual routines. The question of which properties to measure when is subject to the next section.

## 4.3. Processing Flow Example

In section 3.3 the exemplary flow in the system was presented. However, the memory part was only described very briefly. This section focuses on the processing flow in the memory part of the system. Again, the system's task is to "find a red object" in the scene. Figure 4.8 shows the processing flow in the memory. The task "find a red object" is given to the system by activating the corresponding property node in the long-term memory. In Fig. 4.8 (1) the "red" property node is activated. This

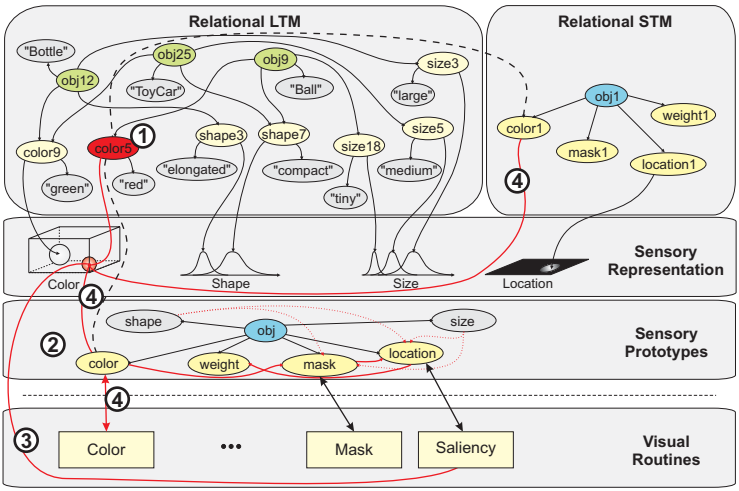


**Figure 4.8.:** The processing flow in the memory for the task "find a red object". (1) The task is given by activating the "red" property node. (2) The attention control determines the need for measuring the color of object candidates and resolves procedural dependencies of the color visual routines by parsing the dependency graph of "color". (3) The expectation of a "red" color is used to bias the saliency computation. This is possible, as the sensory representation of "red" is stored in the node. (4) The measured properties of the object candidate are stored in the short-term memory. In a last step, the color of the object candidate is validated and the task is solved.

activation triggers a sequence of actions in the system aimed at finding an object in the current scene that is similar to the activated one. The attention control now tries to find out what needs to be done, i.e. which properties need to be measured to fulfill the task. The activated node gives a direct hint on what needs to be measured, namely the color of an object candidate. However, to measure an object's color, an object needs to be located beforehand. In this thesis it was proposed to also store these procedural dependencies in the system's memory as described in section 4.1 and shown in Fig. 4.8 (2). In the next section of

this chapter, the exact description of a parsing algorithm resolving the procedural dependencies is given. For now, this process can be seen as a dynamic compilation of a processing pipeline of visual routines. Assume that this pipeline is to locate an object candidate, determine its mask using a segmentation procedure and measure the object candidate's color. As the system knows that the searched object is "red", it can bias the saliency computation with the prototypical sensory representation of "red". This sensory representation is attached to the property node as described before. The triggered saliency computation returns the location of an object candidate together with a coarse pixel size. To store and combine the acquired information, the measured properties are stored in the short-term memory (see Fig. 4.8 (3)). The representation in the short-term memory has exactly the same structure as in the long-term memory. In the following the mask of the object candidate is segmented and the color of the object candidate is determined. The results of both measurements are also stored in the short-term memory (see Fig. 4.8 (4)). In a last step, the measured color of the object candidate is compared to the one activated by the task. In this example, the color matches for the first candidate and the task is solved. In the next section, the processing performed by the attention control (dependency resolving and determining which properties to measure) is presented.

## 4.4. Attention Control

The attention control mechanism is probably the most important part of the system, as this part selects the elements of the scene that are perceived. For humans, this process is both selective in the spatial as well as in the feature domain according to the experiments reviewed in chapter 2. As the process of guiding attention spatially is well researched, the focus in this thesis is primarily on the guidance of attention in the feature domain. Subject of this section is mainly the

question how the system can determine which properties of an object should be measured. Along with the question of which properties to measure, the question on how to organize the acquisition process arises. The last section has shown that the knowledge about how to measure an object's property is stored in the system's memory. However, it is yet unclear in which order the system should measure which property. This is also tackled here. First, the process of modulating the visual routines is described briefly. Afterwards, a scheduling algorithm is proposed that determines which properties to measure when, based on the system's long-term memory. Finally, the assumptions made about the structure of the long-term memory are mentioned.



## Visual Routine and Saliency Modulation

Predictions about object properties can be made by inheriting nodes from the long-term memory as described in section 4.2. These predictions can now be used to modulate the system's processing. As each property node stores a link to its sensory representation, the sensory values $value(p \in P)$ can be recovered. In the current system implementation, the sensory values are stored as k-means approximations except for spatial information which is stored as a map. Here, it is not necessary to distinguish between nodes inherited from the long-term memory and nodes actually measured, as an inherited property $p_{child}$ delivers the value of the parent node $p_{parent}$ transparently $value(p_{child} \in P) = value(p_{parent} \in P | \exists (p_{parent}, p_{child}) \in E_{specializesTo})$. Here, $(p_{parent}, p_{child})$ denotes a tuple of property nodes in the graph connected by an edge. The recovered sensory values are now sent to the visual routines or the saliency computation to modulate the input features both in the spatial and feature domain (see chapter 3 for the system's architecture).

A special case of modulation is the spatially unbiased saliency modulation. In this case, there is no spatial information about the target

object available. However, the locations of other objects should be suppressed to reduce the search space. This mechanism is known as inhibition of return [Itti and Koch 2000]. To generate an inhibition of return map, the locations of all objects in the short-term memory are accumulated and inhibited. The accumulated map is then sent to the saliency computation.

## Parsing the Dependency Graph

As described above, it is possible to represent functional dependencies in the relational memory used here. Furthermore, modifiers are used to account for algorithmic needs of the visual routines bound to property nodes. However, up to now, only direct dependencies are modeled. But how can this information be used to organize the acquisition process in a way that an arbitrary visual routine can be triggered on demand? By parsing the direct dependencies shown in Fig. 4.5, the complete unrolled dependency graph shown in Fig. 4.9 is constructed. Unrolling

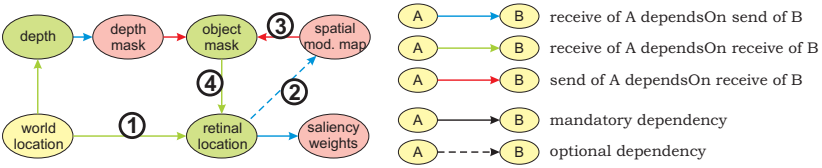


**Figure 4.9.:** Unrolled dependency graph for receiving the three dimensional location of an object. Only "dependsOn" relations are shown here. Dashed lines are optional dependencies, nodes marked in red and green need to perform a send or receive operation respectively. Edge 4 leads to a circular dependency.

the dependency graph can be achieved by applying a depth-first search algorithm [Sedgewick 1988] on the graph constructed by the direct dependency definitions. This algorithm assumes that the parsed graph is a tree. That is, it does not contain cycles. For doing so, a node state marking a node as either "resolving", "visited", "invalid data"

or "valid data" is introduced. The first state means that the node is currently resolving its dependencies. The second state means that the resolving process of the node's dependencies is finished. The third state means that the node has not been visited yet and does not contain up-to-date data. The last state means that the node has not been visited yet, but its data is up-to-date. The last two states are introduced to reuse already acquired data. When looking at the edges two, three and four of Fig. 4.9, one can see that these relations form a circular dependency, as receiving the retinal location depends on sending the spatial modulation map (2), which depends on receiving the object mask first (3). However, receiving the object mask in turn depends on receiving the retinal location (4). This circular dependency violates the tree assumption that was made to be able to unroll the dependency graph. The circular dependency can be detected by marking the nodes as "resolving" when entering them. As soon as the operation on the node is completed, the state is changed to "visited". When entering a node which is already in the state "resolving", a circular dependency is detected. Once detected, the circular dependency needs to be removed to "treeify" the graph. This is possible if one of the edges leading to the circle is cut. But which edge can be cut without breaking the underlying algorithms? Here, the mandatory and optional modifiers introduced earlier come into play. Edge number two of the graph shown in Fig. 4.9 is optional, which means that this dependency is not absolutely necessary for the underlying algorithm. When cutting the graph at edge two, the underlying algorithm is still functional and the circular dependency is resolved. By doing so, the graph is converted to a tree and the depth-first search algorithm can be used. If no optional edge exists, the dependency resolving fails. However, this case does not exist in reality as it implies an iterative algorithm without initial condition.

By combining the depth-first search and the cyclic dependency handling the following update procedure for nodes applies. As the resolving is a recursive problem, a recursive algorithm is chosen, which only works on the subgraph $G_{depend} = (P, E_{dependsOn})$. Here, the property nodes

$P$ are $P \in V$ and the dependency edges $E_{dependsOn}$ are $E_{dependsOn} \in E$. When triggering the measurement of a certain property node the following update procedure is executed:

**Procedure *UpdateNodeValue*:**

**(a) Check the ability of the node to run**

(1) Check the current node $p_{self} \in P$ for valid data $state(p_{self}) \stackrel{?}{=}$ "valid data". If $p_{self}$ already has valid data, skip any operation and ***return*** success.

(2) Check if the current node $p_{self}$ was already visited and thus indicates a cyclic dependency $state(p_{self}) \stackrel{?}{=}$ "resolving". If a cyclic dependency is detected, ***return*** the corresponding error.

(3) Set the state for the current node $p_{self}$ to "resolving" $state(p_{self}) =$ "resolving".

**(b) Updating dependencies**

(1) Get the list with all dependencies for the current node $P_{depend} = \{p \in P | (p_{self}, p) \in E_{dependsOn}\}$.

(2) For each dependency (child node) $p_c \in P_{depend}$ do:

(2.1) Call the update procedure on the child node UpdateNodeValue($p_c$).

(2.2) If the return code contains a cyclic dependency error and the dependency is mandatory $type((p_{self}, p_c) \in E_{dependsOn}) \stackrel{?}{=}$ "mandatory", ***return*** the received error.

(2.3) Remove the child node from the dependency list $P_{depend} = P_{depend} \setminus p_c$ and process the next dependency in the list $P_{depend}$.

**(c) Execute current node's operation**

(1) ***Call*** the requested operation $operation((p_{self}, p_c) \in E_{dependsOn})$ on the current node $p_{self}$. Store the sensor data locally, if a receive operation was requested $operation((p_{self}, p_c) \in E_{dependsOn}) \stackrel{?}{=}$ "receive".

(2) Mark the data of the current node as valid $state(p_{self}) =$ "valid data".

(3) ***Return*** success.

In the pseudo-code algorithm the node's state is set to "valid data" when the data of the node is updated. This is done to reduce the computational demand and reuse as much information as possible. When entering a node with the state "valid data", the depth-first search parsing algorithm knows that the node does not need updating. This leads to a shrinking of the dependency graph, as the dependency subgraph of this node is not processed. If the information stored in a node has not been acquired yet or is outdated, the node state is set to "invalid data". The state transition between "valid data" and "invalid data" can be based on arbitrary criteria, like the elapsed time since the data was acquired or the confidence of the measurement. When applying a criterion that is based on time or the input data, a dynamic pruning of the dependency graph can be observed, as the graph might look different for each parsing pass. Read more on this idea in chapter 6. The ability of dynamic pruning distinguishes the algorithm proposed here from similar approaches used in the domain of compiler construction such as [Ferrante et al. 1987] and [Ballance et al. 1990] or the rule-based approach proposed in [Langley et al. 2009].



To ensure that the algorithmic need of the underlying algorithm is taken into account, an algorithm was proposed to parse the relational memory graph (or rather the dependency subgraph) to resolve the dependencies of a certain node automatically. This makes sure that all information required by the underlying algorithm is available. Here, the depth-first search algorithm is extended to handle circular dependencies and take the validity of the node's data into account. By doing so, a dynamically pruned graph is created with each parsing pass. This distinguishes the framework from blackboard architectures, as these approaches do not represent dependencies between processing modules at all. Currently, the dependencies between the properties are constructed by hand and reflect the algorithmic needs of the visual routines. In future work, learning these relations as suggested in [Blum and Furst 1997] might be incorporated. The generic graph structure of the relational memory leads to a framework that is able to consistently represent both the

relational knowledge of objects and properties as well as the knowledge about the measurement processes and their dynamics. Contrary to classical AI approaches, in this memory framework a persistent anchoring of the property nodes is achieved by storing a link to the sensory representations. Furthermore, the updating of these sensory representation can be controlled by the system because of the stored procedural knowledge. In the next section, the relational framework is used to define a memory architecture for the system.

## Attention Control and Scheduling

The attention control mechanism proposed here is the key element of the system. So what is the role of this mechanism? As mentioned earlier in this thesis, attention is understood as a selection process deciding where to look and which details to store about the focused object. So the problem is twofold. First, there is a spatial aspect of attention, namely to locate object candidates in the current scene. A lot of work has been done in this direction, the probably best known one is [Navalpakkam and Itti 2007]. The authors state that modulating low-level features using knowledge about an object can speed up visual search. In the proposed system this modulation is done as described in chapter 3 and in the previous section. Once focusing on an object candidate, the system needs to assure that the attended object has all properties requested by the given task. This leads to the second, not well-researched aspect of attention: attention in the feature domain. The system needs to acquire *only the information relevant* for solving the current task. But how does it know what is relevant? For tasks already containing a hint on which property is relevant, the system can simply trigger the respective visual routine. If the task is to "find a tiny object", the system immediately knows that it needs to analyze the size of an object candidate.

Figure 4.10 illustrates the process of searching for a "tiny" object and the process of searching for the "toy car". As shown in Fig. 4.10,

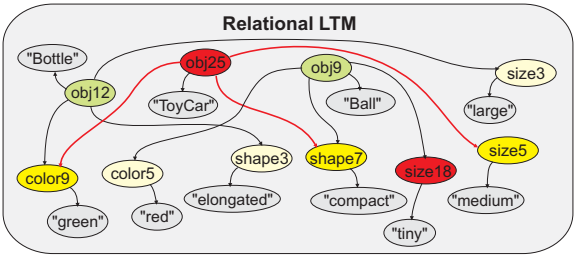


**Figure 4.10.:** The system searches for a "tiny" object and the "toy car", triggered by the activation of the corresponding nodes in the long-term memory (nodes marked red).

the searched properties or objects are activated in the long-term memory. When only a property is activated, the visual routine bound to this property node is called on each object candidate. However, for finding a specific object the procedure is more complex. In order to keep the computational and storage demand low, the goal is to find the minimal set of measurements ensuring that the attended object is the searched one. This way, the amount of information that needs to be stored in the short-term memory as well as the computation time are minimized. As mentioned earlier, the approach proposed in this thesis uses the system's long-term memory to determine characteristic properties of the searched object. Please note that the discriminative power of a certain property strongly depends on concurrently active object hypotheses. In Fig. 4.10, the system has to search a "toy car". This object has attached the property nodes "green", "medium" and "compact" (see red arrows in Fig. 4.10). To search for the object, first a suited object candidate needs to be found in the current scene. The saliency computation is triggered, using the properties of the searched object like color and size to bias the input features. After

finding an object candidate, the system must decide on which property it wants to focus. Given that it actually measures the color "green", there are two valid hypotheses ("bottle" and "toy car"), for the size "medium" only one hypothesis remains ("toy car") and for the shape "compact" two hypotheses remain (see Fig. 4.10). So the information gain is highest for the size measurement, as it reduces the set of possible interpretations to only one. In other examples the measurement of more than one property is necessary to solve the task. Nevertheless, this example shows the principle of the algorithm: Find the property of the searched object $o_s$ that minimizes the set of remaining interpretations or hypotheses for the attended location. If more than one measurement minimizes the set of hypotheses, a second factor comes into play, the cost of a certain measurement. Currently, the computation time is approximated and the cost of a certain visual routine is set to that approximated time. Again, the algorithm only works on a subgraph of the memory $G_{scheduling} = (V, E_{hasProperty})$ consisting of the object nodes $O$, property nodes $P$ with $V = O \cup P$ and "hasProperty" edges $E_{hasProperty} \in E$. The measured value $\mathbf{v}$ is then compared to the value $\mathbf{c}_i = value(p_t)$ of all property nodes $P_t = \{p \in P | \exists (p, p_{type}) \in E_{inheritsFrom}\}$ in memory having the same type $p_{type} = \{p \in P | \exists (p_i, p) \in E_{inheritsFrom}\}$ as the measured property $p_i$, in this example the type is size. The following function is used to determine the activation $a(p_t)$

$$\mathbf{c}_i(p_t) = value(p_t) \tag{4.10}$$

$$\hat{a}(p_t) = e^{-\frac{1}{2\sigma^2}\|\mathbf{c}_i(p_t)-\mathbf{v}\|^2} \tag{4.11}$$

$$a(p_t) = \begin{cases} 0, & \text{for } \hat{a}(p_t) < \xi \\ \hat{a}(p_t), & \text{else} \end{cases}, \ \forall p_t \in P_t \ , \tag{4.12}$$

where $\xi$ is the threshold for the activation. After resetting all activations in memory to zero, the scheduling of the visual routines works as follows:

**Procedure *SearchObject*:**

## (a) Locate an object candidate

(1) Activate the searched object $o_s \in O$ and collect its attached properties $P_s = \{p \in P | (o_s, p) \in E_{hasProperty}\}$.

(2) Locate an object candidate $o_c$ using the saliency map and initialize the set of remaining hypotheses to all objects $O_r = O$.

## (b) Schedule visual routines

(1) While $P_s \neq \emptyset$:

(1.1) Find all remaining object hypotheses $O_h$ sharing properties with the searched object $O_h = \{o \in O_r | \exists p \in P_s : (o, p) \in E_{hasProperty}\}$.

(1.2) Calculate the discriminative power $d_i$ against the remaining hypotheses: $d_i = |\{o \in O_h | \exists (o, p_i) \in E_{hasProperty}\}|^{-1}, \ \forall p_i \in P_s$.

(1.3) Trigger the visual routine on the object candidate $o_c$ for the most discriminative property $p_i : d_i \leq d_j \forall j$ by calling UpdateNodeValue($p_i$). If multiple properties minimize the set, select the one with the least cost. At this point, the dependency parsing described in section 4.4 is used. Remove the selected property from the set $P_s = P_s \setminus p_i$.

(1.4) Find all property nodes $P_t = \{p \in P | \exists (p, p_{type}) \in E_{inheritsFrom}\}$ in memory of the same type $p_{type} = \{p \in P | \exists (p_i, p) \in E_{inheritsFrom}\}$ as the measured property $p_i$.

(1.5) Calculate the activation $a(p_t \in P_t)$ for all property nodes of the same kind $p_{type}$ (e.g. size) using the activation function shown in Eq. 4.12.

(1.6) Propagate the activation $a$ of all activated property nodes $p_m$ with $p_m : a(p_m) \geq \xi$ to their attached objects $O_m = \{o \in O | \exists (o, p_m) \in E_{hasProperty}\}$ using $a(O_m)_t = a(O_m)_{t-1} \cdot a(p_m)$. Determine remaining objects $O_r = O_h \cap O_m : a(O_m)_t \geq \xi$. If the searched object is rejected $o_s \notin O_r$, locate another object candidate (step (a)). If $O_r = \{o_s\}$, the object is found, go to step (c). Otherwise continue with the next property as the object candidate is still ambiguous.

## (c) Link the object to the long-term memory

(1) If the object $o_s$ was found, create a link to the remaining hypothesis $o_r$: $o_r \xrightarrow{\text{specializesTo}} o_s$.

It is obvious that the number of required measurements strongly depends on the task, the searched object itself and the knowledge stored in the long-term memory. Furthermore, a certain structure is assumed in the long-term memory, namely that objects sharing a common property all connect to this property. However, the clustering of these object properties might not be easy in real-world data. The investigation on this issue is part of the future work described in chapter 6.

## 4.5. Discussion

In the first part of this chapter a relational, graph-based memory framework to consistently represent the knowledge about both the world and the sensory apparatus of the system was proposed. This is a major difference to other memory architectures not addressing the representation of the system's sensory apparatus. Here, a novel approach of also representing algorithmic dependencies between sensors was presented which ensures the functionality of the underlying visual routines. The parsing of this dependency graph is done dynamically and automatically by the system, handling circular dependencies and incorporating sensor information that was already acquired about the current scene. By doing so, a dynamic pruning of the dependency graph is implemented, influenced by the state of the system. As a fundamental difference to standard AI approaches, the concept of anchoring sensory nodes in the respective sensory representation is proposed.

In a second step, the memory architecture of the system with its three layers was presented. The first layer describes the structure of an object and the relations to its properties independently from the system implementation or available sensors. A second layer then inherits this object structure and adds the binding to sensors, the functional knowledge of the system and the dependencies between the sensors. By doing so, the memory architecture is connected to the actual system's implementa-

tion and sensors. The third layer of the system implements a short-term and long-term memory that inherits information from the second layer. Here, the short-term and long-term memory share a common structure and sensory definition. In combination with the "inheritsFrom" relation an efficient implementation of a prediction-confirmation loop is possible, where the long-term memory predicts sensory properties that can be measured by a visual routine and stored in the short-term memory. Furthermore, the common structure allows for an easy transfer of information between the two memories.

Finally, an algorithm was presented which is able to actually use the information stored in the long-term memory (both relational world knowledge and functional knowledge) in order to determine the information that needs to be acquired to solve a given visual search task. The decision of this scheduling algorithm is based on the task, the properties of the object to search and the content of the long-term memory. Here the knowledge of the system about its functional relations and dependencies is used to ensure the function of the underlying visual routines. The property information anchored in the sensory representation as described before is used to bias the underlying processing of the system. The goal of the presented algorithm is to identify the minimal set of measurements required to solve the given task. To do so, the short-term memory is incorporated to reuse as much information as possible. In the next chapter, a proof-of-concept implementation of a system described in chapter 3 and chapter 4 will be evaluated.

# 5. Experiments

In order to evaluate the performance of the concept, a system following the architecture and control schema presented in this thesis was imple-

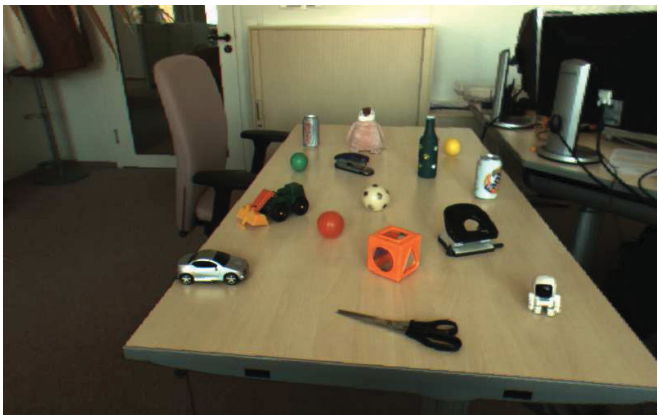

**Figure 5.1.:** This scene shows the system's visual input during the experiments.

mented. The results of the conducted experiments using this proof-of-concept implementation are presented in this chapter. To evaluate the system, a static scene was recorded using a stereo color-camera pair. The left camera image of this recorded scene is shown in Fig. 5.1.

In the last chapters it has become clear that the system's memory plays an important role during the interaction of the system with the scene. In the current implementation, the system is not able to learn contents

of the long-term memory. Therefore, the content was created a priori using the dataset shown in Appendix F. Clusters of the properties color, physical size and shape that are used by the system were created using k-means (see Appendix F for details). The resulting content of the long-term memory is shown in Fig. 5.2. Each object in the long-term

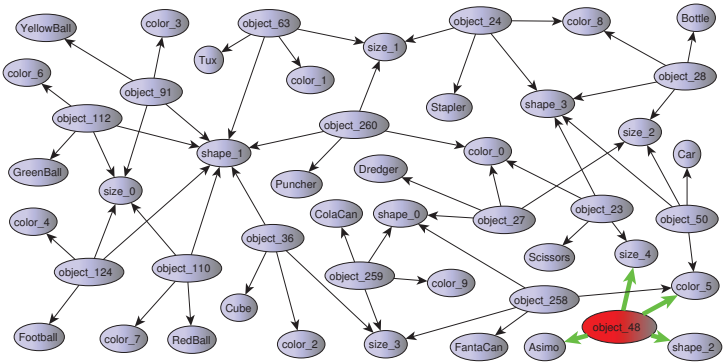


**Figure 5.2.:** The long-term memory shown here contains all objects known to the system. Each object, like the "Asimo" node marked red, is attached (green arrows) to four properties: color, size, shape and a label.

memory has four property nodes: color, physical size, shape and a label. Only the first three properties are measurable, the label is just used to ease the identification of the objects in the input scene for the human observer. To measure the properties, the visual routines discussed in chapter 3 are implemented. A detailed listing of the parametrization of the different algorithms and the control part of the system can be found in Appendix E. In the following, different tasks are given to the system. While the system tries to solve these tasks, the specificity of the resulting representation is evaluated. At this, the system is running continuously, without resetting the system's memory. Parts of these results were already published in [Eggert et al. 2007, Rebhan et al. 2008; 2009a] and [Rebhan et al. 2009b].

## 5.1. Spatial Attention

As already mentioned in chapter 3, the spatial attention and its modulation play an important role during the search for object candidates relevant for the current task. Therefore, the experiments conducted to evaluate the performance of the spatial saliency computation and its modulation are described in this section.

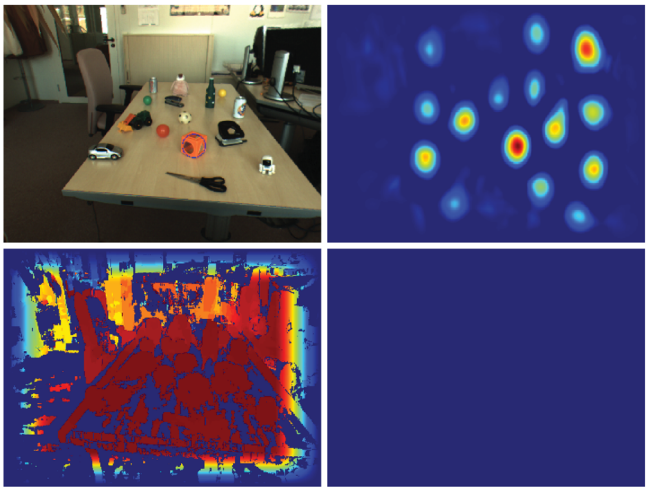

**Figure 5.3.:** The input image (top left) is preprocessed and the features (color and saturation) are computed. Based on these unmodulated features the saliency map is calculated (top right). The spatial modulated map (bottom left) and the modulated color map (bottom right) are not used.

Figure 5.3 shows the input scene and the corresponding unmodulated saliency in the top row. The bottom row shows the two possible modulation maps available to the system. As can be seen in Fig. 5.3, the saliency map has multiple peaks mostly covering the objects on the table. However, when searching for a specific object in the scene, like

the cube, there are too many salient locations that might distract the focus of attention from the searched object. In Fig. 5.4 the color feature channel is biased using the color orange ($\mu = (0.85, 0.39, 0.17)^T$, $\sigma = (0.2, 0.2, 0.2)^T$) as stored in the long-term memory of the system. The resulting modulated color map $\mathbf{t}_{color}$, calculated by using Eq. 3.6,

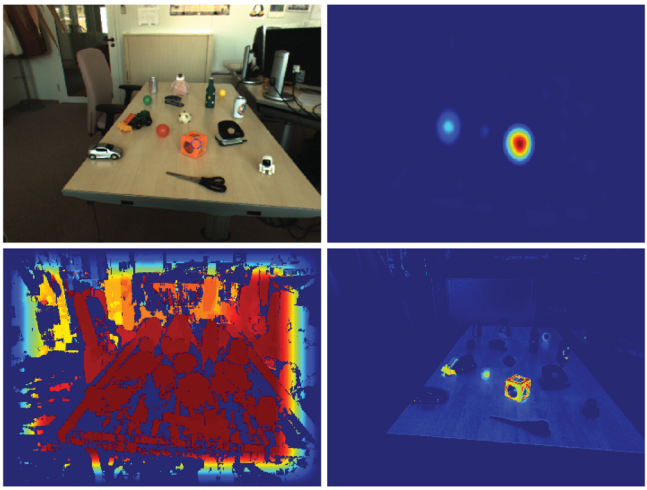

**Figure 5.4.:** The color feature of the input is modulated (bottom right) with a Gaussian bias $\mu = (0.85, 0.39, 0.17)$, $\sigma = (0.2, 0.2, 0.2)$ (orange), leading to a biased saliency map (top right). The spatial modulation map (bottom left) is unused.

is shown in Fig. 5.4 (bottom right). To suppress salient locations introduced by unmodulated feature channels, the weighting of those feature channels is set to zero, whereas the weight of the modulated feature channel is set to one. This leads to the specified top-down weight vector $\mathbf{w} = (0.0, 0.0, 0.0, 0.0, 1.0)^T$ for the channels red, green, blue, saturation and the modulated color map. The weighting is performed as shown in Eq. 3.8. The resulting saliency map (top right of Fig. 5.4) is much more specific for the searched cube, which is known to be or-

ange. To measure the specificity of the saliency map $\mathfrak{s}$ with respect to a searched object, two measures for the signal-to-noise ratio are defined

$$SNR_{abs} = 10 \log \frac{\sum_{x,y} \mathfrak{s}(x,y) \cdot m_{gt}(x,y)}{\sum_{x,y} \mathfrak{s}(x,y) \cdot (1 - m_{gt}(x,y))} \quad (5.1)$$

$$SNR_{rel} = SNR_{abs} + 10 \log \frac{\sum_{x,y}(1 - m_{gt}(x,y))}{\sum_{x,y} m_{gt}(x,y)} \,. \quad (5.2)$$

Here, $\mathbf{m}_{gt}$ is the binary ground-truth mask $m_{gt}(x,y) \in \{0,1\}$ for the area covered by the searched object. The activations $\mathfrak{s}(x,y)$ outside of this area $(1-m_{gt}(x,y))$ are interpreted as noise. $SNR_{abs}$ measures the relation between the absolute activation within and outside the object area. Contrary to this, $SNR_{rel}$ normalizes this absolute ratio by the inner and outer area of the object respectively. The second measure was introduced for comparing the the results to those presented in [Navalpakkam and Itti 2007], where the same equation is used. As in

| | **Color cluster** | | | | |
|---|---|---|---|---|---|
| **Cluster** | 0 | 1 | 2 | 3 | 4 |
| SNR unbiased [dB] [1] | -8.0 | -21.3 | -9.7 | -35.2 | $-^*$ |
| SNR biased [dB] [1] | -6.7 | -10.8 | 1.1 | -15.0 | -15.7 |
| SNR unbiased [dB] [2] | 8.9 | 3.0 | 11.6 | -3.7 | $-^*$ |
| SNR biased [dB] [2] | 10.2 | 13.5 | 22.4 | 16.5 | 11.8 |
| **Gain [dB]** | 1.3 | 10.5 | 10.8 | 20.2 | $-^{**}$ |
| **Cluster** | 5 | 6 | 7 | 8 | 9 |
| SNR unbiased [dB] [1] | -10.6 | -33.2 | $-^*$ | -16.0 | $-^*$ |
| SNR biased [dB] [1] | -5.4 | -12.9 | -5.0 | -12.3 | -15.8 |
| SNR unbiased [dB] [2] | 8.6 | -2.3 | $-^*$ | 7.2 | $-^*$ |
| SNR biased [dB] [2] | 13.7 | 18.0 | 22.0 | 11.0 | 12.4 |
| **Gain [dB]** | 5.1 | 20.3 | $-^{**}$ | 3.7 | $-^{**}$ |

**Table 5.1.:** This table shows the signal-to-noise ratio of the biased and unbiased saliency map for different color clusters. [1] values are calculated according to Eq. 5.1. [2] values are calculated according to Eq. 5.2. Entries marked with * had no signal, the SNR is $-\infty$ and the gain $-^{**}$ is theoretically $\infty$.

the current implementation, only the modulation of the color feature is possible, the ground-truth mask $\mathbf{m}_{gt}$ covers all objects attached to one color cluster in the long-term memory of the system. The signal-to-noise ratio for the biased and unbiased saliency map with respect to the different color clusters is shown in Table 5.1. As Table 5.1 shows, the signal-to-noise ratio is strongly improved by using the known colors of the objects to bias the saliency map. For the color clusters four, seven and nine, the biasing even enables the searching in the first place, as the unbiased saliency map does not contain activation for those clusters. The measured SNR is similar to foreground-biased results presented in [Navalpakkam and Itti 2007]. However, as the table in [Navalpakkam and Itti 2007] shows, the SNR can be improved further by incorporating knowledge about the background statistics. This is part of future work.

Despite the fact that the signal-to-noise ratio is a good measure for the "concentration" of the activity in the saliency map within the area of the searched object, it does not take the read-out characteristics into account. That is, in current spatial attention systems and also in this system, the next focus of attention is usually determined by applying a winner-takes-all algorithm (WTA). These algorithms concentrate on the strongest peak in the saliency map, rather than taking the distribution of activity into account. To perform subsequent fixations, the currently focused object is inhibited using the spatial modulation map. In order to consider the read-out mechanism, another quality measure for the saliency is introduced in this thesis, the "hit rate". Here, an object is marked as "hit" if the focus point $x_{FoA}, y_{FoA}$ selected by the WTA algorithm is within the ground-truth object area $m(x_{FoA}, y_{FoA}) \stackrel{!}{=} 1$. To calculate the hit rate, first the number of subsequent fixations is measured, required by the system to focus on all objects of a certain color cluster at least once. The hit rate is then the ratio between the minimal number of fixations, i.e. the number of objects in the color cluster and the measured number of fixations required by the system to focus at least once on each object. Table 5.2 shows the number of objects for each color cluster, the actually required number of fixa-

tions and the resulting hit rate of the system. Here, only the figures for the biased saliency map are considered. As Table 5.2 shows, the

| Cluster | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ∅ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Objects | 3 | 1 | 1 | 1 | 1 | 3 | 1 | 1 | 2 | 1 | 1.5 |
| Fixation | 6 | 2 | 1 | 1 | 3 | 4 | 1 | 1 | 5 | 2 | 2.6 |
| Hit rate | 0.5 | 0.5 | 1.0 | 1.0 | 0.33 | 0.75 | 1.0 | 1.0 | 0.4 | 0.5 | 0.7 |

**Table 5.2.:** This table shows the number of search steps necessary to find all objects of a certain color cluster. The hit rate quantifies the average number of objects found per fixation.

system requires less than two fixations in a scene to locate an object of a specified color cluster. One can also see that for the color clusters four (football) and eight (stapler and bottle), the performance drops dramatically. Figure 5.5 shows the saliency map and the biased color map of those two clusters. The visual scene used for the experiments

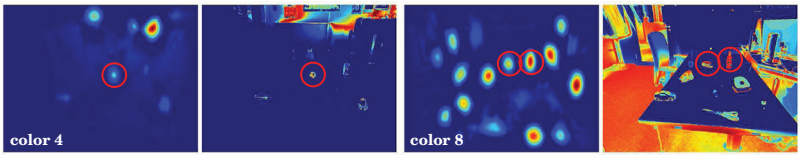


**Figure 5.5.:** The saliency map (left) and the biased color map (right) are shown for the color clusters 4 and 8. The searched objects are marked with red circles.

has similar colors for the searched object and in the background for color cluster four, as shown in Fig. 5.5. This leads to a high activation of the saliency map for the background, which is much stronger than on the actual object (marked with a red circle). Consequently, two fixations are made on the background before the searched object is fixated. Contrary, to this spatially concentrated distractor, multiple objects exist in the scene that have a similar color to color cluster eight. The biased color map in Fig. 5.5 shows a strong activation for the two searched objects. However, due to the lateral dynamics of the

saliency map the activation of the stapler is weakened relatively to the distractors. In turn, the bottle is found rather quickly (second fixation), whereas the stapler requires another three fixations to be found. The extensive illustration of all biased color maps and their corresponding saliency maps can be found in Fig. G.1 of Appendix G. Based on these results, the next experiment is concerned with the organization of the measurement process using the procedural knowledge of the system.

## 5.2. Functional Dependency Modeling

The system architecture proposed in this thesis is designed to allow for a flexible combination of different visual routines to create more complex operations. With the freedom to do so, the need to structure these combinations arises. In chapter 4 an algorithm using the procedural knowledge of the system is proposed, to resolve the algorithmic dependencies between visual routines and thus sensory nodes. This knowledge is stored in the long-term memory (see chapter 4 for details). The expanded graph used throughout the experiments can be seen in Fig. E.1 of Appendix E. During the resolving process, the proposed algorithm is able to reuse the information already acquired about the scene for further measurements. In this section the efficiency of this algorithm is evaluated by extensively comparing the measurement of visual properties with and without reusing the already acquired information. Figure 5.6 illustrates the resolving process using the example of color for different points in time. The illustration shows that in the first time steps the nodes required to measure the color of an object are added. When looking at the state for time $t = 10$, three major measurement steps can be seen:

1. localization of an object candidate consisting of the nodes `retinallocation`, `saliencymask` and `saliencyweights`

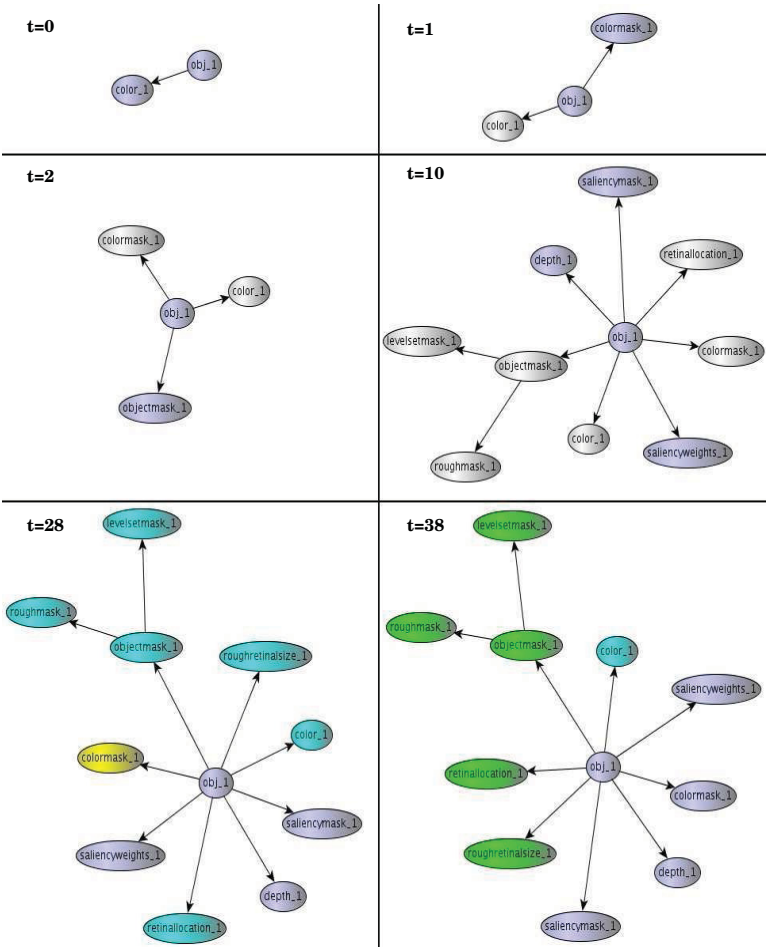


**Figure 5.6.:** This shows the resolving of algorithmic dependencies while measuring an object's color. A node's color signals its current state. Yellow means the node is about to send its data, green means the node received data, blue means the node was ignored / has invalid data and white means the node is waiting.

2. segmentation of the object candidate consisting of the nodes `objectmask`, `levelsetmask` and `roughmask`

3. color measurement of the object candidate consisting of the nodes `color` and `colormask`.

After finishing the addition of the necessary nodes, the visual actions of the sensory nodes are propagated to the visual routines and the dependencies are resolved recursively from the leafs of the tree back to the root. At time $t = 28$, Fig. 5.6 shows the resolved tree in the short-term memory. Now the system waits for the actual sensory information to arrive. Once new sensory information arrives, the system propagates this event through the tree. At $t = 38$ the measurement process is nearly finished, the only missing information is the color itself (marked with cyan in Fig. 5.6), all other required nodes contain valid data at this point in time (marked with green in Fig. 5.6). The complete resolving process is shown in Fig. G.2 of Appendix G.

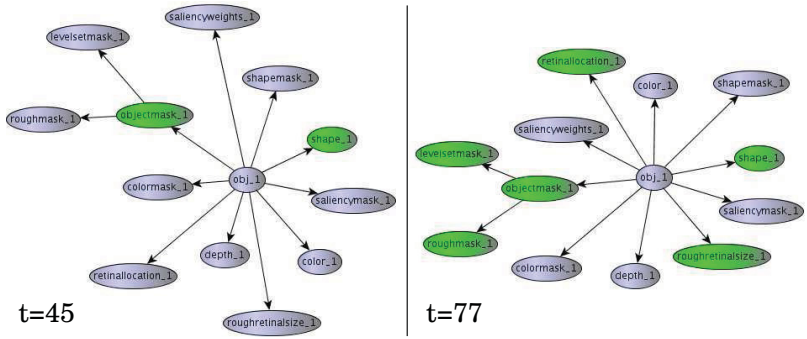


**Figure 5.7.:** When reusing the previously acquired information, only two nodes are updated (left). In case those information is ignored, six nodes need to be updated (right). Reusing previously acquired information saves 32 compute cycles.

As mentioned earlier, the system is able to reuse information acquired before, if it is still valid. Figure 5.7 shows the process of measuring the shape of an object candidate with and without reusing the information acquired during the measurement of color. Figure 5.7 clearly shows, that the number of required visual actions is dramatically reduced during incremental measurements. In this example, two nodes are updated when reusing the information, whereas six nodes need to be updated when ignoring previous measurements. The saving in processing time is even higher (6 versus 38 time steps).

Figure 5.8 compares the selective measurement of properties reusing already acquired information with the excessive measurement of properties. The overall number of measurements is independent of the or-

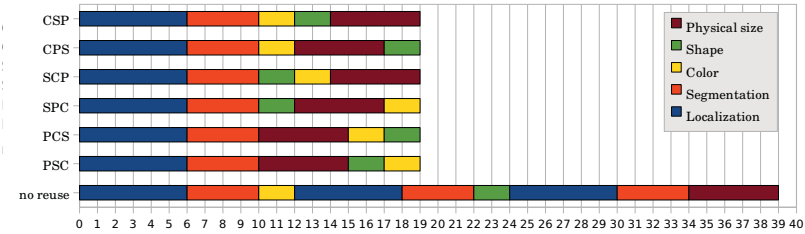


**Figure 5.8.:** Here, the number of updated nodes is shown for reusing and not reusing previously acquired information. The first six rows show incremental measurements of the properties color (C), shape (S) and physical size (P) in all permutations while reusing previous measurements. By doing so, 51% less operations are carried out compared to not reusing the measurements as shown in the last row.

der in which the properties color, physical size and shape are acquired. However, as Fig. 5.8 shows, only 19 nodes are updated when reusing the information previously acquired, contrary to 39 without reusing the information. This is a saving of 51% in processing time. The main share of measurements in the second case is not the measurement of the requested properties, but the relocation and resegmentation of the object. In the next section, the attention control mechanism including

the scheduling of visual actions is examined in more detail. Throughout all those experiments the previously acquired information is reused.

## 5.3. Attention Control and Scheduling

In the previous sections, the experiments focused on the spatial attention and the resolving of dependencies. In this section, the attention control mechanism including the scheduling of visual actions is evaluated experimentally. Different tasks are given to the system, while monitoring its internal decision processes about what to measure when.

### Task: Find a Red Object

The first task given to the system is "Find a red object!". The actual representation of the task is the activation of the color red (cluster seven) in the long-term memory. Here, the property the system has to measure (color) is already contained in the task, no further selection is necessary. Figure 5.9 shows the biased saliency map, the segmentation, the internal status of the system and the final representation of the object in the short-term memory. For detecting a red object in the scene, the system uses the activated color stored in the long-term memory to bias the saliency map. The predicted color is shown in the bottom left image of Fig. 5.9. Using this color to bias the color feature channel, the saliency map has a very strong peak on the red ball. To measure the color of the focused object, a segmentation of the object candidate is necessary (also compare to Fig. 5.6). After segmenting the object as shown in the upper right of Fig. 5.9, the system uses the resulting mask for triggering the color measurement. The measured color can be seen in Fig. 5.9 (lower left) together with the activation of the color clusters in the long-term memory. The searched color cluster (seven, indicated by a green bar in Fig. 5.9) receives the strongest activation

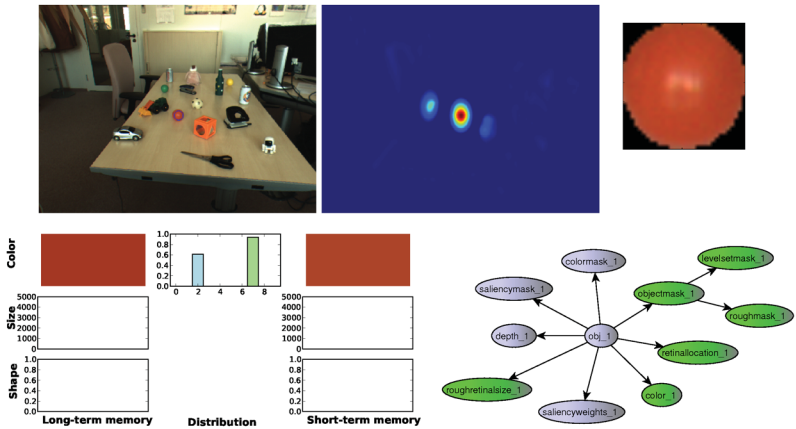


**Figure 5.9.:** Task: Find a red object. From the input (top left), the saliency map (top middle) is calculated using the predicted color stored in the long-term memory (bottom left). By segmenting the best candidate for a red object (top right), the object is found and stored in the short-term memory (bottom right).

and the system assumes the requested object is found. In the lower right corner of Fig. 5.9, the internal representation of the found object is shown. This internal representation only contains the actually measured properties like location, retinal size, object mask and color. Contrary to this, current state-of-the-art systems would store the whole feature set in memory, ignoring the demand for those features.

## Task: Find the Cube

While the previous task directly specified the property the system has to measure, the task "Find the cube!" does not give the system such a hint. Now the system has to find out on its own which property to measure in which temporal order. To do so, the algorithm proposed

in section 4.4 is used. Again, the task is represented by activating the cube (`object_36`) in the long-term memory. The top part of Fig. 5.10 shows the activated object in the long-term memory (red node) and the properties attached to the object (yellow nodes). As described in

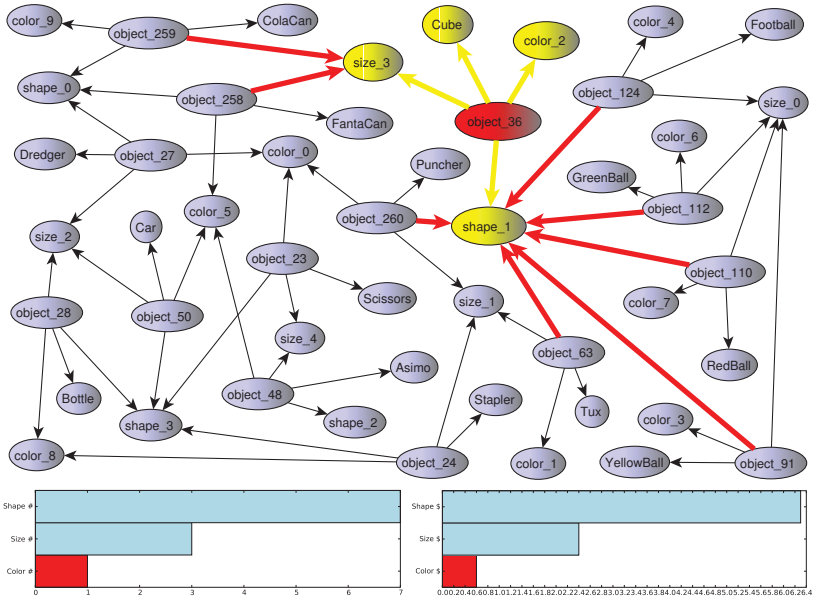


**Figure 5.10.:** Task: Find the cube. By activating the cube in memory (red node) the features corresponding to that object are activated (yellow nodes). The attention control mechanism determines the number of attached nodes (lower left) and thus the costs of the measurement (lower right) for each property. The property with the least cost (red bar), in this case color, is selected to be measured first.

section 4.4, the attention control mechanism in each step determines which property reduces the set of hypotheses for the focused object most. In the lower part of Fig. 5.10 the number of remaining object hypotheses (left) and the combined cost-gain measure (right) is shown

as determined by the system. Based on the cost-gain measure (lower right), the minimum is selected, which in this case is the color property. This is due to the fact that the system does not know any further object with the color (orange) of the cube (compare Fig. 5.10 top). To measure the color, the system first has to locate an object candidate. The saliency map, the spatial modulation map and the segmentation of the focused object are shown in Fig. 5.11. Again, the saliency map

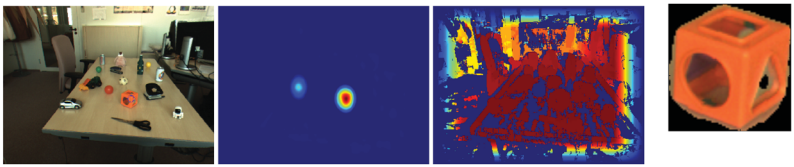

**Figure 5.11.:** Task: Find the cube. The input image (first column) is biased using the color of the cube known to the system and stored in the long-term memory. Together with the spatial modulation map (third column), which suppresses the already found "red object", the saliency map as shown in the second column is calculated. The segmentation of the object is shown in the last column.

is based on the biased color feature (compare Fig. G.1). Additional to the biasing in the color space, the object fixated in the previous experiment (red ball) is suppressed using the spatial modulation map. As the saliency map and the segmentation in Fig. 5.11 show, the first fixation is already on the searched cube. After locating the object, the color of the focused candidate is measured. The predicted object properties, the measured color, the distribution of the color cluster activation and the activation of the objects in the long-term memory are shown in Fig. 5.12. The predicted color of the cube matches the actually measured color of the object candidate very well, as can be seen in the upper part of Fig. 5.12. The color cluster of the cube (marked green) is strongly activated (0.85), which in turn leads to an equally strong activation of the cube object (lower part of Fig. 5.12). Because there is no support for any other hypothesis, the system identifies the currently focused object as the searched cube.

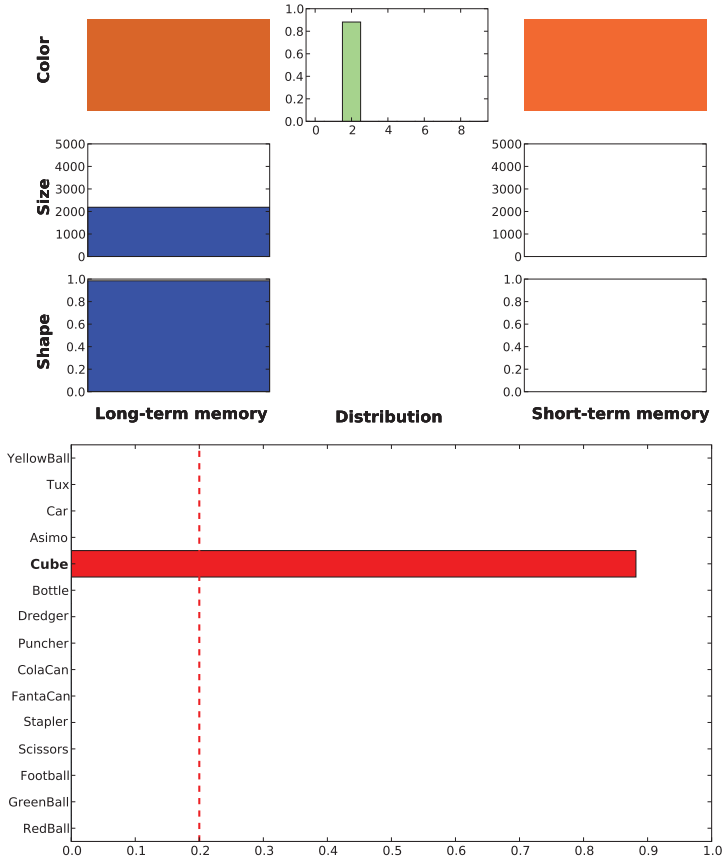


**Figure 5.12.:** Task: Find the cube. The upper figure shows the predicted properties of the cube as stored in the long-term memory, the activation distribution of different property clusters and the actually measured properties. The measured color activates only the predicted cluster. The lower figure shows the activation of all objects in the long-term memory. Only the color of the cube remains activated. As no other object is attached to this cluster, only the searched cube object remains activated. The system assumes that the currently focused object is the cube.

## Task: Find the Bottle

In the previous task the object could be identified using a single property. To increase the difficulty for the system in this experiment an object is requested that is not identifiable with a single property. As Fig. 5.13 shows, at least two hypotheses remain for each single property. Again, the searched object is marked red, the corresponding property nodes are marked yellow. At least two hypotheses remain, so the system needs to perform at least two measurements to disambiguate the

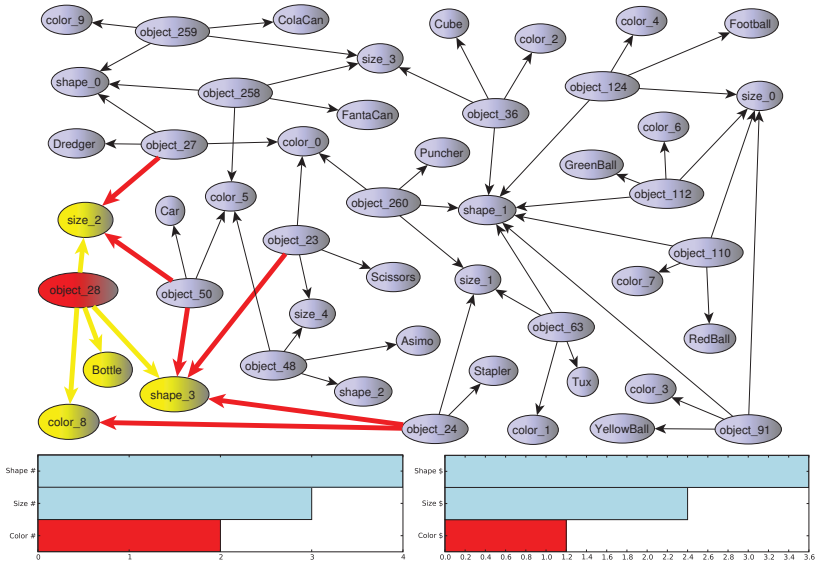


**Figure 5.13.:** Task: Find the bottle. Again the searched object is activated in the long-term memory (red node) together with corresponding features (yellow nodes). The attention control mechanism determines the number of attached nodes (lower left) and thus the costs of the measurement (lower right) for each property. At least two measurements are necessary, as the measurement with the least cost (red bar), in this case color, will generate at least two hypotheses.

focused object candidate. The system first selects the property that minimizes the set of remaining hypotheses. In this case, again the color is the best property to start with, given that the focused object candidate belongs to the predicted color cluster. The measurement process starts with localizing a suited object candidate in the visual scene. Figure 5.14 shows the saliency map, the spatial modulation

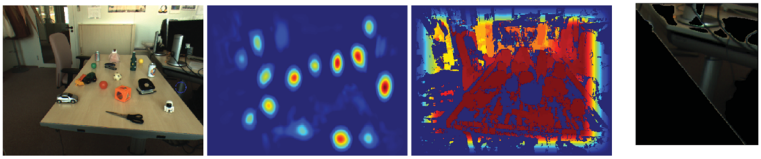

**Figure 5.14.:** Task: Find the bottle. The first object candidate detected by the spatial saliency computation is not the bottle but a part of the table.

map and the segmentation of the focused object candidate. Biasing the color map with the predicted color, leads to multiple peaks in the saliency map. This is expected, as the color cluster (`color_8`) has a hit rate of only 0.4 (see Table 5.2). As a consequence of the distributed saliency map, a "wrong" object is focused first (see the segmentation of Fig. 5.14). Note that this object (part of the table) is not known to the

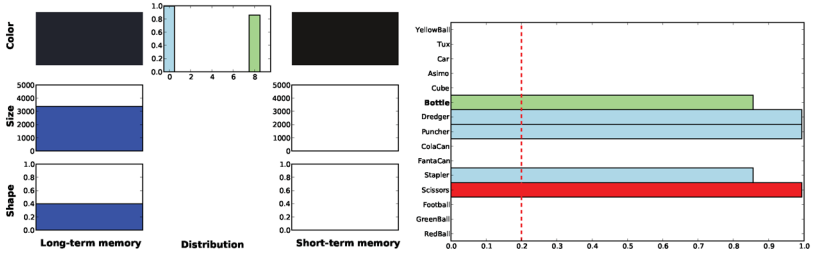


**Figure 5.15.:** Task: Find the bottle. The false object candidate has a color similar to the color of the searched bottle (left), thus the bottle remains activated (right).

system, i.e no object with corresponding properties exists in the long-term memory. Nevertheless, the system measures the color of the object candidate. The predicted color, the actually measured color and the activation distribution for both the color clusters and the objects are shown in Fig. 5.15. The measured color is similar to the predicted color (`color_8`). Furthermore, a second color cluster (`color_0`) is strongly activated. The right hand side of Fig. 5.15 shows the activation distribution among the objects in the long-term memory. Due to the strong activation of the color clusters, five object hypotheses remain for the current candidate. One can see that the predicted number of hypotheses not always matches the actual number of remaining hypotheses. To disambiguate, the system selects the physical size to be measured, according to Fig. 5.13. This property is supposed to reduce the set of remaining hypotheses best. The result of the second measurement is shown in Fig. 5.16. The measured physical size of the object candidate exceeds the predicted size of the bottle to a great extent. Because the part of the table is not known to the system, no size cluster exists that is activated by the measurement (see middle column of Fig. 5.16).



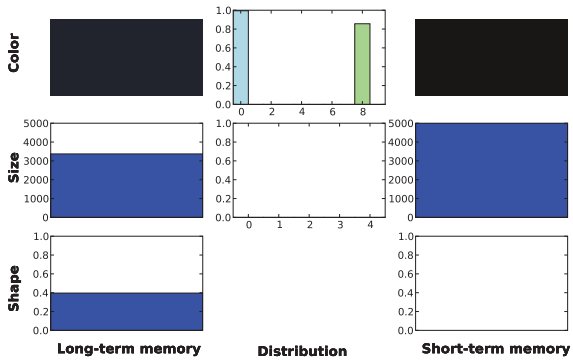


**Figure 5.16.:** Task: Find the bottle. The measured size of the object candidate does not match any known object. Thus the activation of all objects drops to zero (not shown), signaling that the focused object is not the bottle but unknown.

In turn, all object hypotheses that have existed so far are rejected by the system (activation of all objects equals zero). At this point, the system rejects the currently focused object candidate and tries to locate a new candidate. To locate a new object candidate, again the color feature is biased using the predicted color of the searched object. Furthermore, all locations of previously focused objects are inhibited

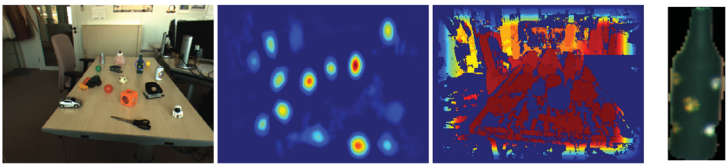

**Figure 5.17.:** Task: Find the bottle. Previous object candidates are suppressed using the spatial modulation map (third column). Now the bottle is the most salient object (second column). Finally, the segmentation of the object candidate is shown.

using the spatial modulation map, as can be seen in Fig. 5.17. This

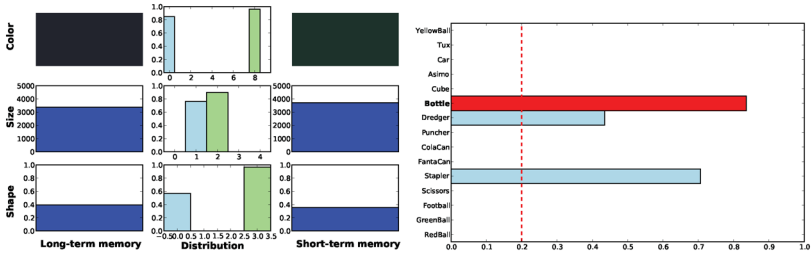


**Figure 5.18.:** Task: Find the bottle. Even though all properties are measured for the second object candidate (left-hand side), the object could not be identified as the bottle unambiguously (right-hand side). Nevertheless, as it is the most probable hypothesis, the system assumes that the bottle is found.

time the system focuses on the searched object. Again the color of the object candidate is measured as shown in Fig. 5.18. Even though this

measurement activates the color cluster of the searched object most (see the color cluster marked green in Fig. 5.18), competing object hypotheses remain. To disambiguate those hypotheses the system performs a physical size measurement, which still does not lead to a clear identification of the object. Even the additional measurment of the candidate's shape and the combination of all three measurements leads to an ambiguous result. However, after exhausting all its possible visual actions, the system identifies the object candidate as bottle, because it has the strongest activation of the remaining hypotheses.

## Task: Find the Asimo

In the two previous examples, the color was always the most discriminating property for the searched object. When looking at the representation of the Asimo in the long-term memory it becomes clear that the color is not the most discriminating property for this object. In Fig. 5.19 one can see that the shape is much better to identify the Asimo. Again, the searched object is marked red, the corresponding properties are marked yellow. The system schedules the measurement of the shape feature first, because only one hypothesis will remain, given that the focused object has the predicted property. A suited object candidate has to be located first. As the experiment conducted in section 5.1 suggests, the search process for Asimo or rather the color of Asimo will be difficult (hit rate for `color_8` is 0.33). According to this prediction, four fixations are necessary to locate the searched object. The saliency maps and the corresponding spatial modulation maps can be seen in Fig. 5.20. The segmentation results for the different fixations can be seen in Fig. 5.21. Also here, the spatial modulation map is used to inhibit previously focused objects. At each fixation the shape of the focused object candidate is measured. The predicted properties of the Asimo and the actually measured properties are shown in Fig. 5.22 for all four fixation points. As one can see, for the first three fixations, the

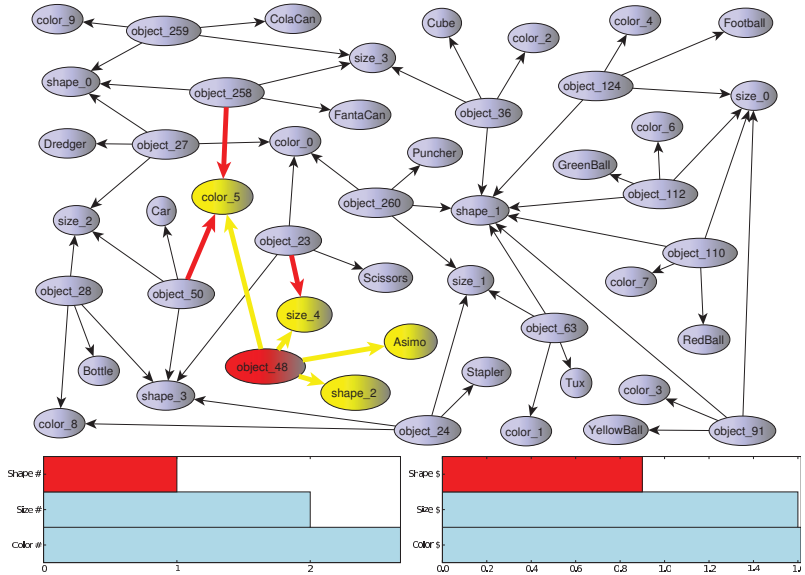


**Figure 5.19.:** Task: Find the Asimo. The searched object is activated in the long-term memory (red node) together with corresponding features (yellow nodes). The attention control mechanism determines the number of attached nodes (lower left) and thus the costs of the measurement (lower right) for each property. In this case, the measurement of the shape is discriminative for the Asimo (red bar).

shape differs too much from the predicted shape. This leads to no activation of the shape cluster attached to the searched object (`shape_2`) and in turn, the Asimo object hypothesis does not get any support (zero activation). The system interprets this zero activation as a rejection of the object candidate and triggers the location of a new one. Finally in the forth fixation, the Asimo shape cluster remains activated (marked green in Fig. 5.22). However, another shape cluster is also activated by the measurement, creating the need for the system to measure another property to disambiguate the object candidate. The system decides to

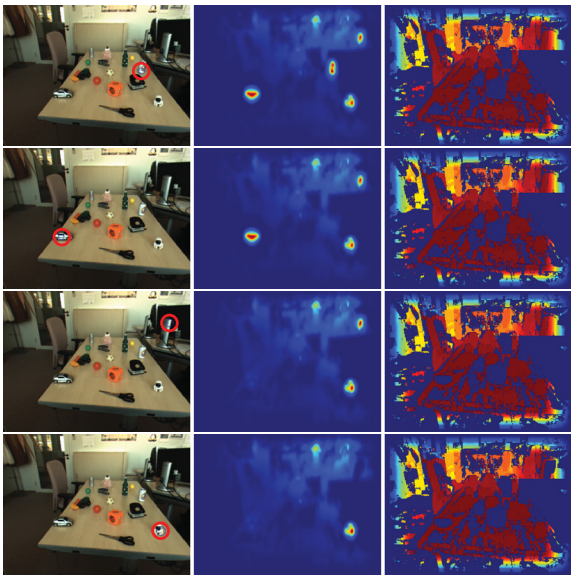

**Figure 5.20.:** Task: Find the Asimo. Four fixations are required to locate the Asimo. The focus of attention, saliency- and spatial modulation maps are shown.



measure the physical size of the object next, as this is the best choice after the shape measurement according to Fig. 5.19. By combining the shape and the physical size measurement, the object candidate can be clearly identified as Asimo (see object activation in Fig. 5.22).

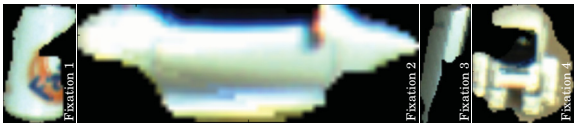


**Figure 5.21.:** Task: Find the Asimo. This shows the segmentation results on the four fixations required to find the location of the searched Asimo.

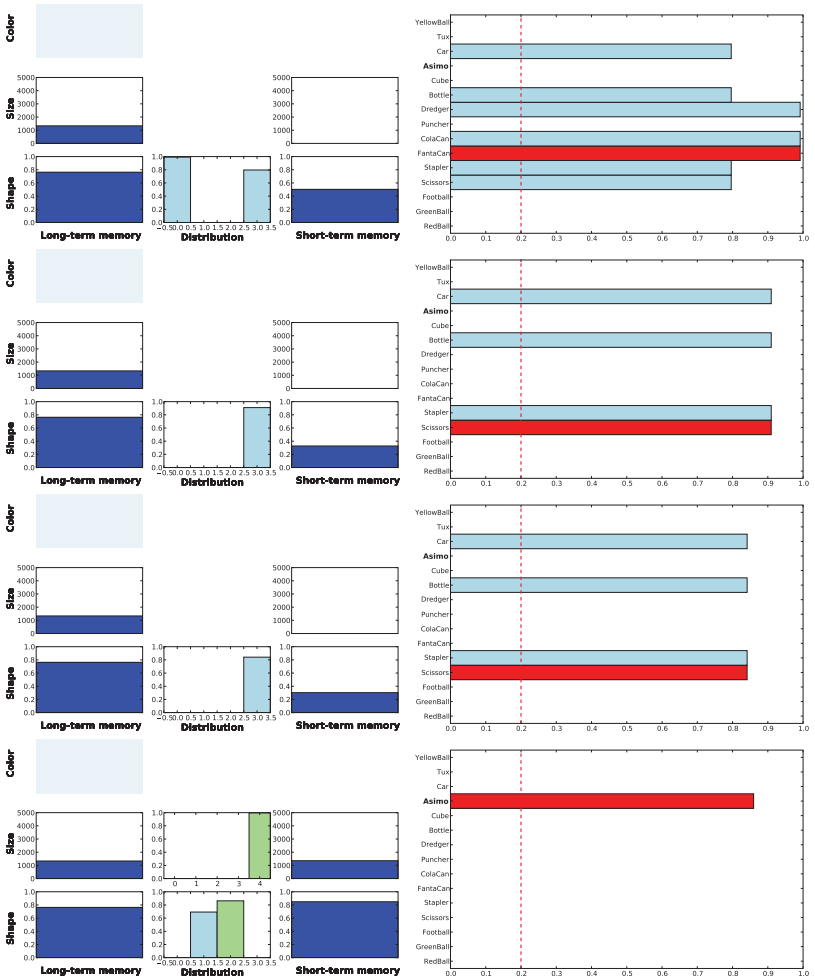


**Figure 5.22.:** Task: Find the Asimo. The first three objects are rejected by measuring their shape. Finally, the Asimo is identified after measuring the size.

## Resulting Internal Representation

After performing all previous tasks, the internal representation of the

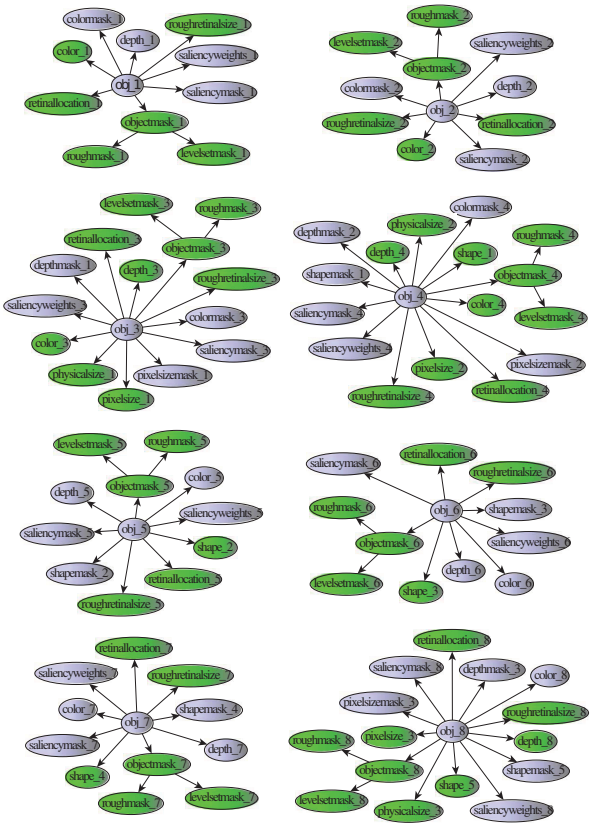


**Figure 5.23.:** After performing the previous experiments, the short-term memory contains eight objects. The identified objects (1:red, 2:cube, 4:bottle, 8:Asimo) are linked to their corresponding prototypes in the long-term memory (not shown).

system's short-term memory contains 8 objects as shown in Fig. 5.23. The first object corresponds to the searched red object, the second one to the cube, the forth to the bottle and the last object corresponds to the Asimo. Additional to these measurements, further knowledge about the objects can be inferred, because all those identified objects directly inherit their properties from the long-term memory. This allows to infer and predict object properties that have not been measured yet.

## 5.4. System Performance

In the last sections, different parts of the system were evaluated. The experiments conducted in this section cover the overall system performance. These performance measures especially focus on the savings in


**Figure 5.24.:** The scan path of the system for finding the Asimo is shown together with the properties measured at each fixation. The blue cross marks the first fixation, the red crosses mark intermediate fixations, the green one marks the last fixation. In the first three fixations only one out of three properties are measured.

terms of memory and computational load of the proposed system in this thesis as compared to state-of-the-art systems. Those systems do not selectively process object properties but rather measure all those properties at each fixation point. Figure 5.24 shows the scan path of the system together with the measured properties at each point using the example of the Asimo object. For the first three fixations of Fig. 5.24, the measurement of a single property is sufficient to reject the object candidate (compare Fig. 5.22). Finally, the Asimo is identified by measuring two properties, the color measurement is omitted.

In the following, all objects in the long-term memory are triggered to evaluate the savings in memory space and computational load. Figure 5.25 shows the number of nodes stored in the short-term memory per object, which gives an overview of the memory load for each search process (see also Table G.1 of Appendix G). The experiment is con-

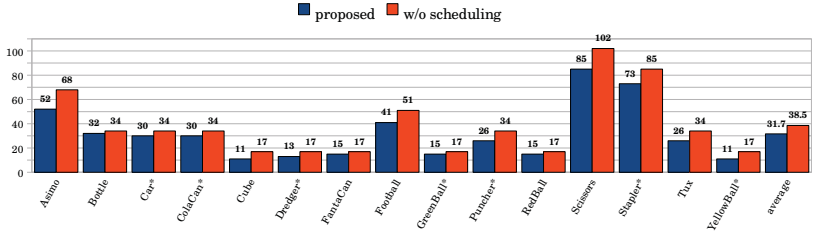


**Figure 5.25.:** This shows the number of nodes stored in the short-term memory with the attention control mechanism and without. On average, the proposed system (blue) stores about 18% less nodes compared to a system that does not schedule its visual actions but measures all properties at each fixation (orange).

ducted for a system with and without the attention control mechanism proposed in this thesis. The system is identical in all other respects (preprocessing, visual routines, etc). Objects marked with a * are not identified correctly and the search process is stopped, once such an object is fixated by the system. A detailed discussion on this can be found at the end of this section. However, as Fig. 5.25 shows, the number of

stored nodes and thus the memory load is reduced for all objects when employing the attention control mechanism proposed in this thesis. On average 18% less nodes need to be stored per object compared to the full measurement of all properties. The saving of memory capacity here ranges from 5.9% for the bottle up to 35.3% for the cube.

To get a better understanding were these savings come from, the number of measured properties are recorded for the system, which reflects the computational load of the system during the search process. The

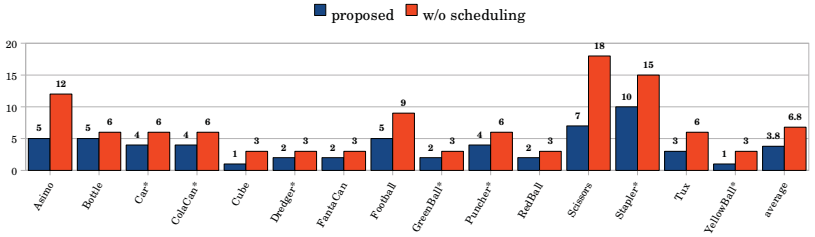


**Figure 5.26.:** This shows the number of properties (color, shape, physical size) measured for each search process. On average, the proposed system (blue) measures 44% less properties than a system that measures all properties per fixation (orange).

results of this experiment are shown in Fig. 5.26 (see also Table G.1 of Appendix G). The comparison shows that the number of measurements is reduced by about 44% when using the attention control mechanism proposed in this thesis. Please note that all state-of-the-art systems lack such a mechanism. To ease the comparison across different objects, Fig. 5.27 shows the number of saved measurements normalized by the number of fixations (see also Table G.1 of Appendix G). In nearly every fixation, the system is able to save at least one measurement to decide if an object candidate should be rejected or accepted. Even for the worst case in this scene (Bottle) the saving is 0.5 measurements per fixation. In general, the worst case scenario for the system would be no saving. For objects with discriminative properties like the

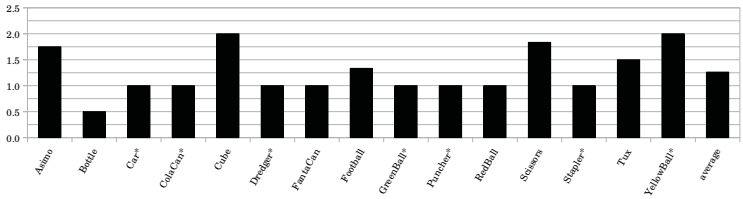


**Figure 5.27.:** This shows the number of measurements of the proposed system saved per fixation compared to a system that measures all properties. In most cases at least one measurement per fixation can be saved, on average the saving is 42%.

cube or yellow ball (color), the Asimo (shape) or the Scissors (shape) the number of saved operations is much higher.

Figure 5.28 shows a comparison of different system configurations. Here, the number of updated nodes is plotted, corresponding with the number of visual actions the system performs. On average, systems

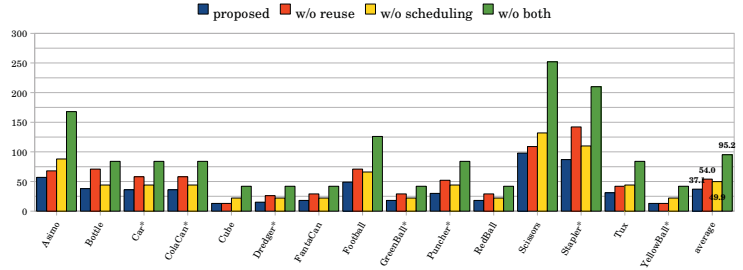


**Figure 5.28.:** This shows the number of visual actions taken during the search for known objects in different system configurations. On average, the proposed system (blue) requires 31% less actions than systems not reusing previous measurements but schedule visual actions (orange), 25% less actions than systems not scheduling visual actions but reusing previous measurements (yellow) and 61% less actions than systems neither reusing measurements nor selecting visual actions (green). Latter correspond to current state-of-the-art vision systems.

that do not reuse previous measurements and systems that do not selectively perform their property measurements perform equally well. However, both configurations require about 31% and 25% more visual actions respectively. When running the system without reusing previous measurements and without selectively processing object candidates, as state-of-the-art systems do, even 61% more visual actions need to be performed while keeping the system performance.

As mentioned earlier, objects marked with a * are not recognized correctly. Even though it is not the focus of this work, Table 5.3 shows the recognition performance for the system configuration used throughout the experiments. It can be easily seen that the system has a very

| | absolute | relative |
|---|---|---|
| Number of objects | 15 | 100.0% |
| True positives | 8 | 53.3% |
| False negatives | 7 | 46.7% |
| False positives | 3 | 20.0% |

**Table 5.3.:** This table shows the performance of the current proof-of-concept system in the given scene. Runs that result in false positive fixations were forced to continue the search.

high false negative rate, even for this low number of objects. Figure 5.29 shows the segmentation results which cause this misbehavior. While analyzing the measurement process, it has been found that in all cases the physical size measurement leads to the rejection of the object candidates. The physical size is calculated as $p_{physicalsize} \propto p_{size} \cdot p_z^2$ according to Eq. 3.18, where the pixel size $p_{size}$ depends on the segmentation of the object. As Fig. 5.29 shows, all objects of the false negative category suffer from either an undersegmentation or oversegmentation. Approaches to improve the initialization of the segmentation and the segmentation itself are described in chapter 6.

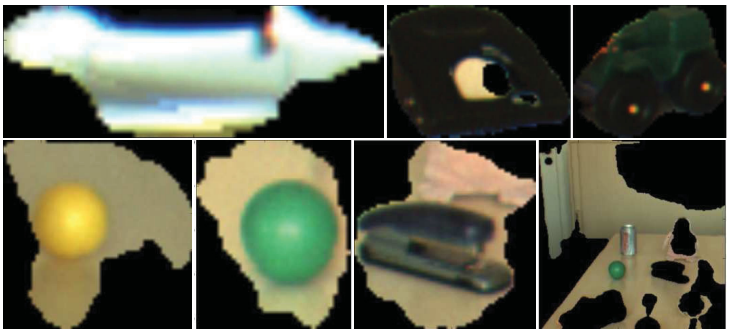

**Figure 5.29.:** Segmentation of the false negative objects shown in Table 5.3. An oversegmentation can be observed for objects in the upper row, while objects in the lower row tend to be undersegmented.

## 5.5. Discussion

In this chapter, a proof-of-concept implementation of a system was evaluated that employed algorithms proposed in this thesis. Visual search tasks of different complexity ("Find a red object" versus "Find the Asimo") were given to the system. The experiments show that the system reacts to this different degree in complexity by only measuring the properties that are necessary to solve the given task. For the red object it only measured the color, while for the Asimo the most discriminative property (the shape) was measured. Extensive tests for all objects show that the system is able to determine the property of an object discriminating the searched object best from all other objects by using the structure stored in the long-term memory. This reduces the number of nodes stored in the short-term memory. In fact, the complexity of the short-term memory corresponds to the task's level of difficulty in the given scene.

Furthermore, it has been shown that both the reuse of already acquired knowledge and the selective acquisition of information reduce the computational and memory load of the system. On average, about 18% less nodes are stored in the short-term memory, about 44% less properties are measured and about 61% less visual actions are taken compared to current state-of-the-art system configurations.

Finally, the recognition rate of the current system configuration was determined, even though this was not the goal of this thesis. It became clear that the high false negative rate (about 47%) mainly originates from the bad segmentation results for some objects. Here, a better initialization of the level-set segmentation is likely to boost the performance of the overall system. Approaches to improve this initialization can be found in the next chapter along with ideas to also improve other system components.

# 6. Future Work

The results presented in the previous chapter show that the system developed during this thesis is able to modulate its visual processing according to a given task. This modulation comprises both spatial and feature attention, leading to a more efficient usage of the computing resources and memory capacity. However, these results can only be a first step towards a task and knowledge guided scene representation. Quite a number of questions and possible algorithmic improvements still remain. These possible enhancements include the improvement of the segmentation, the saliency computation, algorithms for calculating additional object properties, extensions to the attention control algorithm and the handling of dynamic scenes. Those points will be briefly discussed throughout this chapter.



## 6.1. Visual Routines and Preprocessing

In this thesis only very basic algorithms are used to extract visual properties of objects. This was done to concentrate on the system architecture and the attention control mechanism itself rather than on the complexity of the underlying visual routines. However, a lot of potential improvements could be made enabling the system to cope with more complex visual scenes.

## Preprocessing

In the preprocessing domain the control of the input sensors plays an important role, as higher processing stages might suffer from information or precision lost in this stage. Here, especially the control parameters of the camera like the focus, exposure and gain could be controlled depending on the focused region to improve the contrast in the input image. This improved input image then eases the work of following routines, especially the stereo processing. Possible algorithms were already proposed for active vision systems [Bajcsy 1988].

A second important point here is the stabilization of the colors in the input independent from the systematic errors produced by the illumination of the input image. Especially the measurement of the color property would benefit a lot, as it currently matches the measured color in the image directly with the memorized one. Here, the algorithm of [Pomierski and Gross 1996] and later work like [Morel et al. 2009] might be suited to improve the situation.

## Saliency Computation

The saliency computation of the system presented in this thesis only uses very basic feature modulation mechanisms based on top-down information. Especially in this domain a lot of possible improvements can be found in literature. The work of [Frintrop et al. 2005] and [Michalke et al. 2008] for example propose the learning of an optimal set of feature weights to maximize the signal to noise ratio in the saliency map. They show that the search speed for certain objects is greatly improved. Additionally to the learning of the feature weights, the direct modulation of the features is proposed in [Navalpakkam and Itti 2007]. A similar, yet rudimentary, approach was used in the system presented here, where the color feature was modulated using a top-down expected color in the image. However, in the work of [Navalpakkam and Itti 2007] a more

elaborated schema incorporating multiple colors is used to also improve the signal to noise ratio in the saliency map. Despite the improvements the mentioned algorithms promise, all of them use background information to learn the weight and feature sets. Experiments using those algorithms in a more diverse setting have to show if an incorporation of the background information also holds there.

Additional to the modulation in the feature domain, the system proposed in this thesis has the ability to modulate the saliency computing spatially. Here, only the inhibition of return is used in the current implementation, because further information is not available to the system. However, in [Gross et al. 1995] and [Heinke 1997], an advanced learning schema is proposed, providing the means to learn and store spatial relations between objects and parts of an object. Given this ability, the system could speed-up its search process using all acquired knowledge to bias the saliency map. That is, even though the currently focused object is not the searched one, it could give a hint where the searched object can be expected. For example if we search for a computer mouse and currently focus on the computer's keyboard, the system can find the mouse to the right of the currently focused keyboard with a high probability.

## Segmentation

In the last experiment of chapter 5, the measurement of the object fails due to an over- and undersegmentation of the searched object. Indeed, the segmentation plays a crucial role in the system, as a bad segmentation leads to wrong measurements of object properties. In the examples mentioned, the size of the object was not estimated correctly, leading to a rejection of the correct object candidate. Analyzing the problem further shows that the initial conditions for the used Level-Set segmentation included too much of the table. In fact, this is always the case when dealing with elongated or thin objects, because then

the initialization with a circle is inappropriate as Fig. 6.1 shows. One possibility to improve the segmentation, besides the addition of further feature channels, is a better initialization of the algorithm. Here, an

**Figure 6.1.:** The Level-Set segmentation is initialized with a too small circle on the left hand side, whereas the initial circle is too large on the right hand side.

object can be presegmented within the initial circle by applying e.g. a k-means clustering on the region and selecting the cluster of pixels with the color most similar to the one stored in memory. That way, the region is likely to contain more inner pixels of an elongated object and reduces the amount of surrounding pixels.

Furthermore, additional feature channels can provide information on the background. One candidate for such a channel is the depth. By using the plane estimation algorithm proposed in [Einecke et al. 2008] in the present scenario a presegmentation as shown in Fig. 6.2 can be achieved. Using this feature channel in combination with the initialization described above would allow for an improved segmentation result.

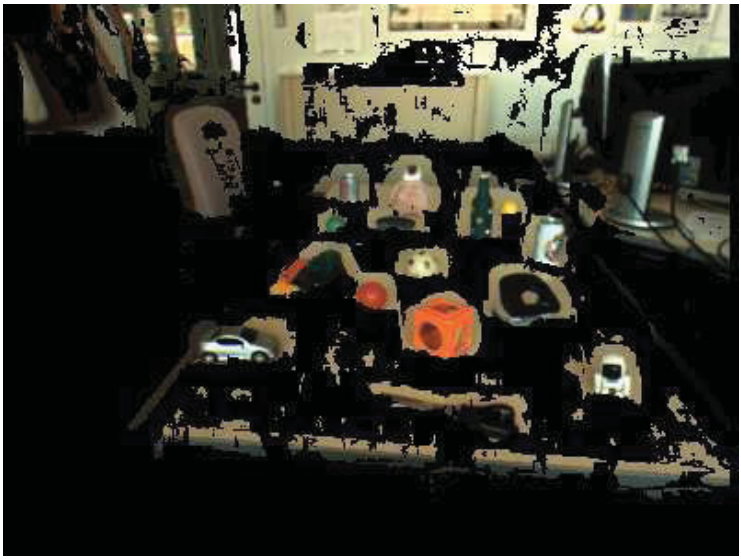

**Figure 6.2.:** A coarse segmentation of the scene is possible by incorporating information about the 3D position of the table in the given scenario.

## Additional Visual Features

When coping with real world scenes, the ability of the system to distinguish between objects is very limited by the used visual routines. This is due to the fact that an object currently is represented using a single color, the depth-normalized area and the aspect ration of the bounding box. While this information is enough to distinguish between objects in the conducted experiments, it probably will not work in more complex scenes. Here, at least a multi-color representation should be used. Additional to the color, the shape representation has to be improved. Possible approaches are proposed in [McNeill and Vijayakumar 2005] and [Tsai et al. 2005]. Experiments using these algorithms have to be

conducted in the future. Another potential discriminative feature is the texture or material of an object. In this domain, the work of [Do and Vetterli 2002] has to be investigated further. Contrary to the mentioned features, the work presented in [Hasler et al. 2009] provides means to learn discriminative features of objects. In the context of the system presented here, this approach could be used to separate two or more objects that are similar in other feature channels through learning.

### Feature Representation

As mentioned earlier in this thesis, currently the object properties are approximated using a k-means clustering algorithm. This approximation does not provide information about the extension of a cluster in the feature space. However, this information might be necessary to distinguish between objects or to modulate the input features in a sensible way. Here, multiple alternatives like histograms, Gaussian Mixture Models [Bishop 2006a], Self-Organizing Feature Maps [Kohonen 1990], Growing Neural Gas [Firtzke 1995] and Radial Basis Functions [Powell 1985, Broomhead and Lowe 1988] exist. The extensive investigation of those algorithms has to show if an improvement compared to the fast and simple k-means algorithm can be achieved. This investigation has to take into account both the discriminative power of the representation and the possibility of feeding the information back to the system.

## 6.2. Memory and Attention Control

For the memory and attention control part of the system, a lot of improvements can be thought of. Currently, the system's procedural as well as its semantic memory is not learned but rather constructed by hand. To let the system learn this information about how its sensory pathways work and how the world around the system is structured is

also subject to future work. Improvements in this part of the system comprise, among others, the topics categorization and clustering, estimation of costs and information gain for visual actions, the way the activation is spread in the memory and the generation of object hypotheses from visual stimuli. These topics are discussed in more detail in this section.

## Alternative Processing Pathways

In the dependency graph representing the algorithmic interaction of visual routines, alternative pathways are possible as shown in Fig. 6.3. Equivalent to the example in Fig. 6.3 for alternative segmentation al-

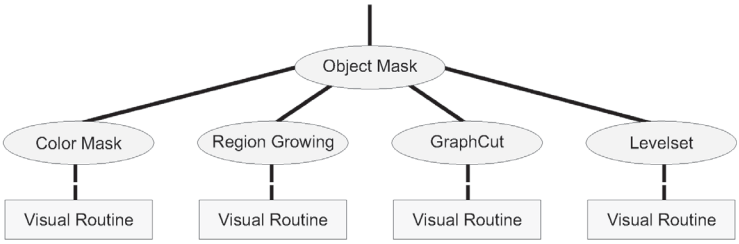


**Figure 6.3.:** The illustration shows the possibilities of alternative pathways using the example of segmentation. Dependent on the required precision of the segmentation mask, different algorithms could be used, implying differences in computing time and demand.

gorithms, alternative processing pathways exist for depth, shape or size. Here, a criterion needs to be established that allows the system to choose one of the alternatives dependent on the constraints imposed by the current computational load, the available processing time and the required precision for the calculation. Once such a criterion is found, the system will be able to select the visual action out of the alternatives that fits best the required precision and time constraints

imposed by the given task. This would be another major step towards a task-dependent processing of visual information.

Possible candidates for such a criterion are the required computing time and the achieved precision of the algorithm, which should be estimated by the system during a learning phase. As suggested earlier in this thesis, the computation time could be interpreted as a cost for the system to measure a property, whereas the acquired information (including its precision and confidence) would serve as a gain or benefit. Interpreting the estimated values this way points out a clear relation between the problem of selecting a processing pathway and the decision making problem handled in game theory. In both fields, the balancing of the risk or cost and the benefit is required. Different approaches proposed in the game theory and decision making like the Minmax theorem [von Neumann 1928] should be evaluated (see [Nisan et al. 2007] for an introduction). Additional to the internal constraints imposed by the system itself, external time requirements could be given for the task at hand.

## Learning of Object Representations

The learning and adaptation of internal representations are important features for a vision system. During this thesis, the learning of new object representations was largely ignored for simplification reasons. However, the learning of new object and property representations is subject to future and ongoing work. Here, two important learning signals can be used. First a novel object is found if the system can not identify the currently focused object. The novelty is expressed by the fact that no object remains activated in the long-term memory after a series of measurements. Second, an object can not be distinguished definitively, even though the whole palette of visual actions was applied. In this case, multiple objects will remain activated in memory after the measurements. In the first case, a new object representation can be generated possibly using the already existing property nodes

and thus sensory clusters. If no suitable sensory cluster exists, a new one needs to be added. The second case is more difficult as here the existing representation is not fine enough to definitively assign the object in question to a feature combination. In this case, the existing sensory representation needs to be refined by either modifying the sensory clusters or by using more precise visual routines. The addition and refinement of sensory clusters, while keeping the already acquired knowledge and represented objects, is strongly related to problems researched in categorization and classification scenarios. Here, algorithms like [Kohonen 1989] or [Kirstein et al. 2008] need to be investigated and evaluated. The possibility of false or ambiguous information in the long-term memory needs to be explicitly considered.

Additionally to learning new objects in the long-term memory, the forgetting of objects in the short-term memory is essential for a continuously running system. Different strategies are possible that incorporate the durability of information, the continuous evaluation of objects with respect to the relevance for the current task and the predictability of the objects in the current scene. That is, if an object is e.g. known to be static, like a table, a fridge or a wall, the position of the objects does not need to be stored in the short-term memory, but can always be retrieved from the long-term memory. The same holds if the properties of an object match the ones stored in the long-term memory and can be predicted over time using existing prediction models. Strategies to decide when properties or whole objects should be erased from the short-term memory depending on the current task need to be investigated here.

## Representation of Object Relations

In the system as proposed in this thesis, the knowledge about relations between objects is currently not used to improve the search speed and robustness. These relations split into the co-occurrence of objects and

into a stable spatial relation between objects. Furthermore, a related behavior between objects needs to be considered for dynamic scenes. Here, multiple approaches exist to learn both spatial relations and the co-occurrence of objects. The work presented in [Gross et al. 1995] and [Heinke 1997] shows that the scan path within a given scene can be dramatically improved by incorporating knowledge about spatial and feature relation between object parts. With a high probability, the algorithm proposed there can also be used to learn relations between different objects rather than parts of the same object. Additionally, approaches using statistical evaluations like the one presented in [Riad et al. 2009] show that a purely statistical learning of such relations is also possible. Alternatively, Bayesian methods could be used to learn the probability of the co-occurrence between two or more objects. Also here further investigations and experiments need to be conducted to evaluate the gain of applying these or similar methods with respect to the overall system performance.

## Activation Spread

In the current implementation, the activation of an object is calculated as the product of its measured properties. A weighting of those properties or the influence of other objects measured in the current scene are completely ignored. However, this rather simple activation calculation mechanism is quite similar to the likelihood calculations done in the naive Bayesian classifier [Zhang 2004, Langseth and Nielsen 2006]. The similarity, together with the simplicity and good performance of the naive classifier, are encouraging to investigate a possible rebase of the activation spread on this framework. By doing so, additional influences on an object's activation can be modeled based on a well-researched probabilistic framework. Beside the similarity to the naive classifier framework, the problem of spreading activations in graph-like structures is also researched in the AI community. Here, different

methods like [Anderson 1983], [Pearl 1982] and [Bishop 2006b] exist (see [Crestani 1997] for an overview). The investigation of these and other methods is also subject to future work.

## Fast-Forward Hypothesis

During this thesis, the system was always provided with a certain specific search task steadily inducing object hypotheses. However, for the normal operation of a visual system, this might not always be the case. Here, the hypotheses of the system for a currently focused object might be created by a certain property, e.g. color, in a fast-forward manner. Even though this case looks similar to the specific task case, there is a fundamental difference. In the case handled in this thesis, only a single hypothesis for the searched object is induced by the given task, whereas in the fast-forward case, multiple hypotheses have to be considered. The reduction of this set of hypotheses is then the goal of the system. A possible algorithm to handle this could look like this:

**Procedure *IdentifyObject*:**

**(a) Initialization**

(1) Reset all activations in memory to zero.

**(b) Spread activation**

(1) Use the measured value $\mathbf{v}$ and calculate the activation $a(p_t \in P_t)$ for all properties $P_t = \{p \in P | \exists (p, p_{type}) \in E_{inheritsFrom}\}$ in memory that are of the same type $p_{type}$ as the measurement. The activation is calculated using Eq. 4.12.

(2) Propagate the activation of nodes $a(p_t)$ to the objects $a(o)_t = \sum a(p_i) \forall p_i \in p_t$ and determine the hypothetical object candidates $O_h = \{o \in O | o : a(o)_t \geq \xi\}$.

(3) If only one candidate remains $\|O_h\| \stackrel{?}{=} 1$, the object is identified. The system can stop here.

**(c) Schedule visual routines**

(1) Calculate the estimated number of remaining object hypotheses for a property $p_i$, given that the visual routine measures this property $d_i = \|\{o \in O_h | \exists(o, p_i) \in E_{hasProperty}\}\|, \ \forall p_i \in P$.

(2) Calculate the "risk" $r_{type}$ of triggering a visual routine $p_{type}$ by determining the "uniformity" of the distribution over the number of remaining hypotheses for the given routine $p_{type}$. The rationale here is that equally distributed remaining hypotheses are suited best to eliminate as many hypotheses as possible with a single measurement of a visual routine. This is the case, as contrary to peaked distributions the risk of eliminating only a few hypotheses is small. One possible measure for the uniformity is the variance in the remaining hypotheses $r_{type} = E((d_i - E(d_i))^2), \ \forall p_i \in P_r$ with $P_r = \{p \in P | \exists(p, p_{type}) \in E_{inheritsFrom}\}$.

(3) Choose the property and thus visual routine having the smallest risk $p_{type} : \text{argmin} r_{type}, \ \forall r_{type}$ and measure it.

(4) Continue with step (b).

Using this method, the system itself could generate hypotheses in cases where no specific task is given to build-up a representation of its visual vicinity. Here, the level of detail of this representation is determined by the knowledge of the system and the ability to distinguish objects in the scene.

## 6.3. Dynamic Scenes

Up to now, the system is restricted to static scenes by its visual abilities. That is, the system has no means to process motion or other spatio-temporal changes. However, those changes happen quite often in the real world. In this section, possible approaches, algorithms and enhancements should be discussed that enable the system to also deal with dynamic scenes in the realm of object and ego-motion. Here, the changes required in the preprocessing, additional visual routines and changes in the memory and control structure are discussed.

## Preprocessing

When dealing with dynamic scenes, the extraction of motion in the scene at hand is crucial. The perceived motion in the input image can be either introduced by a movement of the system itself, the motion of objects in the scene or by a combination of both cases. This makes it necessary to estimate or measure both the movement of the system and the motion in the image. To do so, the system needs to be provided with an algorithm to compute the optical flow in the image and means to estimate its own movement. For calculating the optical flow in the image, multiple approaches like [Horn and Schunck 1981] or [Willert et al. 2006] exist (see [Beauchemin and Barron 1995] for a survey). The implementation and testing of one or more of those algorithms is part of future work.

## Visual Routines

Based on the flow-field and the ego-motion estimation discussed in the preprocessing section, visual routines have to be added to determine and track the motion of an object. Here, the detection of moving objects plays a major role. Basically, there exist two ways to integrate the motion into the saliency computation as illustrated in Fig. 6.4. The simplest way of integrating motion is to interpret the optical flow as an additional feature channel fed to the existing saliency computation. Even though this is a very simple modification, this interpretation has some down-sides. When feeding the motion channel to the existing framework, it is also subject to both spatial and feature weighting, which might lead to a suppression of locations containing motion. This is an undesirable behavior for the system, because especially the moving parts of the scene need to attract the attention of the system quickly to synchronize the system's representation with the real visual scene. Therefore, the second method, namely the addition of an independent

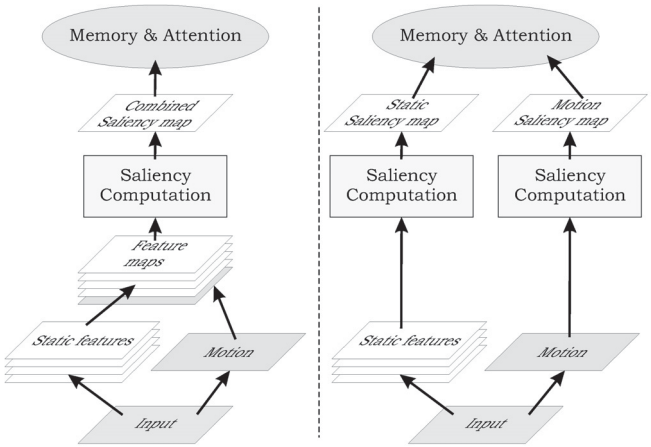


**Figure 6.4.:** Two possibilities of integrating the motion channel into the system's saliency. First, the optical flow-field can be integrated as yet another feature channel as shown on the left, or an additional motion saliency can be computed as shown on the right.

motion saliency, is preferred. While doing so, the existing saliency computation algorithm can also be used for the motion saliency. Additional to the object moving in the scene, the ego-motion of the system also introduces an optical flow. The compensation of this ego-motion as suggested in [Schmüdderich et al. 2008] reduces this effect and allows the system to concentrate on local motion of moving objects rather then the global motion generated by itself. After calculating the motion saliency map, the two saliency maps need to be balanced. That is, a decision needs to be made when a switching between the maps is required. Here, a task-dependent threshold on the motion saliency map is suggested. Whenever this threshold is exceeded in a location of the motion saliency map, the system switches its attention to this location in the current scene. The variation of the threshold allows the system

to concentrate more on its current task (this information is contained in the static saliency map) or to concentrate more on the maintenance of the current scene (which requires to concentrate on changes in the scene).

Beside the motion saliency computation, the tracking of objects and the identification of object trajectories are interesting. The tracking of objects using e.g. [Lucas and Kanade 1981, Ristic et al. 2004] or [Zhang et al. 2009], is necessary to be able to update, maintain and synchronize the system's internal representation with the real visual scene. Here, especially the continuous integration of information over multiple time frames for one object allows a more robust representation of the tracked object. The identification of trajectories might provide an additional discriminative feature to distinguish between objects. The review of the literature concerning trajectory identification and the implementation and investigation of such algorithms is also part of future work.

## Spatio-temporal Prediction

The pure detection and measurement of motion is not sufficient to keep the system's internal representation synchronous to the outer visual world. The prediction of this internal representation in time and space must be an integral part of the system's architecture when dealing with dynamic scenes. Here, the system can also benefit from using its knowledge about typical object movements by using possibly learned prediction models for the different objects. One example for such a prediction model is a static or identity prediction for highly immobile objects like walls, windows or furniture. Furthermore, the joint movement of objects could be exploited. When for example an object is moved, the system could combine the measurements of the person and the object into a joint prediction model as e.g. the work presented in [Prankl et al. 2009] suggests. It is important to predict the whole internal representation and afterwards select items that require

an update of the location or properties. The update then is done in a prediction-confirmation loop using the known and predicted properties of the objects represented internally. The implementation and selection of different prediction models are also part of future work.

## 6.4. Discussion

Many different enhancements and planned future investigations were presented in this chapter. All of these proposals benefit from the flexible system architecture proposed in this thesis and most of them could be implemented in a straight-forward way. This shows that the architecture of the system and the memory as well as the concept of the control mechanism can serve as a solid foundation for the investigation of further questions regarding a task and demand-driven scene representation. In the next chapter, this thesis will be briefly summarized.

# 7. Summary

The goal of this thesis is to build-up a scene representation in a task-driven way utilizing the knowledge of the system about the world and the current scene. The idea is proposed to only acquire those information about the current visual scene that is needed to solve the given task. After reviewing the literature for related work in chapter 2, it becomes clear that current state-of-the-art systems do not consider this kind of selective processing in the feature space. Instead, all these systems rather focus on the selective processing of locations in the image. That is, once those systems fixate a certain object candidate, they measure all information about this candidate regardless of the actual need for the given task.

To selectively process information in both the feature and spatial domain as described before, flexible processing pathways are required. Therefore, fixed saliency-segmentation-classification processing pipelines found in most systems have to be split up into functional parts that can be flexibly combined afterwards. This is because the selective acquisition of information requires the selective triggering of visual routines. In turn this also requires a dynamical and flexible system architecture that allows such a demand-driven combination of processing modules. Such a system architecture is proposed in chapter 3 of this thesis along with a simple spatial attention mechanism and simple visual routines for measuring object properties. Here, each visual routine is highly specialized to measure only one property of an object. To acquire more complex information, those routines need to be combined in a suitable way.

The combination of visual routines requires the knowledge on which routines have to be executed in order to extract certain information from the scene (segmentation before color measurement, etc). For doing so, the storage of procedural knowledge in the long-term memory is proposed in chapter 4. This procedural knowledge is represented consistently as a graph along with the knowledge of the system about the world. An algorithm is presented that allows to parse the graph containing the procedural knowledge, enabling the system to combine visual routines in the correct temporal order to process information in a sensible way. Furthermore, the memory architecture presented in chapter 4 stores a link between sensory nodes of the memory graph and the corresponding sensory representations. This anchors the nodes of the long-term memory in the sensor space, allowing for an easy biasing of visual routines in these spaces. Furthermore, this is a major difference to classical AI approaches that only work on symbolic information.

Beside the ability to order the visual routines temporally, the system needs to decide which properties it has to measure for solving the current task. An attention control mechanism is presented in chapter 4 that bases its decision on the long-term knowledge of the system. Here, the minimal set of measurements is selected that is needed to solve the given task. Experiments conducted in chapter 5 involving visual search tasks of different complexity show that the application of the proposed algorithms leads to a reduction of both the computational and memory load compared to state-of-the art algorithms. The resulting representation is task-related, as only the information to solve the task is acquired. Furthermore, the complexity of the given task in a certain scene determines the number of measurements and thus the computational load. However, the experiments also show that the system suffers from a high false negative rate while evaluating object candidates. The reason for this are over- and undersegmentations of objects due to a suboptimal initialization of the segmentation algorithm. Possible solutions to these problems, as well as improvements for other system components and the extension to dynamic visual scenes, are proposed in chapter 6.

To summarize, the main contributions of this thesis are the flexible system architecture, the graph based memory architecture with anchored sensory nodes and the attention control mechanism. Those contributions allow to build a system that flexibly combines its processing pathways in a task- and demand-driven way. The experiments conducted in this thesis show that the selective processing of visual information beyond the spatial domain results in an immense reduction of both computational and memory load while solving the given task. That way, a task-driven scene representation can be realized serving as a foundation for learning new objects and interpretations of the current scene.

# A. Amari Field Dynamics

In [Amari 1977], Amari presents the framework for an inhibitory lateral coupling within a homogeneous neural field. This kind of neural field dynamics is used as a lateral coupling mechanism while relaxing the contrast maps in the course of the saliency map calculation. For a single layer field, as used in this thesis, the membrane potential $\mathbf{u}$ changes depending on the decay constant $\tau$, the lateral coupling kernel $\mathbf{w}$, the driving input $\mathbf{s}$ and the global resting potential $h$ according to

$$\tau\dot{\mathbf{u}} = -\mathbf{u} + \mathbf{w} * f(\mathbf{u}) + h + \mathbf{s} \ . \tag{A.1}$$

Here, $f(\cdot)$ is a nonlinear function, e.g. sigmoidal, applied to each element of the neural field. To implement the field dynamics described above, an iterative numerical solution to the differential equation is required. For doing so, the difference quotient is used in this thesis, which reads as

$$\tau\frac{\mathbf{u}_{t+\Delta t} - \mathbf{u}_t}{\Delta t} = -\mathbf{u}_t + \mathbf{w} * f(\mathbf{u}_t) + h + \mathbf{s}_t \ . \tag{A.2}$$

This assumes that the time difference $\Delta t$ is very small compared to the decay constant $\Delta t \ll \tau$. By rearranging the equation above, one gets the following approximated solution for the differential equation

$$\mathbf{u}_{t+\Delta t} = \mathbf{u}_t + \frac{\Delta t}{\tau}\left(-\mathbf{u}_t + \mathbf{w} * f(\mathbf{u}_t) + h + \mathbf{s}_t\right) \ . \tag{A.3}$$

An investigation on equilibrium point, stable solutions, etc can be found in [Amari 1977]. The values used for the parameters $\Delta t$, $\tau$, $h$ and the lateral kernel $w$ can be found in Appendix E.

# B. Retinal Size Estimation

In some stages of the system presented in this thesis a coarse retinal size estimation of an object is required, e.g. for the inhibition of objects or as a starting condition for a later segmentation. Currently, this size estimation is computed in the saliency module on the basis of the Gaussian resolution pyramid. Figure B.1 illustrates that the Gaussian resolution pyramid used in the saliency computation is equivalent to the filtering of the input image with a set of Gaussian filters. The

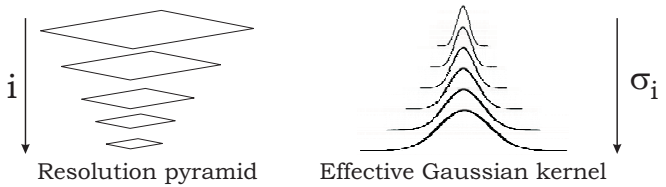


**Figure B.1.:** The application of a Gaussian resolution pyramid is equivalent to the filtering of the input image with a set of Gaussian filters.

actual size of a Gaussian filter at the $i$-th level of the pyramid can be expressed with its standard deviation $\sigma_i$ as

$$\sigma_i = 2^i \cdot \sigma_0 \ , \tag{B.1}$$

with $\sigma_0$ being the standard deviation of the Gaussian kernel used for down-sampling in the pyramid. When applying these Gaussian filters on different sizes of an (ideal) object the filter responses shown in Fig. B.2 can be observed for a single pixel. The size here is con-

sidered to be represented by a circle with an approximated radius. It

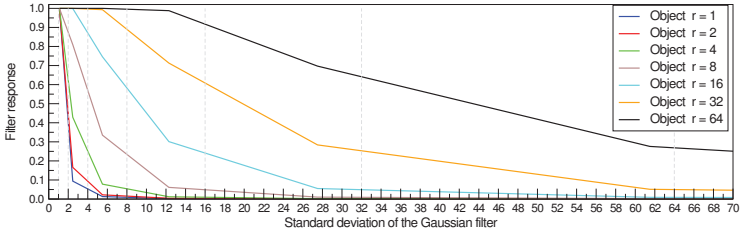


**Figure B.2.:** This Figure shows the Gaussian filter responses for different sizes of a star object at a single pixel. The drop in the filter response correlates with the size of the object.

can be seen that the drop of the filter response correlates with the size of the filtered object. By detecting that point the size of the object can be estimated. However, the filter responses are not that docile for real-world objects as Fig. B.3 shows. As can be seen, for real-world

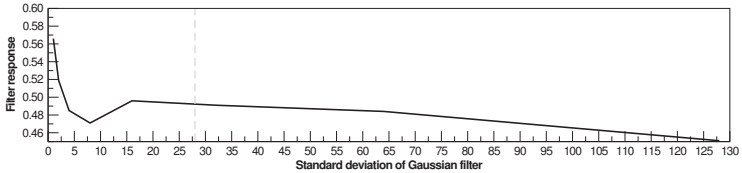


**Figure B.3.:** This Figure shows the Gaussian filter responses for a real-world object. The used object has a radius of approximately 28 pixel.

images the size can not be estimated by detecting the drop of the filter response. Therefore, the radius of the object $r(x, y)$ at the position $(x, y)$ is averaged over the different sizes. At this the size $2\pi\sigma_i^2$ at each

level $i = 0, .., N$ is weighted by the filter responses $f_i$

$$2\pi r(x,y)^2 \propto \frac{\sum_i f_i(x,y) \cdot 2\pi\sigma_i^2}{\sum_i f_i(x,y)} \tag{B.2}$$

$$r(x,y)^2 \propto \frac{2\pi \sum_i f_i(x,y)\sigma_i^2}{2\pi \sum_i f_i(x,y)} \tag{B.3}$$

$$r(x,y) \propto \sqrt{\frac{\sum_i f_i(x,y)\sigma_i^2}{\sum_i f_i(x,y)}}\ , \tag{B.4}$$

with $\sigma_i = 2^i\sigma_0$ as defined in Eq. B.1.

# C. K-means Clustering



In this thesis, the k-means cluster algorithm as proposed in [Lloyd 1982] is used to extract for example an object's color or depth. Therefore, the previously defined $kmeans(\cdot)$ operator is described in more detail in this appendix. The basic steps of the k-means clustering can be seen in Fig. C.1. The goal of the k-means algorithm is to cluster the $n$ data

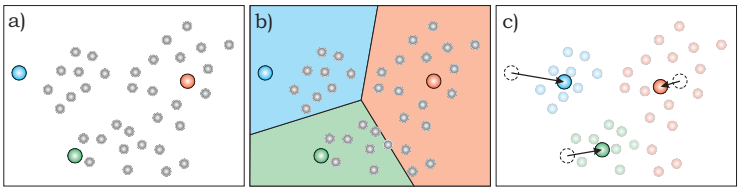


**Figure C.1.:** One iteration of the k-means algorithm on the data shown in a) involves the Voronoi tessellation of the data according to the cluster centers (red, green and blue) as shown in b). Afterwards, the cluster centers are adapted as shown in c) until convergence.

points $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ into $k$ clusters. To do so, the cluster centers $\mu_1(t=0), ..., \mu_k(t=0)$ are initialized randomly. In a first step, the data points $X$ are assigned to the clusters such that the sets $S_1(t), ..., S_k(t)$ contain the points that fulfill

$$S_i(t) = \{\mathbf{x} \in X : \| \mathbf{x} - \mu_i(t) \| < \| \mathbf{x} - \mu_j(t) \| \ \ \forall j \ i \neq j\} \ . \tag{C.1}$$

In a second step the cluster centers $\mu_1(t), ..., \mu_k(t)$ are updated so that

$$\mu_i(t+1) = \frac{1}{|S_i(t)|} \sum_{\mathbf{x} \in S_i(t)} \mathbf{x} \; . \tag{C.2}$$

These two steps are iterated until either the cluster centers converge $\mu_i(t+1) = \mu_i(t) \; \forall i$ or the maximal number of iterations are reached.

# D. Optimal Bounding Box Calculation



In chapter 3 the operator $boundingbox(\cdot)$ is defined, which computes the optimal bounding box containing a set of points. This bounding box is later used to calculate the rough shape of a focused object. The algorithm to compute the optimal bounding box, given a set of sample points $S = \{s_1, ..., s_I\}$, is proposed in [Toussaint 1983] and is described in this Appendix.

The calculation of the optimal bounding box as described in [Toussaint 1983] requires the ordered convex hull $P = \{p_1, ..., p_n\}$ of the point set $S$. Ordered in this context means that the points are sorted clockwise, starting with the bottom-most one. To calculate the convex hull $P$ the algorithm proposed in [Graham 1972] is used here. This algorithm has a time complexity of $O(n \log n)$. Figure D.1 shows the convex hull of the input points $S$. Starting from the clockwise ordered convex hull $P = \{p_1, ..., p_n\}$ two pairs of orthogonal calipers are selected in the way shown in Fig. D.2. These two orthogonal pairs of calipers with the support points $p_i, p_j, p_k, p_l$ are rotated about the minimal angle $\Phi = min\{\Phi_i, \Phi_j, \Phi_k, \Phi_l\}$ between their respective successive points $p_{i+1}, p_{j+1}, p_{k+1}, p_{l+1}$. For each of the rotations the area is calculated enclosed by the resulting rectangle. Finally, the minimal area with the respective set of calipers is computed, representing the minimal bounding box.

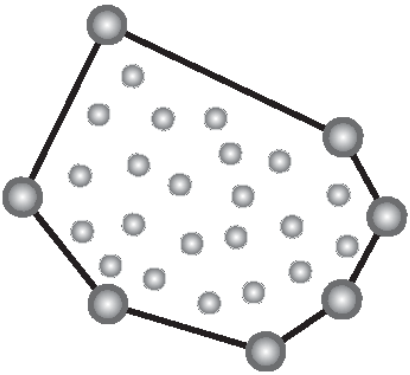

**Figure D.1.:** The large points represent the convex hull around the set of input points.

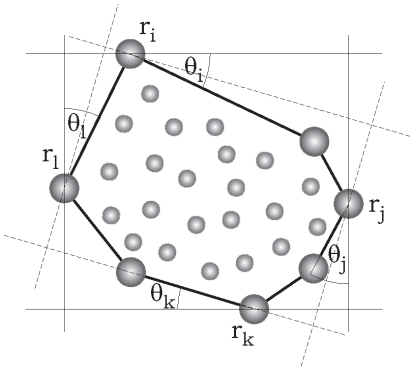


**Figure D.2.:** Two pairs of orthogonal calipers (solid lines) are chosen such that they are parallel to the $x$ and $y$ axis. Now, the calipers are rotated clockwise about the smallest angle ($\Phi_k$ in this example) between their successive support points. The resulting calipers are shown with dashed lines.

# E. System Parametrization

## Dependency Graph

Figure E.1 shows the expanded dependency graph used in the proof-of-concept system. This graph is stored in the sensory representation part of the long-term memory (see chapter 4 for details).

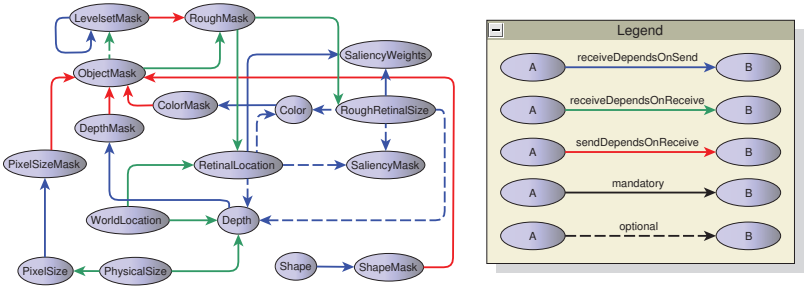


**Figure E.1.:** This figure shows the expanded dependency graph stored in the long-term memory. The colors of the arrows between the nodes represent the action interdependency between two nodes.

# Algorithm Parameters

| Parameter | Value |
|---|---|
| **Image parameters** | |
| size [pixel] | 800x600 |
| colorspace | RGB |
| value range | 0.0..1.0 |
| **Depth calculation parameters** | |
| Size correlation window | 17x17 |
| # disparities | 96 |
| **Saliency parameters** | |
| Decay constant $\tau$ | 1.0 |
| Iteration timestep $\Delta t$ | 1.0 |
| # iterations | 5 |
| Level in Gaussian resolution pyramid $I$ | 8 |
| Lateral difference of Gaussian coupling kernel $\mathbf{w}$ with $\sigma_+$ | 10 |
| $\sigma_-$ | 18 |
| Resting potential $h$ | 0.0 |
| **Feature extraction parameters** | |
| # color clusters | 3 |
| # depth clusters | 2 |
| # iterations | 10 |
| **Level-Set parameters** | |
| $\mu$ | 1.0 |
| $\nu$ | 0.5 |
| # bins for histogram | 12 |

**Table E.1.:** System parameters of the used algorithms for reference.

# F. Experimental Dataset

## Dataset


**Figure F.1.:** This figure shows the set of objects used to fill the system's long-term memory. The objects are hand-segmented. Blue represents the background.

# Color Clusters

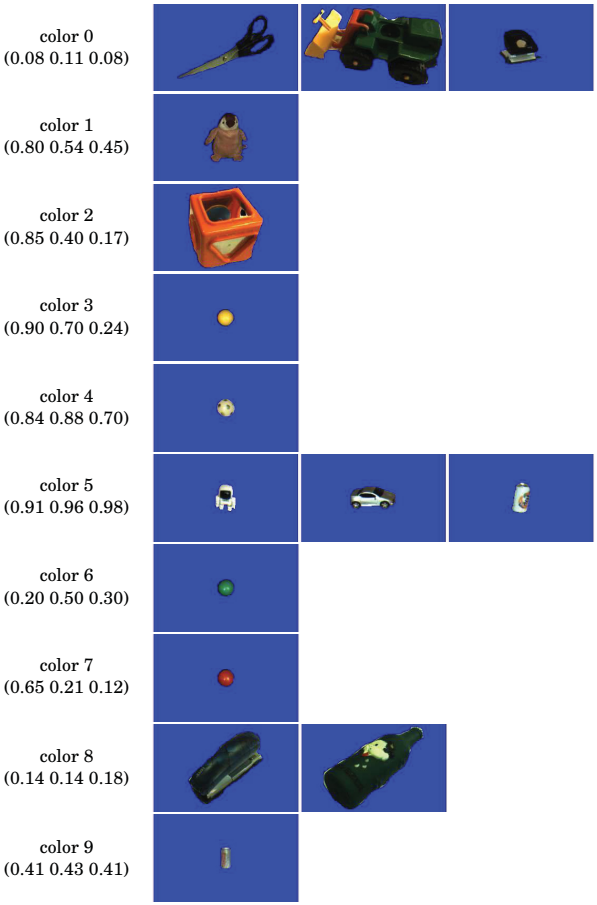


**Figure F.2.:** All clusters in the color space, their cluster centers and the assigned objects are shown.

## Physical Size Clusters

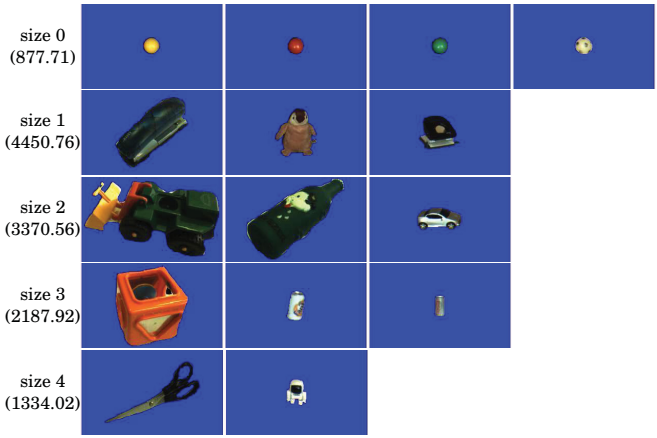


**Figure F.3.:** This figure shows all clusters in the physical size space, their cluster centers and the objects assigned to each cluster.

## Shape Clusters

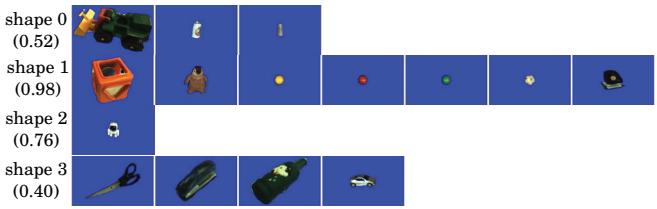


**Figure F.4.:** This figure shows all clusters in the shape space, their cluster centers and the objects assigned to each cluster.

# G. Experimental Result Images

## Biased Color Map and Saliency

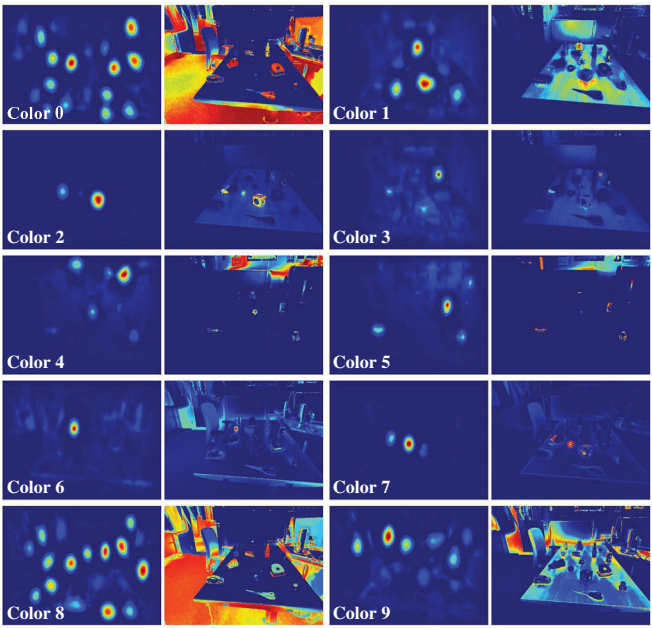


**Figure G.1.:** This figure shows the saliency map (left) and the corresponding biased color map (right) for all color clusters known to the system.

# Dependency Resolving

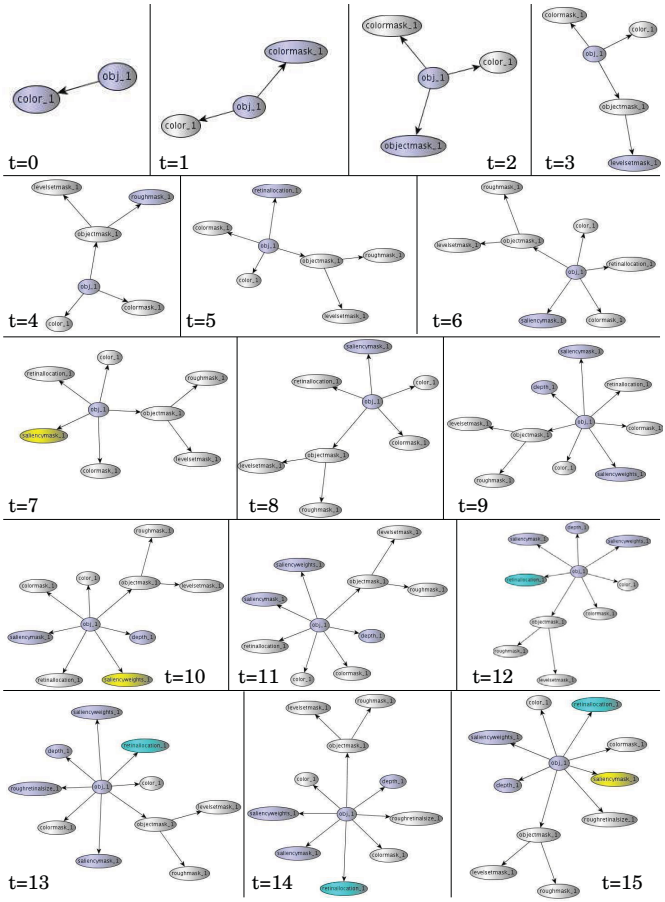


**Figure G.2.:** The dependency resolving process for a color measurement is shown.

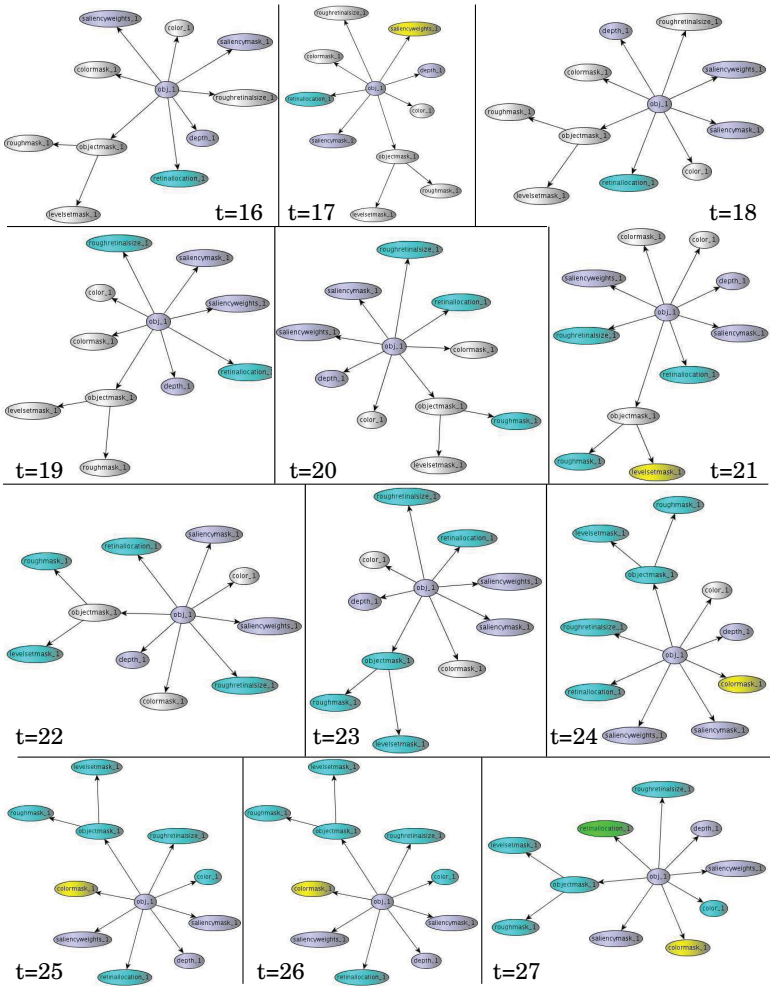


**Figure G.3.:** The dependency resolving process for a color measurement is shown (continued).

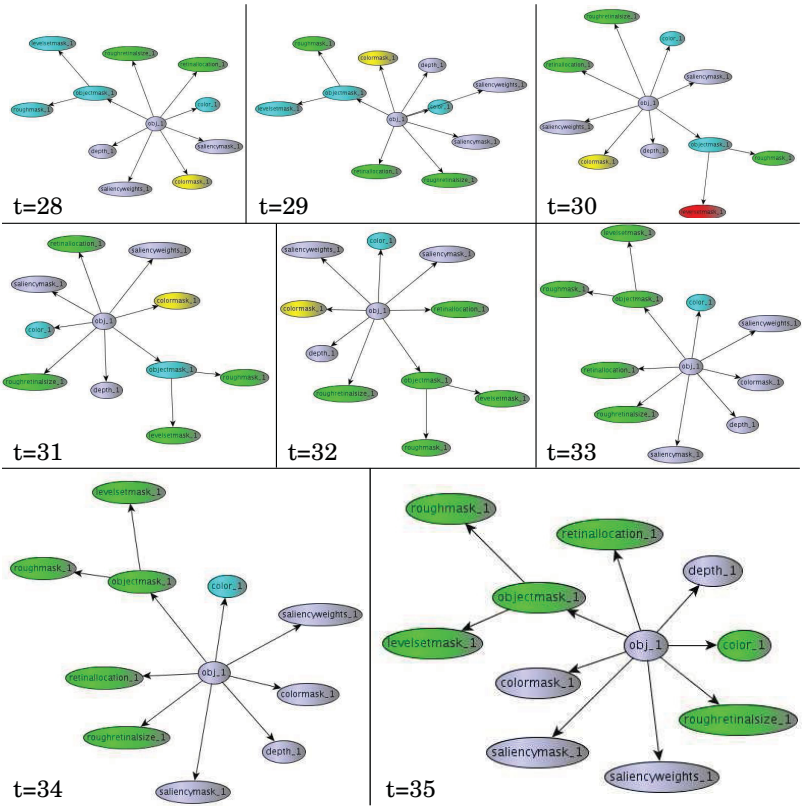


**Figure G.4.:** The dependency resolving process for a color measurement is shown (continued).

## Scan paths

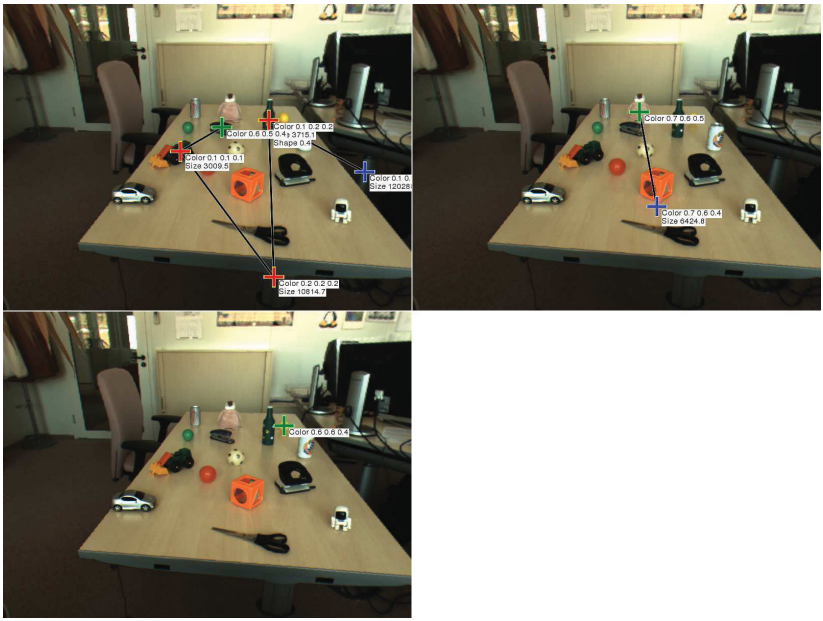


**Figure G.5.:** This figure shows the scan paths for all objects. At each fixation the measured properties are given. The blue cross marks the first and the green one the last fixation. Red crosses mark intermediate fixations. (Continued on next pages).

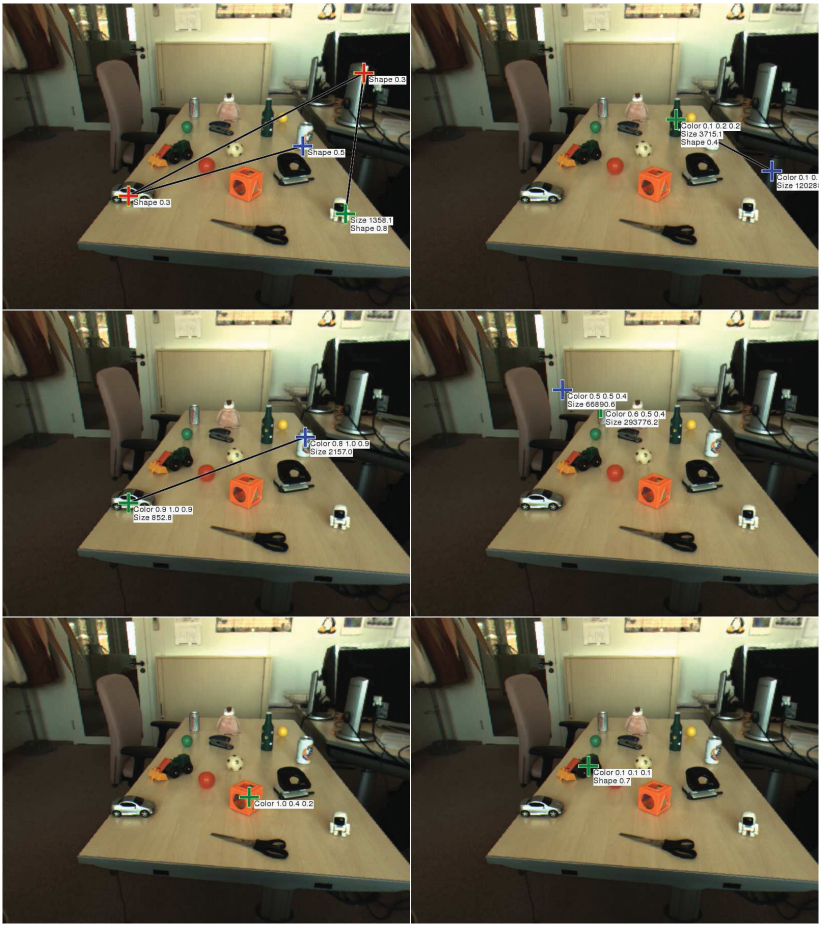


**Figure G.6.:** Scan paths continued. The blue cross marks the first and the green one the last fixation. Red crosses mark intermediate fixations.

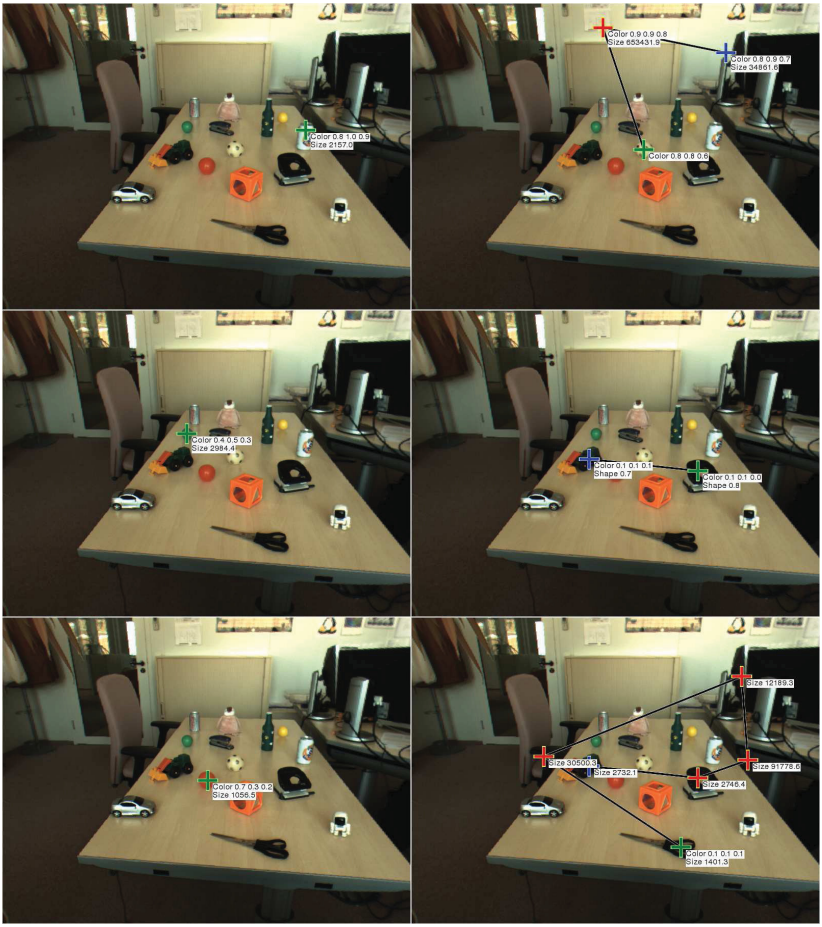


**Figure G.7.:** Scan paths continued. The blue cross marks the first and the green one the last fixation. Red crosses mark intermediate fixations.

## Measurements

| Object | # Fixations | # Properties | # Updates | # STM Nodes |
|---|---|---|---|---|
| Asimo | 4 | 5 | 57 | 52 |
| Bottle | 2 | 5 | 38 | 32 |
| Car | 2 | 4 | 36 | 30 |
| ColaCan | 2 | 4 | 36 | 30 |
| Cube | 1 | 1 | 13 | 11 |
| Dredger | 1 | 2 | 15 | 13 |
| FantaCan | 1 | 2 | 18 | 15 |
| Football | 3 | 5 | 49 | 41 |
| GreenBall | 1 | 2 | 18 | 15 |
| Puncher | 2 | 4 | 30 | 26 |
| RedBall | 1 | 2 | 18 | 15 |
| Scissors | 6 | 7 | 98 | 85 |
| Stapler | 5 | 10 | 87 | 73 |
| Tux | 2 | 3 | 31 | 26 |
| YellowBall | 1 | 1 | 13 | 11 |

**Table G.1.:** This table shows the performance of the current proof-of-concept system in the given scene.

# Bibliography


Aloimonos, Y. (1990). Purposive and qualitative active vision. In *Proceedings of the 10th International Conference on Pattern Recognition*, volume 1, pages 346–360.

Aloimonos, Y. (1993). *Active Vision Revisited*, chapter Introduction, pages 1–18. Lawrence Erlbaum Associates, Hillsdale.

Aloimonos, Y., Weiss, I., and Bandyopadhyay, A. (1988). Active vision. *International Journal of Computer Vision*, 1(4):333–356.

Amari, S. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 27:77–87.

Anderson, J. R. (1983). A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behaviour*, 22:261–295.

Aziz, M. Z. (2009). *Behavior Adaptive and Real-Time Model of Integrated Bottom-Up and Top-Down Visual Attention*. PhD thesis, University of Paderborn.

Aziz, M. Z. and Mertsching, B. (2008). Visual search in static and dynamic scenes using fine-grain top-down visual attention. In *Proceedings of the 6th International Conference on Computer Vision Systems*, volume 5008 of *LNCS*, pages 3–12.

Bajcsy, R. (1985). Active perception vs. passive perception. In *Proceedings of the IEEE Workshop on Computer Vision: Representation and Control*, pages 55–62.

Bajcsy, R. (1988). Active perception. In *Proceedings of the IEEE, Special issue on Computer Vision*, volume 76, pages 996–1005.

Ballance, R. A., Maccabe, A. B., and Ottenstein, K. J. (1990). The program dependence web: A representation supporting control-, data-, and demand-driven interpretation of imperative languages. In *Proceedings of the ACM SIGPLAN 90 Conference on Programming Language Design and Implementation*, volume 25, pages 257–271.

Ballard, D. H. (1991). Animate vision. *Artificial Intelligence*, 48(1):57–86.

Ballard, D. H., Hayhoe, M. M., and Pelz, J. B. (1995). Memory representations in natural tasks. *Cognitive Neuroscience*.

Bauckhage, C., Wachsmuth, S., Hanheide, M., Wrede, S., Sagerer, G., Heidemann, G., and Ritter, H. (2008). The visual active memory perspective on integrated recognition systems. *Image and Vision Computing*, 26(1):5–14.

Beauchemin, S. S. and Barron, J. L. (1995). The computation of optical flow. *ACM Computing Surveys*, 27(3):433–467.

Bishop, C. M. (2006a). *Pattern Recognition and Machine Learning*, chapter 9, pages 430–439. Springer.

Bishop, C. M. (2006b). *Pattern Recognition and Machine Learning*, chapter 8, pages 393–418. Springer.

Blum, A. L. and Furst, M. L. (1997). Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300.

Boykov, Y. Y. and Jolly, M. P. (2001). Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *Proceedings of the 8th IEEE International Conference on Computer Vision*, volume 1, pages 105–112.

Braumann, U.-D., Böhme, H.-J., and Gross, H.-M. (1995). An episodic knowledge base for object understanding. In *Proceedings of the European Symposium on Artificial Neural Network*, pages 175–180.

Bridgeman, B., Hendry, D., and Stark, L. (1975). Failure to detect displacement of the visual world during saccadic eye movements. *Vision Research*, 15(6):719–722.

Broomhead, D. S. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*.

Buswell, G. T. (1935). *How People Look at Pictures*. University of Chicago Press.

Chun, M. M. and Nakayama, K. (2000). On the functional role of implicit visual memory for the adaptive deployment of attention across scenes. *Visual Cognition*, 7(1):65–81.

Crestani, F. (1997). Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, 11:453–582.

Do, M. N. and Vetterli, M. (2002). Wavelet-based texture retrieval using generalized gaussian density and kullback-leibler distance. *IEEE Transactions on Image Processing*, 11(2):146–158.

Eggert, J., Rebhan, S., and Körner, E. (2007). First steps towards an intentional vision system. In *Proceedings of the 5th International Conference on Computer Vision Systems*.

Eggert, J. and Wersing, H. (2008). Approaches and challenges for cognitive vision systems. In *Creating Brain-like Intelligence*, number 5436 in LNCS, pages 215–247. Springer Verlag.

Einecke, N., Rebhan, S., and Eggert, J. (2008). Depth from perspective transformations. In Fujimura, K., Sendhoff, B., and Tsujino, H., editors, *5th HRI Global Workshop*.

Ferrante, J., Ottenstein, K., and Warren, J. D. (1987). The program dependence graph and its use in optimization. *ACM Transactions on Programming Languages and Systems*, 9(3):319–349.

Firtzke, B. (1995). A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632.

Frintrop, S., Backer, G., and Rome, E. (2005). Goal-directed search with a top-down modulated computational attention system. In *Proceedings of the 27th Annual meeting of the German Association for Pattern Recognition*, volume 3663 of *LNCS*, pages 117–124.

Fua, P. (1993). A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 6(1):35–49.

Gibson, J. J. (1966). *The Senses Considered as Perceptual Systems*. Houghton Mifflin, Boston.

Gonzalez, R. and Woods, R. E. (2002). *Digital Image Processing*. Prentice Hall Press.

Graham, R. L. (1972). An efficient algorith for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132–133.

Gross, H.-M., Böhme, H.-J., Heinke, D., Pomierski, T., and Möller, R. (1994). Self-organizing a behaviour-oriented interpretation of objects in active-vision. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 58–61.

Gross, H.-M., Heinke, D., Böhme, H.-J., Braumann, U.-D., and Pomierski, T. (1995). A behaviour-oriented approach to an implicit "object-understanding" in visual attention. In *IEEE International Conference on Neural Networks*, pages 657–662.

Gross, H.-M., Körner, E., Böhme, and H.-J., Pomierski, T. (1992). A neural network hierarchy for data and knowledge controlled selective visual attention. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 825–828.

Hamker, F. H. (1998). *Visuelle Aufmerksamkeit und lebenslanges Lernen im Wahrnehmungs-Handlungs-Zyklus*. PhD thesis, Technical University of Ilmenau.

Hamker, F. H. (2007). The mechanisms of feature inheritance as predicted by a systems-level model of visual attention and decision making. *Advances in Cognitive Psychology*, 3(1–2):111–123.

Hasler, S., Wersing, H., Kirstein, S., and Körner, E. (2009). Large-scale real-time object identification based on analytic features. In *Proceedings of the 18th International Conference on Artificial Neural Networks*, volume 5769 of *LNCS*.

Hayhoe, M. (2000). Vision using routines: A functional account of vision. *Visual Cognition*, (7):43–64.

Hayhoe, M., Bensinger, D., and Ballard, D. (1998). Task constraints in visual working memory. *Vision Research*, 38(1):125–137.

Heinke, D. (1997). *Selbstorganisation einer zeitlichen Objektrepräsentation*. PhD thesis, Technical Univerity of Ilmenau.

Heinke, D. and Gross, H.-M. (1993). A simple selforganizing neural network architecture for selective visiual attention. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 63–66.

Henderson, J. M. and Hollingworth, A. (2003). Eye movements and visual memory: Detecting changes to saccade targets in scenes. *Perception and Psychophysics*, 65(1):58–71.

Henderson, J. M., Weeks, P. A., and Hollingworth, A. (1999). The effects of semantic consistency on eye movements during complex scene viewing. *Experimental Psychology: Human Perception and Performance*, 25(1):210–228.

Herzog, G. and Wazinski, P. (1994). Visual translator: Linking perceptions and natural language descriptions. *Artificial Intelligence Review*, 8(2-3):175–187.

Hirschmüller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 807–814.

Hollingworth, A. (2005). The relationship between online visual representation of a scene and long-term scene memory. *Experimental Psychology: Learning, Memory and Cognition*, 31(3):396–411.

Horn, B. K. P. and Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*, 17:185–203.

Intraub, H. (1997). The representation of visual scenes. *Trends in Cognitive Sciences*, 1(6):217–221.

Irwin, D. and Andrews, R. (1996). *Attention and performance XVI*, chapter 6, pages 125–155. MIT Press, Cambridge, MA.

Irwin, D. E. (1992). Memory for position and identity across eye movements. *Experimental Psychology: Learning, Memory and Cognition*, 18(2):307–317.

Itti, L. and Koch, C. (2000). A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40(10–12):1489–1506.

Itti, L. and Koch, C. (2001). Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203.

Jonides, J., Irwin, D. E., and Yantis, S. (1982). Integrating information from successive fixations. *Science*, 215(4529):192–194.

Just, M. A. and Carpenter, P. A. (1976). Eye fixations and cognitive processes. *Cognitive Psychology*, 8(4):441–480.

Kahneman, D., Treisman, A., and Gibbs, B. J. (1992). The reviewing of object files: Object-specific integration of information. *Cognitive Psychology*, 24(2):175–219.

Kirstein, S., Wersing, H., Gross, H.-M., and Körner, E. (2008). A vector quantization approach for life-long learning of categories. In *Proceedings of the International Conference on Neural Information Processing*, pages 803–810.

Kohonen, T. (1989). *Self-Organization and Associative Memory*. Springer-Verlag, third edition.

Kohonen, T. (1990). The self-organizing map. In *Proceedings of the IEEE*, volume 78, pages 1464–1480.

Langley, P., Choi, D., and Rogers, S. (2009). Acquisition of hierarchical reactive skills in a unified cognitive architecture. *Cognitive Systems Research*.

Langseth, H. and Nielsen, T. D. (2006). Classification using hierarchical naive bayes models. *Machine Learning*, 63(2):135–159.

Lloyd, S. P. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.

Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. pages 674–679.

Luck, S. J. and Vogel, E. K. (1997). The capacity of visual working memory for features and conjunctions. *Nature*, 390:279–281.

MacKay, D. (1973). *Handbook of senory physiology*, chapter Visual stability and voluntary eye movement, pages 307–331. Springer, Berlin.

Marr, D. (1982). *Vision: a computational investigation into the human representation and processing of visual information*. W. H. Freeman, San Francisco, San Francisco.

McConkie, G. W. and Rayner, K. (1976). Identifying the span of the effective stimulus in reading. Technical report, Cornell University, Ithaca, NY. School of Education.

McNeill, G. and Vijayakumar, S. (2005). 2d shape classification and retrieval. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1483–1488.

Michalke, T., Fritsch, J., and Goerick, C. (2008). Enhancing robustness of a saliency-based attention system for driver assistance. In *Proceedings of the 6th International Conference on Computer Vision Systems*, pages 43–55.

Morel, J.-M., Petro, A. B., and Sbert, C. (2009). Fast implementation of color constancy algorithms. volume 7241.

Mumford, D. and Shah, J. (1989). Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*.

Navalpakkam, V. and Itti, L. (2005). Modeling the influence of task on attention. *Vision Research*, 45(2):205–231.

Navalpakkam, V. and Itti, L. (2007). Search goal tunes visual features optimally. *Neuron*, 53:605–617.

Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V. (2007). *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA.

O'Regan, J. K. (1992). Solving the "real" mysteries of visual perception: the world as an outside memory. *Canadian Journal of Psychology*, 46(3):461–488.

O'Regan, J. K. and Lévy-Schoen, A. (1983). Integrating visual information from successive fixations: does trans-saccadic fusion exist? *Vision Research*, 23(8):765–768.

Pashler, H. (1988). Familiarity and visual change detection. *Perception and Psychophysics*, 44(4):369–378.

Pearl, J. (1982). Reverend bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the American Association of Artificial Intelligence National Conference on Artificial Intelligence*, pages 133–136.

Pomierski, T. and Gross, H.-M. (1996). Biological neural architectures for chromatic adaptation resulting in constant color sensations. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 734–739.

Powell, M. J. D. (1985). Radial basis functions for multivariable interpolation: A review. In *Proceedings of the IMA Conference on Algorihms for the Approximation of Function and Data*, pages 143–167.

Prankl, J., Antenreiter, M., Auer, P., and Vincze, M. (2009). Consistent interpretation of image sequences to improve object models on the fly. In *Proceedings of the 7th International Conference on Computer Vision Systems*, volume 5815 of *LNCS*, pages 384–393.

Pylyshyn, Z. (1989). The role of location indexes in spatial perception: a sketch of the finst spatial-index model. *Cognition*, 32(1):65–97.

Rebhan, S., Einecke, N., and Eggert, J. (2009a). Consistent modeling of functional dependencies along with world knowledge. In *Proceedings*

*of the International Conference on Cognitive Information Systems Engineering*, pages 341–348.

Rebhan, S., Richter, A., and Eggert, J. (2009b). Demand-driven visual information acquisition. In *Proceedings of the 7th International Conference on Computer Vision Systems*, volume 5815 of *LNCS*, pages 124–133.

Rebhan, S., Röhrbein, F., Eggert, J., and Körner, E. (2008). Attention modulation using short- and long-term knowledge. In *Proceedings of the 6th International Conference on Computer Vision Systems*, volume 5008 of *LNCS*, pages 151–160.

Rensink, R. A. (2000). The dynamic representation of scenes. *Visual Cognition*, 7(1–3):17–42.

Rensink, R. A., O'Regan, J. K., and Clark, J. J. (1997). To see or not to see: The need for attention to perceive changes in scenes. *Psychological Science*, 8(5):368–373.

Rensink, R. A., O'Regan, J. K., and J., C. J. (2000). On the failure to detect changes in scenes across brief interruptions. *Visual Cognition*, 7(1–3):127–145.

Riad, M. M., Röhrbein, F., Einecke, N., and Eggert, J. (2009). Exploiting spatial relational knowledge for visual cognitive tasks. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, pages 1535–1540.

Ristic, B., Arulampalam, S., and Gordon, N. (2004). *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House.

Röhrbein, F., Eggert, J., and Körner, E. (2007). Prototypical relations for cortex-inspired semantic representations. In *Proceedings of the 8th International Conference on Cognitive Modeling*, pages 307–312.

Rothkopf, C. A., Ballard, D. H., and Hayhoe, M. M. (2007). Task and context determine where you look. *Journal of Vision*, 7(14):1–20.

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1–3):7–42.

Schmüdderich, J., Willert, V., Eggert, J., Rebhan, S., Goerick, C., Sagerer, G., and Körner, E. (2008). Estimating object proper motion using optical flow, kinematics and depth information. *IEEE Transactions on Systems, Man, and Cybernetics*, 38(4):1139–1151.

Sedgewick, R. (1988). *Algorithms*. Addison-Wesley, second edition.

Simons, D. J. (2000). Current approaches to change blindness. *Visual Cognition*, 7(1–3):1–15.

Simons, D. J. and Ambinder, M. S. (2005). Change blindness. *Current Directions in Psychological Science*, 14(1):44–48.

Simons, D. J. and Levin, D. T. (1997). Change blindness. *Trends in Cognitive Sciences*, 1(7):261–267.

Sonka, M., Hlavac, V., and Boyle, R. (1998). *Image Processing, Analysis and Machine Vision*, chapter 5, pages 176–190. PWS Publishing, 2nd edition.

Soto, D., Hodsoll, J., Rotshtein, P., and Humphreys, G. W. (2008). Automatic guidance of attention from working memory. *Trends in Cognitive Sciences*, 12(9):342–348.

Toussaint, G. T. (1983). Solving geometric problems with the rotating calipers. In *Proceedings of IEEE MELECON*.

Treisman, A. and Gormican, S. (1988). Feature analysis in early vision: Evidence from search asymmetries. *Psychological Review*, 95(1):15–48.

Triesch, J., Ballard, D. H., Hayhoe, M. M., and Sullivan, B. T. (2003). What you see is what you need. *Journal of Vision*, 3(1):86–94.

Tsai, A., Wells, W., Warfield, S., and Willsky, A. (2005). An EM algorithm for shape classification based on level sets. *Medical Image Analysis*, 9(5):491–502.

Tsotsos, J. K. (1992). On the complexity of active vs. passive visual search. *International Journal of Computer Vision*, 7(2):127–141.

Ullman, S. (1984). Visual routines. *Cognition*, 18:97–159.

von Neumann, J. (1928). Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320.

Weiler, D. and Eggert, J. (2007). Multi-dimensional histogram-based image segmentation. In *Proceedings of the 14th International Conference on Neural Information Processing*, volume 4984 of *LNCS*, pages 963–972.

Willert, V., Eggert, J., Adamy, J., and Körner, E. (2006). Non-gaussian velocity distributions integrated over space, time and scales. *IEEE Transactions on Systems, Man and Cybernetics B*, 36(3):482–493.

Wolfe, J. M. (1994). Guided search 2.0 a revised model of visual search. *Psychonomic Bulleting & Review*, 1(2):202–238.

Yarbus, A. L. (1967). *Eye Movements and Vision*. Plenum Press, New York.

Zhang, C., Eggert, J., and Einecke, N. (2009). Robust tracking by means of template adaptation with drift correction. In *Proceedings of the 7th International Conference on Computer Vision Systems*, volume 5815 of *LNCS*, pages 425–434.

Zhang, H. (2004). The optimality of naive bayes. In *Proceedings of the 17th Florida Artificial Intelligence Research Society Conference*, pages 562–567.