MONTAN
UNIVERSITÄT
WWW.UNILEOBEN.AC.AT

# Automatic Scene Interpretation with Totally Occluded Objects

**Martin Antenreiter**

Dissertation
submitted to
Montanuniversität Leoben

in partial fulfilment of
the requirements for the degree of
Doktor der montanistischen Wissenschaften

Leoben, July 2016

*To Nicole*
*and my children Jana and Lena*

# Contents

# Abstract

In this thesis we propose and evaluate a customizable computer vision system with cognitive abilities that tracks multiple objects over a long period of time, even if several of the relevant objects are totally occluded by other objects. At any time the system gives plausible object locations of all relevant objects, independently of their visibility, by maintaining possible interpretations of the observed visual input data. We use an approach that combines bottom-up visual processing with top-down reasoning and inference. Furthermore, our computer vision system has learning capabilities. These learning capabilities are used to obtain more robust tracking results if the visual appearance of relevant objects changes gradually over time. Our modular vision system allows to use several tracking algorithms from the literature, as long as they fit our minimum interface requirements. Template trackers, mean-shift trackers, and interest-point based trackers are employed to show the adaptability of our vision system.

Consequently, in the second part of this thesis, we study the effect of combining different types of low-level visual features. The key intuition is that a system using a rich set of low-level visual features should be more robust than a system relying on only a single visual feature. The problem is which visual features are well suited for a specific relevant object. A method is proposed which combines the outcome of detectors based on different visual features using a support vector machine (SVM). Our feature combination is tested on the standard Visual Object Classes challenges (VOC) datasets. Results on the VOC datasets show that our method significantly improves over the performance of detectors which use only a single visual feature.

# Kurzfassung

In dieser Arbeit wird ein Computer Vision System mit kognitiven Fähigkeiten vorgeschlagen und evaluiert. Dieses System verfolgt mehrere Objekte über einen langen Zeitraum, sogar wenn diese Objekte vollkommen von anderen Objekten verdeckt sind. Zu jedem Zeitpunkt liefert das System plausible Positionen der relevanten Objekte, unabhängig von der aktuellen Sichtbarkeit der Objekte. Dies wird dadurch erreicht, dass das System mögliche Interpretationen auf Grundlage der beobachteten visuellen Eingangsdaten erzeugt. Der verwendete Ansatz kombiniert eine Bottom-up-Verarbeitung mit einer Top-down-Schlussfolgerung. Das System hat zusätzlich die Fähigkeit zu lernen. Diese Lernfähigkeit wird dazu verwendet, um Objekte besser verfolgen zu können, wenn diese ihr visuelles Aussehen über den Beobachtungszeitraum langsam verändern. Das modulare System ermöglicht es, verschiedene Algorithmen zur Objektverfolgung aus der Literatur zu verwenden, wenn diese die geforderte Minimalschnittstelle erfüllen können. In den Versuchen werden Template-Tracker, Mean-Shift-Tracker und Interest-Point basierende Tracker eingesetzt, um die Anpassungsfähigkeit des Systems zu zeigen.

Im zweiten Teil der Arbeit wird die Kombination von verschiedenen visuellen Merkmalen untersucht. Die Intuition dahinter ist, dass ein System mit vielen visuellen Merkmalen stabiler funktionieren sollte als ein System welches nur ein visuelles Merkmal verwendet. Das Problem ist jedoch, dass die richtigen visuellen Merkmale für ein bestimmtes Objekt zuerst ermittelt werden müssen. Es wird ein Verfahren vorgeschlagen, welches die Ergebnisse von verschiedenen Detektoren mit unterschiedlichen visuellen Merkmalen kombiniert. Die Kombination der Detektoren wird mit der Hilfe einer Support-Vector-Maschine (SVM) erreicht und mit den Datensätzen der Visual Object Classes Challenge (VOC) getestet. Die Ergebnisse auf den VOC Datensätzen zeigen, dass die kombinierten Detektoren signifikant besser sind als jene die nur ein visuelles Merkmal verwenden.

This page intentionally contains only this sentence.

Chapter 1

# Introduction

## 1.1 Problem Definition

An important research field in computer vision is object tracking [72, 18, 93, 71, 35]. There exists a vast number of algorithms for object tracking, and typically these algorithms rely on assumptions like constant illumination, uniform background, smoothness of motion or a minimal amount of occlusion. In constraint environments these assumptions can be fulfilled and thus the trackers work well and mostly in real-time.

In realistic environments not all assumptions are satisfied, though. This violation of assumptions limits the applicability in automated surveillance, traffic monitoring, vehicle navigation, and personal assistance system. Most of these tracking algorithms ignore that objects of interest may be occluded by other objects and—in the worst case—are not visible during an extended period of time. Only a few publications focus on the problem of occlusion in video sequences [76, 11, 5, 12, 89, 43]. In contrast, humans deal with occlusion events very well, e.g. experiments have shown that even infants at the age of 4 month accurately predict the reappearance of objects after total occlusion [109, 44]. Hence our goal is to build a vision system with cognitive abilities that is able to give possible object locations even if the object of interest is totally occluded over an extended period of time.

As motivating example a video sequence from a surveillance camera is shown in Figure 1.1. In the first row of Figure 1.1, the person opens the car door and enters the car. After that, the person closes the door, starts the engine, and drives the car until it crashes into another car. There is no visual evidence of the person, but a human is able to give answers to the

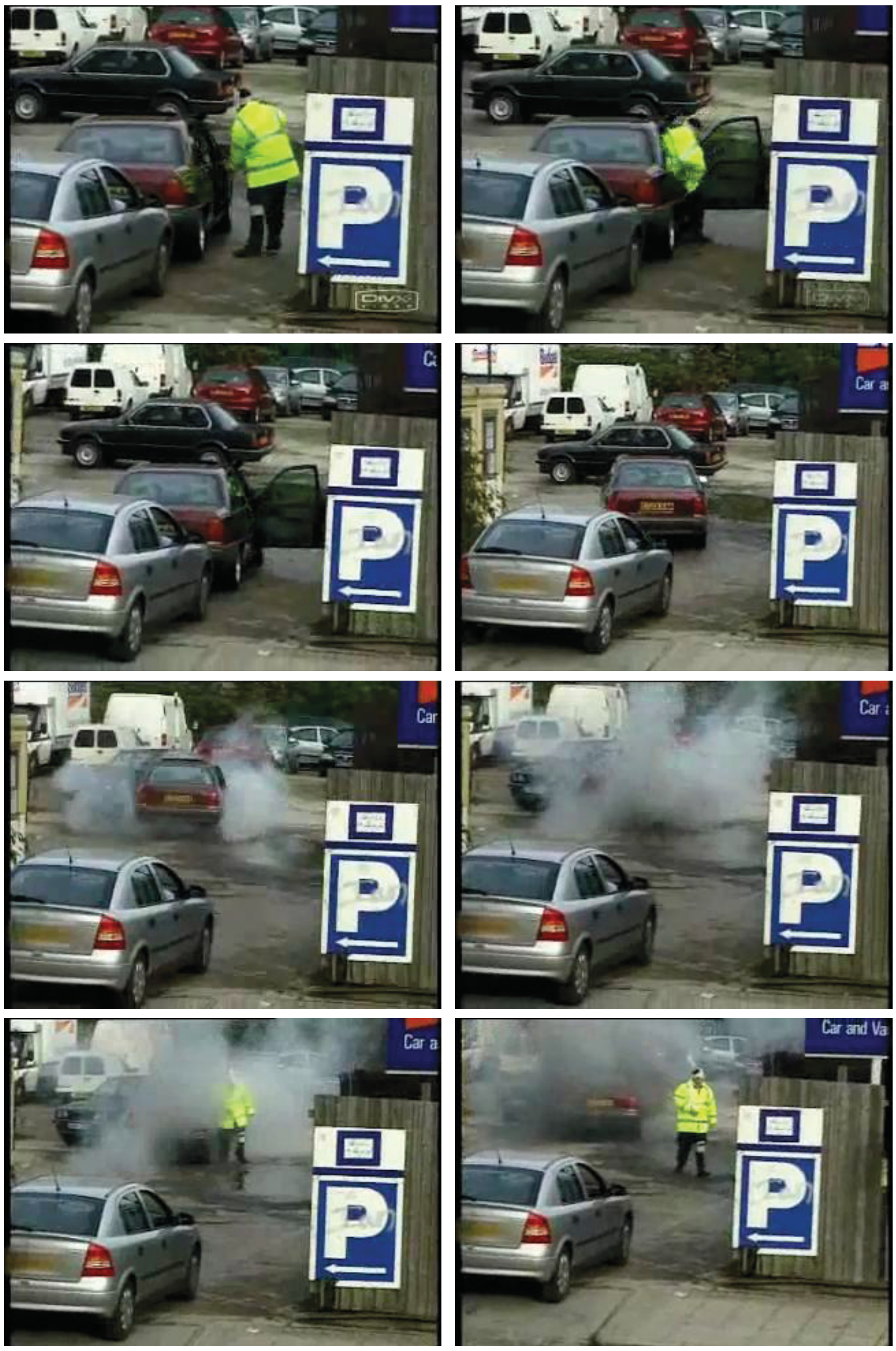


Figure 1.1: Images of a surveillance camera: A person enters a car and crashes this car into another car. In some frames there is no visual evidence of the person and the car. A human can predict the positions of both. A cognitive vision system should be able to give similar position predictions.

following questions:

1. Where is the person after the third image?
2. Where can the person reappear, after the car is driven to a different location?

Simple tracking algorithms cannot answer these questions, because they only rely on visual information.

In this work we follow the idea that a customizable vision system with cognitive abilities is divided into modules, in particular into a high-level reasoning module and a low-level vision module. The high-level reasoning module consists of all the components that are needed to answer the above questions. The low-level vision module is composed of tracking algorithms from the literature. These tracking algorithms are separate components and we term them as low-level vision components.

The objectives of our cognitive vision system are:

1. It can describe the interactions between the relevant objects in a video stream.
2. It can give plausible object locations of all relevant objects even if several of the relevant objects are totally occluded by others.
3. It can learn new visual appearance information of the relevant objects to improve future tracking results.

We model the cognitive vision system given the objectives above and the basic structure using a high-level reasoning module and a low-level vision module. Based on this structure we describe a possible solution. We introduce interfaces between the modules and describe the minimum requirements of each module.

If the visual input is unambiguous then the system relies on bottom-up processing. This is the common method to process visual input. Bottom-up and top-down processing are used in combination, when objects interact and ambiguities arise. For our solution we define interface requirements that can be easily fulfilled by many existing tracking algorithms. Our main requirement for low-level vision components is that they can provide confidence values for object locations. Additional possible object locations and their related confidence values may be requested by the high-level reasoning module. With this approach, special properties of the tracking algorithms are ignored, and we obtain a simple and generic interface between the high- and the low-level modules. In this work we show the capabilities of a system with such a generic interface between low-level vision module and high-level reasoning module.

Our approach for the high-level reasoning module is based on the idea that a system which builds a consistent scene interpretation using all relevant objects, is more robust than a system which uses only a set of independent object trackers. Therefore, our system maintains possible interpretations of the observed visual input data. We show that for each interpretation a quality measure can estimate the consistency of the interpretation. The most consistent interpretations can be used to answer question like: "Where will a person reappear, after driving a car to a different location?", as in the above motivating example.

## 1.2 Cognitivism Paradigm

There are different types of cognition and several research paradigms. In this section we explain the cognitivism paradigm which we follow in this thesis. The cognitivism paradigm assumes that cognitive behavior is a type of information processing. Input data is fed into a cognitive system, it is processed and new generated data is stored into a knowledge base. Based on the data from the knowledge base the cognitive system acts accordingly and shows cognitive behavior. Traditional artificial intelligence treats this aspect of cognitive behavior. Information about the world is represented by abstract symbols, called facts. These facts are processed by rules to generate new facts. Therefore, these rules are often referred to as production rules. Production rules represent knowledge about the world. This knowledge is often domain specific. An inference engine controls which facts and rules are applied in a given situation. New facts can be used to plan or act in the world. This approach is called the information processing approach to cognition [73, 48, 87, 58].

### 1.2.1 Problems of the Cognitivism Paradigm

Systems with deterministic inference engines were successfully applied to organic chemistry [16], medical diagnosis [15, 49], and computer configuration [75]. Some of these rule-based systems performed as well as experts in their fields. An overview of those systems can be found in [110]. However, even thought there were successful systems, the cognitivism paradigm failed in other domains. The major points of criticism were the *frame problem*, the *symbol grounding problem*, and the *combinatorial explosion problem*. We discuss all three problems and their influences on our solution.

## The Frame Problem

The frame problem is the difficulty to formally handle properties of the current state that are not affected by an action or event. It was first described as a technical problem in deductive logic based reasoning engines [74]. In a frame we have a set of properties with assigned values. This properties define the current state of our problem domain. An action is executed and as a result, we have a new frame with almost the same values for most of the properties. Only a few properties will change from one frame to the next frame. These consequences can be defined by a rule. However, in deductive logic we also have to define rules for all the unchanged properties, otherwise the reasoning engine cannot conclude that some properties have not changed. This is impractical.

An example with two actions shows the frame problem. If we want to describe two actions *placing* and *painting* an object A, then we have two rules of the form

1. "Object A has color X" holds after "*painting* object A with color X"

2. "Object A is in object Y" holds after "*placing* object A in object Y"

Now, we assume the initial state of object A as following: object A has color red and is placed on a desk. First, the color of the object is changed by painting it blue. After that, the object is placed in a drawer. Intuitively, we can conclude that the object A is now in the drawer and has color blue. However, this cannot be concluded by a reasoning engine using deductive logic. The problem is that one rule is missing. We do not have a rule which states that the color of an object does not change during placement. Thus, we can only conclude that the object A is in the drawer. If we want a system which can conclude the intuitively correct results, then we have to add two additional rules:

1. “Object A has color X” holds after “placing object A in object Y” if “object A had color X before”
2. “Object A is in object Y” holds after “painting object A with color X” if “object A was in object Y before”

Writing such rules for actions are annoying and leads to errors if rules are forgotten. Suppose, we have N properties for an object and M actions to modify the properties, then we have to specify $N \times M$ rules expressing the effect of the actions, even if most of the actions effect only one property.

One solution to this problem is that everything remains unchanged if not explicated mentioned; this is called *the common sense law of inertia*. A formal solution was introduced in [95] and Shanahan concluded in [104] that the frame problem is solved from a mathematical viewpoint.

In this thesis we follow the approach that everything remains unchanged if not explicated mentioned and we define rules only for changing properties. Our problem of tracking occluded objects is quite specific and therefore only a few rules have to be defined (see Sec. 2.4.3).

### The Symbol Grounding Problem

Harnad defines the *Symbol Grounding Problem* [45] using Searle’s Chinese Room argument [101]: A human who only understands English is sitting in a room with a rule book written in English, paper sheets, and a pencil. The door has a slot where paper sheets can be exchanged with a person outside of the room. The rule book gives instructions on how to deal with incoming Chinese characters. Further, it gives instructions which Chinese characters have to be drawn on an empty sheet and passed through the slot as an answer. A Chinese-speaking expert—sitting outside the room—can throw a question on a sheet of paper into the room. Chinese characters are used to express the question. The human inside generates an appropriate answer with the help of the rule book and writes the answer in Chinese characters on a sheet and throws it through the slot. Thus, a Chinese-speaking expert can communicate with the English speaking person inside the room. The expert can be convinced that he or she is communicating with another Chinese-speaking person.

Searle’s Chinese Room is an illustrative example to argue about cognitive behavior. Harnad argues in [45] that the human inside the room is manipulating symbols and does not understand the meaning of the symbols, therefore this information processing cannot be cognitive behavior. The room can be seen as a computer. The computer program is the rule book and the central processing unit (CPU) of a computer is the human operator inside the room. Harnad writes in [45] that understanding cannot be just symbol manipulation and that symbols have to be grounded bottom-up by sensor data. Otherwise, the system processes meaningless symbols and produces new meaningless symbols.

Alan Turing proposed the Turing test in [113]. It is intended to test the presence of intelligence in machines. The test consists of an interrogator and two test candidates sitting in separated rooms. One of the test candidates is a computer and one a human. The interrogator uses a keyboard to communicate with both test candidates. The interrogator's task is to determine which candidate is the computer. Therefore, the interrogator asks questions to both candidates. The human operator sees the questions on a screen and uses a keyboard for writing the answers. The computer and the human operator send their answers back to a screen device in the interrogator's room. After the conversation the interrogator has to decide which of the two candidates is the computer. If the interrogator can only guess then the computer has passed the Turing test, because it shows intelligent behavior like a human.

Searle argues in [101] that passing the Turing test does not imply understanding. In this work we ignore the philosophical question of what constitutes understanding. We are just interested in the external behavior of the system.

### The Combinatorial Explosion Problem

In 1973 the Lighthill report [65] criticized that artificial intelligence suffers from the combinatorial explosion problem. The combinatorial explosion problem is the problem that an agent takes an enormous amount of time to execute an single action because it is searching for the perfect sequence of actions to reach a goal. The problem is encountered when an agent can in every step choose from multiple actions to reach different states. Only very small problems with a limit number of actions and states can be fully solved in this way.

The combinatorial explosion can be seen by analyzing chess. If a computer wants to find the best move for a given chess position then we have to try out all legal moves from the starting position. After that, we have to find the best move for the opponent and also try out all legal moves; followed by the search for the next best move of the computer. This procedure is repeated until the end of the game is reached. After calculating all possible plays we can choose a play that results in a win. C. E. Shannon analyzed the chess game in this way in [105]. The mean number of legal moves in a typical chess position is 32.3 [24, 25, page 22]. Therefore, we have approximately $10^3$ different chess positions after one move by white and black. If we assume a typical game lasts 40 moves then there will be $10^{120}$ different possible games (game-tree complexity). A computer which analyzes $10^9$ games per second will need approximately $3.2 \cdot 10^{106}$ years for the first move. This brute-force search is impractical for many practical problems. Therefore, artificial intelligence researchers have discovered good heuristics for reducing the combinatorial explosion. Nowadays, chess programs use an evaluation function which evaluates if a given variation is worth to be examined. The search space is reduced enormously after rejecting many variations after a few move calculations. After 47 years of research the computer Deep Blue from IBM [3, 17] won against world chess champion Garri Kasparov in 1997.

In this thesis we propose a heuristic to reduce the search space for cognitive vision application. Our proposed approach will not compute all possibilities, it has to compute all reasonable ones.

Chapter 2

# System Architecture

This chapter explains the desired properties of the system. It also proposes a suitable architecture with its interfaces. Antenreiter and Auer first described this architecture in [5].

## 2.1 Design Goals

### 2.1.1 Interaction of Low-Level Objects Appearance and High-Level Location Reasoning

Image processing can be seen as a simple *bottom-up* process. A bottom-up process extracts low-level features from an image such as edges, corners, and blobs. Next, it groups images features to parts and aligns these parts to a 3D model. The 3D model is assigned to a particular object [14]. The object represents high-level information. This high-level information can be used to interpret the scene, e.g. how objects are interacting in a given scene. Thus, bottom-up processing accumulates low-level information to build high-level information.

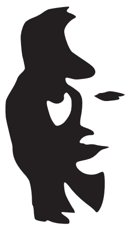


Figure 2.1: An ambiguous image [106].

However, images can be ambiguous; and therefore, biological vision systems apply a more complex processing. Figure 2.1 shows an ambiguous image. It shows a woman's face in shadow, or the silhouette of a man playing a saxophone. The same low-level image features have two interpretations.

This example exposes that the human vision system involves *top-down* processing. A hypothesis is proposed and its validity is verified. Also, neuroscientists have shown that feedback structures from higher-level processing exist in the macaque monkey vision system [98, 97]; and similar feedback structures have been found in the cat vision system [107, 108]. Thus, higher-level visual processing areas transfer information to the low-level areas.

Biological vision systems seem to use bottom-up and top-down processing, a so-called hybrid control structure [91]. This motivates the control structure for our system. The goal is that the architecture supports bottom-up as well as top-down processing.

### 2.1.2 Interchangeability of Components and Interface Simplicity

Complex system are usually divided into components. Each component has defined interfaces to other components. This approach reduces the dependencies between components, and components can be more easily exchanged if components with different properties are needed [83]. Therefore, components should be connected only via well defined interfaces. This simplifies the communication between them.

The next goal is that the interface between the low-level vision processing and high-level vision processing can be fulfilled by many image processing algorithms. Interfaces which rely on special properties of a specific image processing algorithm are not desirable. Therefore, the interfaces should be simple and generic. This results in a system where the components can be easily exchanged. For example, if objects can only be distinguished by their color information, then a suitable image processing algorithm will be an algorithm that uses color as a visual feature. In a new scenario all objects have the same color and they can only be distinguished by their contours. It should be enough to exchange the low-level image processing algorithm to adapt the system to the new scenario.

However, one drawback of this approach is that the system cannot use special properties of the image processing algorithm, even if these special properties would simplify the work in the high-level vision processing. This disadvantage must be accepted if a simple and generic interface is favored.

## 2.2 Overview of the Proposed Architecture

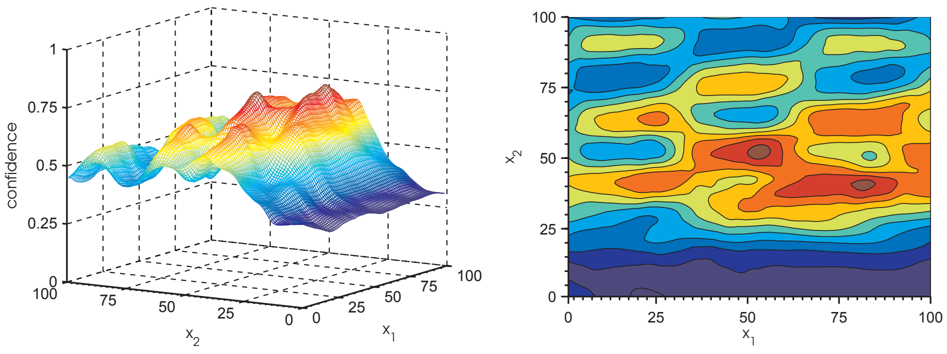
Our proposed system consists of two main modules. The first module processes the incoming image data and reports possible object locations (bottom-up processing). The second module builds possible scene interpretations (bottom-up processing) and refines these interpretations incorporating additional data from the earlier interpretations (top-down processing).

The first module extracts low-level features from the images. Therefore, its name is *low-level vision module.* The low-level vision module is decomposed into *low-level vision components.* A *low-level vision component* exists for each object of interest. It works with a suitable vision algorithm for detecting the object and uses a model of the object's visual appearance. If the low-level vision component detects the object, then it reports the position with a confidence value to the second module—the *high-level reasoning module.*

The high-level reasoning module collects all reported positions and confidence values of all objects. The confidence value gives an indication of how well the object model fits the visual appearance. Positions and confidence values are used to build *hypotheses.* A hypothesis is an assumption about the states of all relevant objects; and thus, an interpretation of the scene. It is constructed from the vision inputs of the current input image and from one of the hypotheses about the previous input image.

The high-level reasoning module has five main components. It consists of the *vision controller*, the *hypothesis generator*, the *hypothesis inspector*, the *hypothesis graph*, and the *hypothesis visualizer*. The vision controller sends commands to the low-level vision components in the low-level vision module. The state of an object in a hypothesis affects the commands for the low-level vision component, such as requests to detect a specific object at a certain location, or the command to update the object model with new visual features at a specified position.

When new detection results are available, the hypothesis generator constructs with *construction rules* new hypotheses. Construction rules are general rules of how physical objects can interact in the physical world. The new hypotheses are inspected by the hypothesis inspector. The hypothesis inspector uses an *evaluation function* to evaluate the consistency of a sequence of hypotheses.



(a) 3d plot (b) Heat map

Figure 2.2: Detector's confidence values are shown from a region of interest of 100 × 100 pixels. Figure 2.2(a) is a 3d-plot of the confidence values. Figure 2.2(b) is a heat-map of the confidence values. More than one local maximum exists in the detection result. In this case, the low-level vision component reports all good detection results to the high-level reasoning component.

The data storage of the high-level reasoning module is the hypothesis graph. The hypothesis graph stores all maintained hypotheses. A hypothesis has forward and backward connections to hypotheses for the previous and the next frame.

The last component is the hypothesis visualizer. It displays a selected hypothesis as an image. The hypothesis visualizer draws the boundaries of the objects, even the estimated positions of the occluded objects, into the corresponding input image. Additionally, a textual description of the hypothesis can be generated.

### 2.2.1 Low-Level Vision Module

In this section, the low-level vision module is described in more detail. The low-level vision module is responsible for the processing of the image data. The module is divided into low-level vision components. Each component is assigned to an object in the scene. The visual properties of an object determine suitable low-level vision components for detection. Every low-level vision component is usable if it is able to detect the object reliably and if it implements the following interface. This interface between the low-level vision components and the high-level reasoning module consists of five commands issued by the high-level reasoning module:

1. **Search:** The low-level vision component searches in a region of interest and reports all likely locations of an object. Figure 2.2 shows a typical detection result of a template detector. The low-level vision component may find many local maxima in the region of interest. In our proposed system the detector does not decide which local maximum is the best one. The low-level vision component only reports good candidates of possible locations. Therefore, the low-level vision component does a pre-selection. However, the decision about the best location of an object is done in the high-level reasoning module. In order to accomplish this, we require that the low-level vision component reports a confidence value between zero and one for each detection. A value of zero means: 'no object is detected at that position' and a value of one means: 'object is detected with very high probability'. Additionally, we require that the low-level vision component reports the boundary of the object as a two dimensional polygon. The Java code for this interface is

   ```
   class DetectorResult {
       double confidence;
       Polygon boundary;
   }
   ```

   and

   ```
   List<DetectorResult> searchWindow(int x, int y,
                                     int width, int height).
   ```

   A rectangle specifies the region of interest; it is defined by the upper-left $(x, y)$ coordinate, width, and height.

2. **Update:** The low-level vision component updates its appearance model of the object with pixel information at a given position. The reasoning module reports the visible object area. The low-level vision component can use this information to update the appearance model. Therefore, the definition for the update command is

   ```
   void updateModel(List<Polygon> visibleBoundaries).
   ```

   A correctly updated model, with the given visible boundaries, gives better detection results in future frames. A wrong update unlearns the appearance model. Then, the detection rates and confidence values of the component will decrease in future frames. As a consequence, the high-level reasoning can release components with wrong appearance models, see below.

3. **Set location:** The object's location is provided by the high-level reasoning module. The high-level reasoning module adapts the boundary of the object according to a hypothesis state.

   ```
   void setLocation(Polygon boundary)
   ```

   This operation can be used when the low-level vision component maintains an internal state on the position of an object, e.g. a Kalman tracker [56], and the high-level reasoning module decides that the visual input data is not informative enough; for example, when an object is highly occluded by other objects.

4. **Clone:** The low-level vision component creates a copy of itself and passes it to the high-level reasoning module. The command is defined as

   ```
   Detector clone().
   ```

   In ambiguous situations—a low-level vision component reports two or more good detection results— a correct update of the appearance model may be difficult. Therefore, the high-level reasoning module can duplicate the detector and update both detectors with different visual information.

5. **Release:** The low-level vision component is released. This operation can be used if the high-level reasoning module is confident that it does not need the low-level vision component anymore.

   ```
   void release()
   ```

### 2.2.2 High-Level Reasoning Module

Figure 2.3 depicts the main parts of our system, the processed video, and the resulting hypotheses graph. In addition, dashed lines with arrows show the data flow in the system. The processing steps are explained using Figure 2.3.

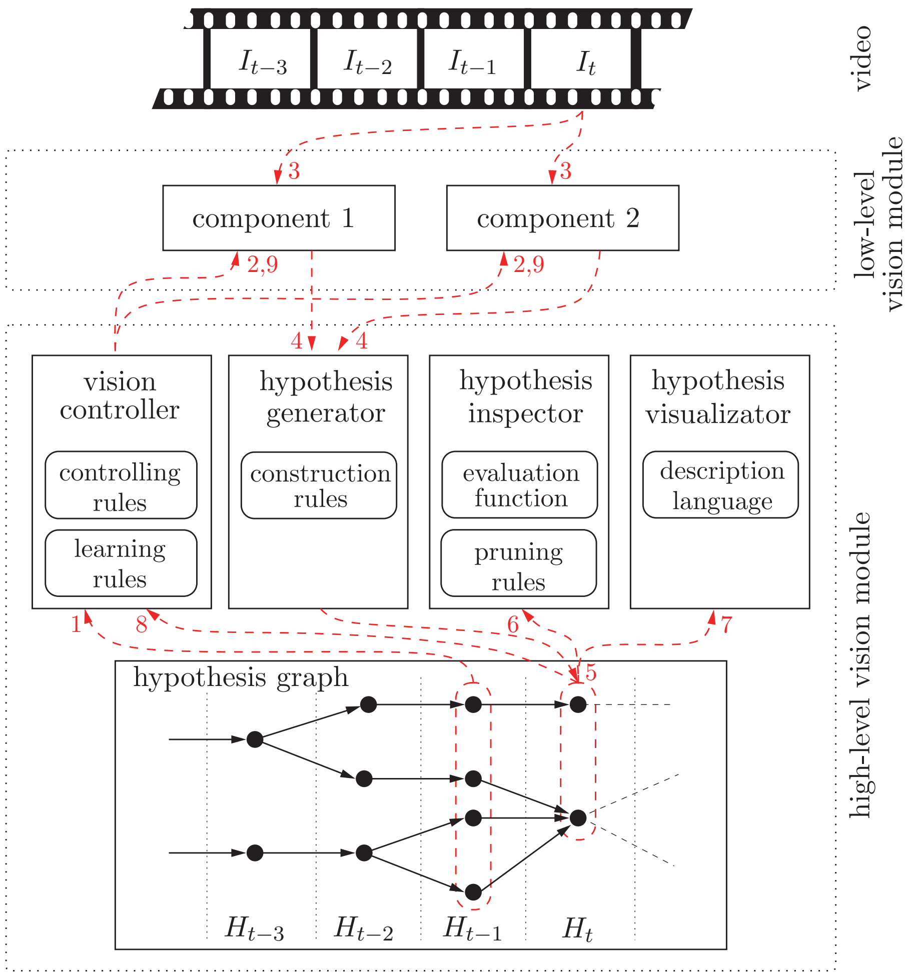


Figure 2.3: Overview of the system architecture with data flow. Dashed lines with arrows show the data flow. The numbers near the arrows indicate the processing order.

It is assumed that a video was processed until image $I_{t-1}$. According to Figure 2.3, there exists a set $H_{t-1}$ with four hypotheses for the image $I_{t-1}$; these hypotheses are marked as black dots in the hypothesis graph.

In the first step, the vision controller reads all four hypotheses from set $H_{t-1}$ (step 1). It sends control commands to low-level vision components depending on the state of the hypotheses (step 2). The low-level vision components process the next image $I_t$ (step 3). The hypothesis generator receives the detection results (step 4). It generates new hypotheses using the construction rules. If additional visual information is needed for hypotheses, then the hypothesis generator is handing over the control to the vision controller again. The steps 2–4 are repeated until no new hypotheses needs additional visual information from image $I_t$. After that, the hypothesis generator stores all generated hypotheses $H_t$ into the hypothesis graph (step 5). In the sixth step, the hypothesis inspector evaluates the generated hypotheses. It uses an evaluation function to calculate for each hypothesis a confidence value (step 6). Next, the hypothesis inspector marks unlikely hypotheses according to the pruning rules. Marked hypotheses are ignored in subsequent images. The hypothesis visualizer displays the most likely hypothesis on a screen (step 7).

The last steps are the clean-up and learning phase. First, the vision controller reads all hypotheses from set $H_t$. The hypotheses are analyzed (step 8). The vision controller releases resources for unlikely hypotheses. For the remaining hypotheses, it generates learning commands depending on the objects' states. These learning commands are sent to the low-level vision components (step 9). After that, the system is ready to process the next image of the video sequence.

## 2.3 Components of the Low-Level Vision Module

In this section all low-level vision components which were tested with the system are introduced. There exist various methods for tracking objects. An overview can be found in [119] where trackers are grouped by their methods. However, there exists no general-purpose tracker. The visual properties of the target objects influence the selection of the low-level vision components. Therefore, a subset of the described low-level vision components is used simultaneously in the test videos.

### 2.3.1 Adaptive Template Tracker

An advantage of a template tracker is that it stores spatial and appearance information of an object. Template trackers gave good localization results for two test videos. Template trackers work well if the appearance does not change over time. This is partially fulfilled in some test videos because objects are shown from their side view most of the time.

The tracker is made more robust against modest changes of the appearance by learning. It stores the original template $T_0$ and an adaptive template $T_t$. The high-level reasoning module can update the adaptive template to incorporate new appearance information. Eq. (2.1) defines a weighted average of appearance information using a learning rate $\alpha$.

$$\begin{aligned} T_t &= \alpha \cdot T_{t-1} + (1-\alpha) \cdot {I_t}^{x,y} && (2.1) \\ {T_t}^{\beta} &= \beta \cdot T_0 + (1-\beta) \cdot T_t && (2.2) \end{aligned}$$

${I_t}^{x,y}$ is a region at location $(x, y)$ in frame $t$. The region has the same size as the template $T$. The location $(x, y)$ is obtained by template matching or by the high-level reasoning module. Eq. (2.2) with the parameter $\beta$ controls the amount of information used from the original template and the adaptive template. If $\beta$ is set to one then the tracker only uses the information from original template $T_0$. On the other hand, if $\beta$ is set to zero it only uses the information from the adaptive template.

As measure of similarity between templates and image regions the normalized cross-correlation $c^{x,y}$ is used. The definition of the normalized cross-correlation $c^{x,y}$ is

$$c^{x,y} = \frac{\sum\limits_{x',\,y'} \left( {T_t}^{\beta}\left(x', y'\right) - \overline{T}_t^{\beta} \right) \cdot \left( I_t\left(x + x', y + y'\right) - \overline{I}_t^{x,y} \right)}{\sqrt{\sum\limits_{x',\,y'} \left( {T_t}^{\beta}\left(x', y'\right) - \overline{T}_t^{\beta} \right)^2 \cdot \sum\limits_{x',\,y'} \left( I_t\left(x + x', y + y'\right) - \overline{I}_t^{x,y} \right)^2}} \qquad (2.3)$$

where the summation $x', y'$ is over the pixels of the template.

$$\overline{T}_t^{\beta} = \frac{1}{rows \cdot cols} \sum_{x',\,y'} {T_t}^{\beta}\left(x', y'\right) \qquad (2.4)$$

$$\overline{I}_t^{x,y} = \frac{1}{rows \cdot cols} \sum_{x',\,y'} I_t\left(x + x', y + y'\right). \qquad (2.5)$$

It has several advantages compared to other similar measures like sum of absolute differences (SAD) or cross-correlation. First, the normalized cross-correlation $c^{x,y}$ is immutable against intensity changes across the image. Second, the value of $c^{x,y}$ does not depend on the size of the template. These properties are obtained by subtracting the mean of the template $\overline{T}_t^{\beta}$ from every template pixel ${T_t}^{\beta}\left(x', y'\right)$. Accordingly, the mean of the image region $\overline{I}_t^{x,y}$, where the template is placed at $(x, y)$, has to be calculated and subtracted from the image pixels $I_t\left(x + x', y + y'\right)$. The resulting values are

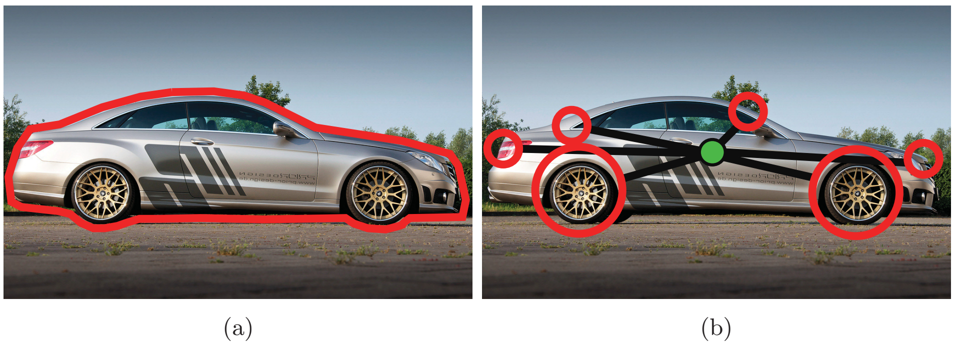


Figure 2.4: (a) A target object and its marked boundary (image source [90]). (b) A part based model which stores the spatial configuration of important parts to the object center (star model).

scaled to unity norm. Therefore, the normalized cross-correlation $c^{x,y}$ has a range from $-1$ to 1. A perfect match of template and image region gives a similarity of 1.

### 2.3.2 Interest-point Based Tracker

Another low-level vision component is a detector using a part based model. The model consists of small parts of the object where the spatial configurations between the parts are stored. Figure 2.4(a) shows an image with a marked target object. In Figure 2.4(b) important parts of the object are selected (red circles). A simple model is to store all spatial configuration between any two parts. This object representation was introduced by Agarwal et al. in [2]. An improved model was proposed by Leibe et al. [61]. It stores the spatial configuration of parts in respect to the object center. The model is a star-shaped representation. Figure 2.4(b) shows this representation. It is used in the low-level vision component. For each part the spatial configuration to the object center is stored (green circle in Figure 2.4(b)).

A part based object representation has several advantages. It is robust against partial occlusion of the target object. If a small number of parts cannot be detected then the object center can still be determined by the remaining parts. Part based models can also deal with local variations in object structure, like human body parts in different configurations. Additionally, if we use an interest-point detector for detecting the local parts and this interest-point detector returns a scale for each detected part, then the

**Algorithm 1:** Incremental clustering of the codebook entries.

```
Input: $codebook, descriptors, threshold$
Output: $codebook$
1  foreach $descriptor \in descriptors$ do
2      $min \leftarrow \infty$
3      $nearest \leftarrow descriptor$
4      foreach $entry \in codebook$ do
5          $d \leftarrow$ distance($entry$, $descriptor$)
6          if $d < threshold \wedge d < min$ then
7              $min \leftarrow d$
8              $nearest \leftarrow entry$
9          end
10     end
11     $codebook \leftarrow codebook \setminus nearest \cup$ merge($nearest$, $descriptor$)
12 end
13 return $codebook$
```

scale of an object can be inferred. The exhaustive search over all possible scales can be avoided.

Scale invariant interest-point detectors exist since several years. Their properties were studied extensively in [23, 67, 69, 81, 54, 78, 79, 10]. We have selected the Difference-of-Gaussian (DoG) operator for the detection of local parts. The SIFT descriptors [68] represent the local parts in the model. After applying the DoG operator and calculating the SIFT descriptors, the detector obtains for each detected local part a 2D position $(x_1, x_2)$, a scale $s$, an orientation $o$, and a 128-dimensional SIFT descriptor. The SIFT descriptor is scale invariant but varies with rotation. Therefore, the orientation of the SIFT descriptor has to be considered in a part based model.

### Building Codebook

In an ideal experimental setup with a static scene, the detector extracts equal SIFT descriptors for each frame. However, in a practical experiment the detector has to deal with noise, e.g. noise from the camera sensor. Additionally, the camera observes moving objects in a scene. A frame is a 2D projection of a 3D world; therefore, the local parts in a 2D projection can show up slightly distorted, depending on the movements of the 3D object. Hence, the algorithm averages SIFT descriptors detected from the current frame

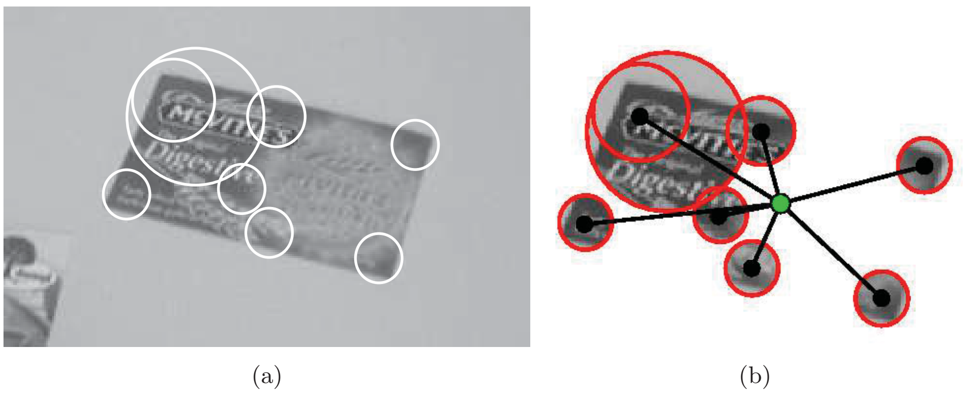


Figure 2.5: (a) Shows detected local parts as circles. A larger scale is represented by a larger diameter of a circle. (b) The star-shaped object representation is sketched with the centroid of the boundary. The lines from the centroid represent the recorded information of codebook entries and relative position to the centroid (Images from [7]).

with similar SIFT descriptors in the codebook. SIFT descriptors which are dissimilar to existing codebook entries are added as new codebook entries. Alg. 1 describes the incremental clustering procedure.

The averaged SIFT descriptors build our vocabulary of local appearances; it is called *codebook*. An averaged SIFT descriptor in the codebook is termed as *codebook entry*. The detector uses the codebook entries to construct a star-shaped object representation.

### Building Model

A target object has at least one star-shaped object representations, termed *model*. These models are build from training images. In every training image the boundary of the object is marked. The DoG operator is applied and all detected local parts within the boundary are used to build the model. The SIFT descriptors are computed from the local parts and matched with our codebook entries. The reference point of the model is the centroid of the boundary. The model records for each matched codebook entry the relative position to the centroid. Figure 2.5(a) shows the detected local parts of an object and Figure 2.5(b) an example of a star-shaped model.

### Detecting Objects

Objects are detected using a voting schema to find object centers. First, the DOG operator detects regions of interests. SIFT descriptors describe the regions of interest. Each SIFT descriptor is matched with stored codebook entries. After that, each model is considered and the detected SIFTs assigned to codebook entries vote for possible object centers. This voting is done with the *generalized Hough transform* [50, 9] using a four dimensional voting space. The detected local parts vote for a scale $s_{vote}$, an object location $\mathbf{x}_{vote} = (x_1, x_2)$, and an object model $m$. The scale $s_{vote}$ and the object location $\mathbf{x}_{vote}$ are defined as

$$s_{vote} \quad = \quad \frac{s_{detect}}{s_{occ}} \tag{2.6}$$

and

$$\mathbf{x}_{vote} \quad = \quad \mathbf{R}\mathbf{x}_{occ}s_{vote} + \mathbf{x}_{detect}. \tag{2.7}$$

In Eq. (2.6) $s_{detect}$ is the scale of the detected interest point in the current image and $s_{occ}$ denotes the scale of the occurrence in the object model. In Eq. (2.7) $\mathbf{x}_{detect}$ is the location of the detected interest point in the current image, $\mathbf{x}_{occ}$ denotes the location of the object center with respect to an occurrence of the model, and $\mathbf{R}$ is a rotation matrix. The matrix describes the rotation from the model to the image orientation.

Figure 2.6(a) shows a voting for an object model. The star-shaped model of the car is shown in Figure 2.4(b). Three SIFT descriptors are found on the target object, marked as red circles. One SIFT descriptor is located on the upper right corner of the roof. The two other SIFT descriptors are located on the two tires. We assume that the two tires look similar. Therefore, they are matched with the same codebook entry. The corner of the roof is assigned to a different codebook entry. Given an object model $m$, the codebook entry of the roof will vote for one object center $\mathbf{x}_{vote}$ and object scale $s_{vote}$. For each tire the generalized Hough transform obtains two votes; one on the left side of the tire and one on the right side. Votes are accumulated in the *Hough accumulator array.* Entries in the Hough accumulator array with many votes have a high correlation to the original object model.

The generalized Hough transform has high computational and memory

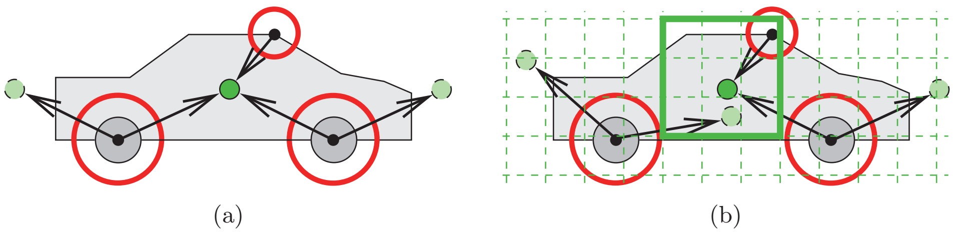


Figure 2.6: (a) Three SIFT descriptors are found (red circles) and matched to two codebook entries: one for the right corner of the roof and one for the two tires. The detected parts votes for three possible object centers (green circles). The object center has three votes. (b) SIFT descriptor of the left tire has slightly orientation offset error. Therefore, one vote is stored in a bin below the object center. Bin locations are drawn as dashed rectangles. The mean shift refinement uses the connected neighbors (green box) to find the correct object center.

requirements. Therefore, in the implementation continuous values of $\mathbf{x}_{vote}$ and $s_{vote}$ are discretized to bins. For example, values between 0.875 – 1.125 vote for an object scale of 1.0. Similar rounding applies for values of $\mathbf{x}_{vote}$. Coordinates between $(\lfloor x_i \rfloor - 0.5, \lfloor x_i \rfloor + 0.5]$ vote for $\lfloor x_i \rfloor$. The discretization reduces the computational efforts and memory requirements. However, it also introduces boundary effects.

Figure 2.6(b) shows a boundary effect for 2D-coordinates. Green dashed rectangles mark the bins. One SIFT descriptor—the one for the left tire—has an orientation offset error. Therefore, the vote is stored in a bin below the correct object center.

The mean-shift refinement addresses the boundary effects in a post-processing step. It examines the connected neighbor bins. The green box in Figure 2.6(b) surrounds the immediate neighbor bins. If neighbor bins contain votes which supports the initial object center then a refined object center is calculated by weighting each vote to the initial object center and re-estimate a new mean object center. These steps are repeated until a stable object center is found. Figure 2.6(b) is a schematic representation and shows only a 2D-neighborhood. The implementation of the mean-shift refinement operates in a 3D-space, using the 2D-coordinates $\mathbf{x}_{vote}$ and the scale $s_{vote}$.

Codebook entries which vote for an object center, scale, and model are the result of the mean-shift refinement. The detector uses the positions of the voting codebook entries to estimate the rotation, scaling, shear, and translation in respect to the original learned object model. The goal is to project the object boundary of the model to the current image. An affine transformation [46] is assumed; therefore, the detector uses the codebook correspondences to estimate the affine homography $\mathbf{H}_{aff}$. The Eq. (2.8) defines the affine transformation for homogeneous coordinate vectors $p$ and $p'$

$$p' = \mathbf{H}_{aff} \cdot p = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} \tag{2.8}$$

The affine homography is a $3 \times 3$ project matrix where the last rows equals to $(0, 0, 1)$. A least median of squares linear fitting method [99, 47] estimates the six parameters[1] of the matrix $\mathbf{H}_{aff}$. The original boundary is transformed with the estimated $\mathbf{H}_{aff}$, marking the object boundary in the current image.

## Model Updates

Additionally, our high-level reasoning module can update the object model. Local parts may exist inside the projected object boundary which are not matched with the object model. These local parts are possible candidates for a model improvement. If a learning event is triggered by the high-level reasoning module, then the low-level vision component distinguishes between two cases. This is done to avoid the learning of new model parts which does not belong to the object. The first case is a non-moving object. In this case, the low-level vision component adds new parts represented as codebook entries into the model. In the second case the object is moving and the algorithm uses motion information. The low-level vision component estimates the motion of the object $v_{object}$ and the motion for each part $v_{part}$. The motion from a part $v_{part}$ is estimated from the current input image and previous input image. After that, the low-level vision component adds parts with similar motion to the model. A similar motion is defined as

$$\|\mathbf{v}_{part} - \mathbf{v}_{object}\|_2 < \frac{\mathbf{v}_{object}}{2}. \tag{2.9}$$

[1] `http://www.ics.forth.gr/~lourakis/homest/`: Homest version 1.2 is used. Homest is a C/C++ Library for robust, non-linear homography estimation from Manolis Lourakis.

**Algorithm 2:** Detecting and updating part-based object models.

```
Input: codebook, object models, image $I_t$
Output: object locations, codebook, object models
1  Detect keypoints in image $I_t$
2  Assign keypoints to codebook entries
3  Codebook entries vote for models $m$, scale $s_{vote}$ and position $\mathbf{x}_{vote}$
4  Sort candidate locations ordered by the number of votes
5  foreach candidate $c_i$ do
6      Mean-shift refinement of candidate $c_i$
7      Compute affine homography $\mathbf{H}_{aff}$
8      Project object boundary to image space
9      Compute confidence $\frac{n_{voted}}{n_{detected}}$
10     if learning is triggered by reasoning then
11         Add new parts to model
12         if $\frac{n_{voted}}{n_{detected}} > 0.2$ then
13             Update part statistics $r_{m,i}$
14             Delete unreliable parts from model $m$
15         end
16     end
17 end
18 Update codebook entries with Alg. 1
19 return object locations, codebook, object models
```

With this approach the low-level vision component can add codebook entries of local parts that were not detected in learning images.

We want to keep the number of parts in a model at a moderate size. Therefore, a recognition rate $r_{i,m}$ for each part $p_i$ in the model $m$ is calculated and depending on these recognition rates the low-level vision component decides if parts are deleted from a model. Recognition rates are only updated if an object model is detected reliably.

Only a few SIFT descriptors are needed to detect a model reliable [68] thus we define that an object model is detected reliably if more than 20% of the detected SIFT descriptors inside the projected boundary vote for the object model.

### Reliability of Detection

The ratio of

$$\frac{n_{voted}}{n_{detected}} \tag{2.10}$$

defines the reliability for an object model $m$. The number of SIFT descriptors inside the projected boundary are counted in $n_{detected}$. The value $n_{voted}$ contains the descriptors which voted for the model $m$. This reliably measure fulfills the properties for confidence values. Therefore, it is used in the high-level reasoning module.

The recognition rate for a part in a model $m$ is defined as

$$r_{i,m} = \frac{n_{i,m}}{N_{i,m}} \tag{2.11}$$

where $n_{i,m}$ is how often a part $p_i$ is detected during tracking divided by the number of frames $N_{i,m}$ the object model was found reliably since the part $p_i$ was added. After updating the recognition rates, local parts with a low recognition rate $r_{i,m} < 0.25$ are deleted from models. This threshold was found using a test video where the recognition rates where analyzed.

### Final Remarks

The initial codebook entries and object models are generated from a set of learning images. In every learning image we mark the object boundary of the target objects. In our experiments we used one image for learning the object models and codebook entries. We trust our online learning procedure to build good object models and meaningful codebook entries during tracking. A summary of all steps of the detection is given in Alg. 2.

## 2.3.3 Mean-Shift Tracker

The mean-shift tracker was proposed by Comaniciu et al. in [21]. In mean-shift tracking the object's model is represented by a probability density function (pdf) in the feature space. The feature space in our implementation is the RGB color space. The pdf $q$ represents the model of the object to be tracked by the tracker. A histogram with $m$ bins is used to estimate the pdf $q$ from the first image

$$\hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1\ldots m} \quad \text{with constrain} \quad \sum_{u=1}^{m} \hat{q}_u = 1. \tag{2.12}$$

In subsequent frames pdfs $p(\mathbf{x})$ of candidate regions at locations $\mathbf{x} = (x_1, x_2)$ have to be estimated. Therefore, we calculate $m$ bin histograms at different locations $\mathbf{x}$:

$$\hat{\mathbf{p}}\left(\mathbf{x}\right)=\left\{\hat{p}_u\left(\mathbf{x}\right)\right\}_{u=1\ldots m} \quad \text{with constrain} \quad \sum_{u=1}^{m}\hat{p}_u\left(\mathbf{x}\right)=1. \tag{2.13}$$

In our implementation the histograms are calculated from pixels $\mathbf{x}_i$ located in a $\sqrt{n}\times\sqrt{n}$ squared region. The value of $\sqrt{n}$ depends on the object size. The squared region is centered at the centroid $\mathbf{x}_c$ of the target object. The function $b$ maps the RGB color pixel at position $\mathbf{x}_i$ to the correct bin of the $m$ bin histogram. A convex and monotonic decreasing kernel profile $k(x)$ is used to assign smaller weights to pixels farther from the centroid. A profile of a kernel $K(\mathbf{x})$ is defined as a function $k:[0,\infty)\rightarrow\mathbb{R}$ such that $K(\mathbf{x})=k\left(\|\mathbf{x}\|^2\right)$. For each bin of the histogram, $\hat{q}_u$ is calculated as

$$\hat{q}_u=C_q\sum_{i=1}^{n}k\left(\left\|\frac{\mathbf{x}_c-\mathbf{x}_i}{h}\right\|^2\right)\delta\left[b\left(\mathbf{x}_i\right)-u\right] \tag{2.14}$$

where $\delta\left[n\right]$ is the Kronecker delta function. It returns one at $n=0$, otherwise the function returns the value zero. The bandwidth $h$ defines the number of pixels considered around the centroid $\mathbf{x}_c$. Finally, $C_q$ is the normalization constant to enforce the condition $\sum_{u=1}^{m}\hat{q}_u=1$. Therefore, we have

$$C_q=\frac{1}{\sum_{i=1}^{n}k\left(\left\|\frac{\mathbf{x}_c-\mathbf{x}_i}{h}\right\|^2\right)}. \tag{2.15}$$

The pdf of a candidate region centered at $\mathbf{x}$ is estimated accordingly as

$$\hat{p}_u\left(\mathbf{x}\right)=C_p\sum_{i=1}^{n}k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\delta\left[b\left(\mathbf{x}_i\right)-u\right] \tag{2.16}$$

using the normalization constant $C_p$

$$C_p=\frac{1}{\sum_{i=1}^{n}k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}. \tag{2.17}$$

The distance between the target and candidate distributions is defined as

$$d(\mathbf{x})=\sqrt{1-\rho\left[\hat{\mathbf{p}}\left(\mathbf{x}\right),\hat{\mathbf{q}}\right]} \tag{2.18}$$

where

$$\hat{\rho}(\mathbf{x}) \equiv \rho\left[\hat{\mathbf{p}}(\mathbf{x}), \hat{\mathbf{q}}\right] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{x}) \cdot \hat{q}_u} \tag{2.19}$$

is the Bhattacharyya coefficient between $\hat{\mathbf{p}}(\mathbf{x})$ and $\hat{\mathbf{q}}$. The Bhattacharyya coefficient is a divergence-type measure [55, 66].

The distance between the two distributions in Eq. (2.18) has to be minimized. Finding the best candidate region in an image is equivalent to maximizing the Bhattacharyya coefficient $\hat{\rho}(\mathbf{x})$ in Eq. (2.19). A Taylor expansion of $\hat{\rho}(\mathbf{x})$ is done around the initial location $\mathbf{x}^{(0)}$. Then a gradient-based optimization is used to shift from the initial location towards the location maximizing the Bhattacharyya coefficient. One optimization step from the initial location $\mathbf{x}^{(0)}$ to the next location $\mathbf{x}^{(1)}$ where $\hat{\rho}\left(\mathbf{x}^{(0)}\right) \leq \hat{\rho}\left(\mathbf{x}^{(1)}\right)$ is

$$\mathbf{x}^{(1)} = \frac{\sum_{i=1}^{n} \mathbf{x}_i w_i k'\left(\left\|\frac{\mathbf{x}^{(0)}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} w_i k'\left(\left\|\frac{\mathbf{x}^{(0)}-\mathbf{x}_i}{h}\right\|^2\right)} \tag{2.20}$$

where the weight $w_i$ are

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u\left(\mathbf{x}^{(0)}\right)}} \cdot \delta\left[b\left(\mathbf{x}_i\right) - u\right]. \tag{2.21}$$

The complete derivation of Eq. (2.20)–(2.21) can be found in [19]. Eq. (2.20) is used with the new location $\mathbf{x}^{(1)}$ to calculate the next best location $\mathbf{x}^{(2)}$. Calculations of Eq. (2.20)–(2.21) are repeated until the Euclidean distance between the previous and the new location is less than $\epsilon$.

Additionally, an Epanechnikov kernel [116, page 312] is used; therefore, the derivation of kernel profile $k'(x)$ is constant [19]. The mean-shift step in Eq. (2.20) reduces to a weighted average

$$\mathbf{x}^{(k)} = \frac{\sum_{i=1}^{n} \mathbf{x}_i w_i}{\sum_{i=1}^{n} w_i}. \tag{2.22}$$

In [63], the efficiency of mean-shift tracking is improved by using random subsampling from the $\sqrt{n} \times \sqrt{n}$ squared region. Therefore, in each optimization step the algorithm extracts $n_s$ random subsamples from the region centered at $\mathbf{x}^{(k-1)}$ and then the weighted average Eq. (2.22) becomes

**Algorithm 3:** Mean-shift tracker with random subsampling.

```
Input: model $\hat{\mathbf{q}}$, target candidates $\hat{\mathbf{p}}(\mathbf{x})$ from current image,
       previous location $\mathbf{x}^{(0)}$, number of random samples $n_s$,
       maximum number of mean-shift iterations $it_{max}$
Output: estimated location $\mathbf{x}^{(k)}$ and Bhattacharyya coefficient $\hat{\rho}$
        at $\mathbf{x}^{(k)}$
1  for $k = 1$ to $it_{max}$ do
2      $\mathbf{X} = \texttt{randomSamples}(\mathbf{x}^{(k-1)}, region, n_s)$
3      foreach $\mathbf{x}_i \in \mathbf{X}$ do
4          $w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u\left(\mathbf{x}^{(k-1)}\right)}} \cdot \delta\left[b\left(\mathbf{x}_i\right) - u\right]$
5      end
6      $\mathbf{x}^{(k)} = \frac{\sum_{i=1}^{n_s} \mathbf{x}_i w_i}{\sum_{i=1}^{n_s} w_i}$
7      if $\left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\|_2 < \epsilon$ then break
8  end
9  $\hat{\rho}\left(\mathbf{x}^{(k)}\right) = \sum_{u=1}^{m} \sqrt{\hat{p}_u\left(\mathbf{x}^{(k)}\right) \cdot \hat{q}_u}$
10 return $\left(\mathbf{x}^{(k)}, \hat{\rho}\left(\mathbf{x}^{(k)}\right)\right)$
```

(see Alg. 3, step 6)

$$\mathbf{x}^{(k)} = \frac{\sum_{i=1}^{n_s} \mathbf{x}_i w_i}{\sum_{i=1}^{n_s} w_i}. \tag{2.23}$$

The computational complexity of mean-shift tracking with random subsampling is independent of the object size. The random subsampling is done in step 2 of Alg. 3.

### 2.3.4 Hand Detector

We use a color blob detector for detecting the hands of a human. First, every frame is converted from the RGB color space to the HSV color space [36]. After that, a median filter with a $3 \times 3$ mask is used on the transformed frame to reduce color and luminance noise. A pixel is classified as skin if all channels of the HSV image are between predefined thresholds. The thresholds for the skin color are estimated from a few frames of each video. After classifying the skin pixels in a frame, we have a binary image. The binary image is then processed with morphological operations [102, 103]. Morphological operations add or delete pixels from an image. These are applied to enhance the resulting boundaries of the hands. First, scattered

**Algorithm 4:** A detector for skin color blobs

```
Input: RGB image $I = (I_R, I_G, I_B)$, skin color intervals $(\theta_H, \theta_S, \theta_V)$,
       blob size $\theta_{size}$
Output: Bounding boxes and boundaries of detected blobs
1 $(I_H, I_S, I_V) \leftarrow$ rgb2hsvImage($I$)
2 $(I'_H, I'_S, I'_V) \leftarrow$ (filter$_{3\times3}(I_H)$, filter$_{3\times3}(I_S)$, filter$_{3\times3}(I_V)$)
3 $mask \leftarrow$ apply($\theta_H, I'_H$) $\wedge$ apply($\theta_S, I'_S$) $\wedge$ apply($\theta_V, I'_V$)
4 $se \leftarrow$ createStructureElement('circle', 3)
5 $mask \leftarrow$ close(open($mask$, $se$), $se$)
6 $blobs \leftarrow$ findBlobs($mask$)
7 $bigblobs \leftarrow \{b : b \in blobs \wedge$ sizeOf($b$) $> \theta_{size}\}$
8 return $\{$(boundary($b$), boundingBox($b$)) $: b \in bigblobs\}$
```

and false positive pixels are removed by the morphological operation erode followed by a dilate. This operation is also known as morphological open operation.

Resulting regions can contain holes; therefore, the detector removes these holes with a morphological close operation. The morphological close operation is a dilation followed by an erosion. Every morphologic operation needs a structure element. A structure element defines the neighbor pixels which are taken into account. Thus, it defines the working radius. The used structure element is a circle with radius three. The remaining regions, referred to as blobs, are candidates for hands of a human being. The detector rejects blobs which have fewer pixels than a given threshold. This threshold is adapted depending on the typical size of a hand in a video sequence. After that, we extract the bounding boxes and the boundaries of the blobs. An overview of the blob detector algorithm can be found in Alg. 4.

## 2.4 Components of the Reasoning Module

The high-level reasoning module consist of five components (see Figure 2.7), the vision controller, the hypothesis generator, the hypothesis inspector, the hypothesis visualizer, and the hypothesis graph. The hypothesis graph is the main data structure of the high-level reasoning module.

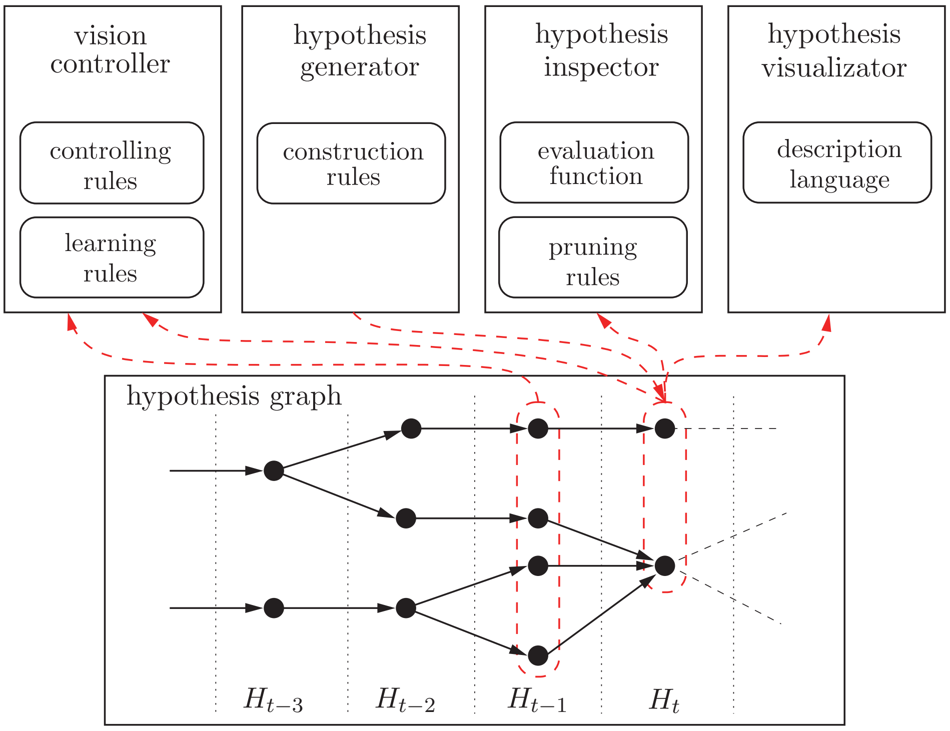


Figure 2.7: Overview of the high-level reasoning module

### 2.4.1 Hypothesis Graph

Every image in a video sequence has at least one interpretation in our system. An interpretation stores the locations of the observed objects. Interpretations are termed *hypotheses*.

If a scene is ambiguous the system generates additional hypotheses for one image. Ambiguous scenes arise by more than one reasonable detection result of a low-level vision component as explained in Sec. 2.2.1. Additionally, a scene can be ambiguous if there is no visual evidence for an object anymore. Even if there is no visual evidence for an object the system stores a location for it. The simplest case is a temporal detection failure of a low-level vision component. If this situation occurs, the reasoning system has to predict the most likely locations of an object. It uses the previous detection results for the prediction.

### Events: Occlusion and Reappearance

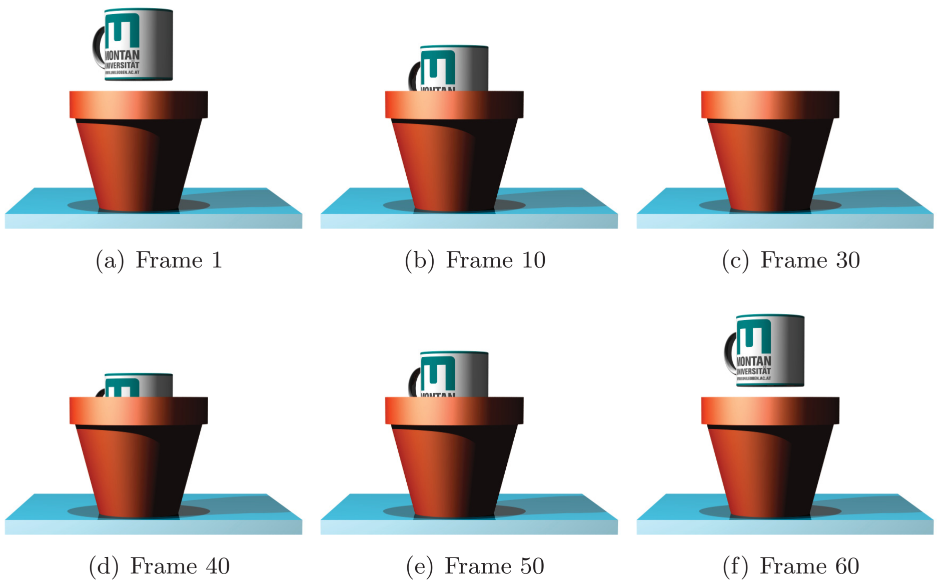
An occlusion event occurs when one or more objects occlude the visual appearance of another object. This is a challenging case because the object may be disappeared over a long time. The goal of our cognitive vision system is that it can give plausible object locations even for occluded objects. Additional, this information is needed to reinitialize the tracker during the reappearance of the object.

Occlusion events will happen if an object is placed inside a larger object or is occluded by one or more other objects. The larger object may be moved to a different location and then the smaller object is taken out. Therefore, objects can reappear at different places after a period of time. In these situations, the system has to take the past hypotheses into account. We use a graph structure to represent these dependencies. Each hypothesis has one or more forward connections to hypotheses for the next frame. Additionally, it has backward connections to its parent hypotheses for the previous frame.

A hypothesis can have more than one parent. This occurs when the visual data are ambiguous and the visual data are not sufficient. In this case, we have two or more *hypothesis paths* which merge into one hypothesis. A hypothesis path is a sequence of consecutive hypotheses in the hypothesis graph. A hypothesis path corresponds to a particular interpretation of the video sequence.

### Example: The Visual Data is Insufficient

Figure 2.8 illustrates an example of a sequence where the visual data is insufficient. A mug disappears behind a pot (Figure 2.8(a)–(c)). The system generates two hypotheses. One hypothesis interprets the visual information as the mug is behind the pot. The other hypothesis assumes that the mug is inside the pot. Then the mug is taken out and reappears (Figure 2.8(d)–(f)). At the end of the sequence, the system has a hypothesis which has two parent hypotheses. The observed event has two possible interpretations and none can be falsified. Figure 2.9 shows the input images on the left side, starting with the first image at the top. On the right side the hypothesis graph is shown with two hypothesis paths. Hypotheses are drawn as perspective views. Figure 2.9 shows that both hypothesis paths are correct interpretations. The visual data are inadequate and the hypothesis paths


(a) Frame 1 (b) Frame 10 (c) Frame 30

(d) Frame 40 (e) Frame 50 (f) Frame 60

Figure 2.8: A mug disappears and reappears on top of a pot.

are merged into one hypothesis.

### Example: The Visual Data is Sufficient

If the sequence of input images is slightly different, the system is able to reject wrong interpretations and stop the processing of unlikely hypothesis paths. Figure 2.10 sketches this situation. The pot moves to a different location before the mug is taken out. The reasoning can reject the wrong hypothesis path. In this case, the information of the remaining hypothesis path is used by the vision controller to find the mug at reasonable locations—nearby the pot[2].

In addition, if we slightly vary the input images then the left hypothesis path is more likely as shown in Figure 2.11. These simple examples show that small variations in the sequence of input images can lead to different interpretations.

[2] Figure 2.10 is simplified because it ignores another possible interpretation of the image data. The third interpretation would be that both objects moved to the left side. Our system takes care of this situation, we omitted it only in our explanation for simplicity.

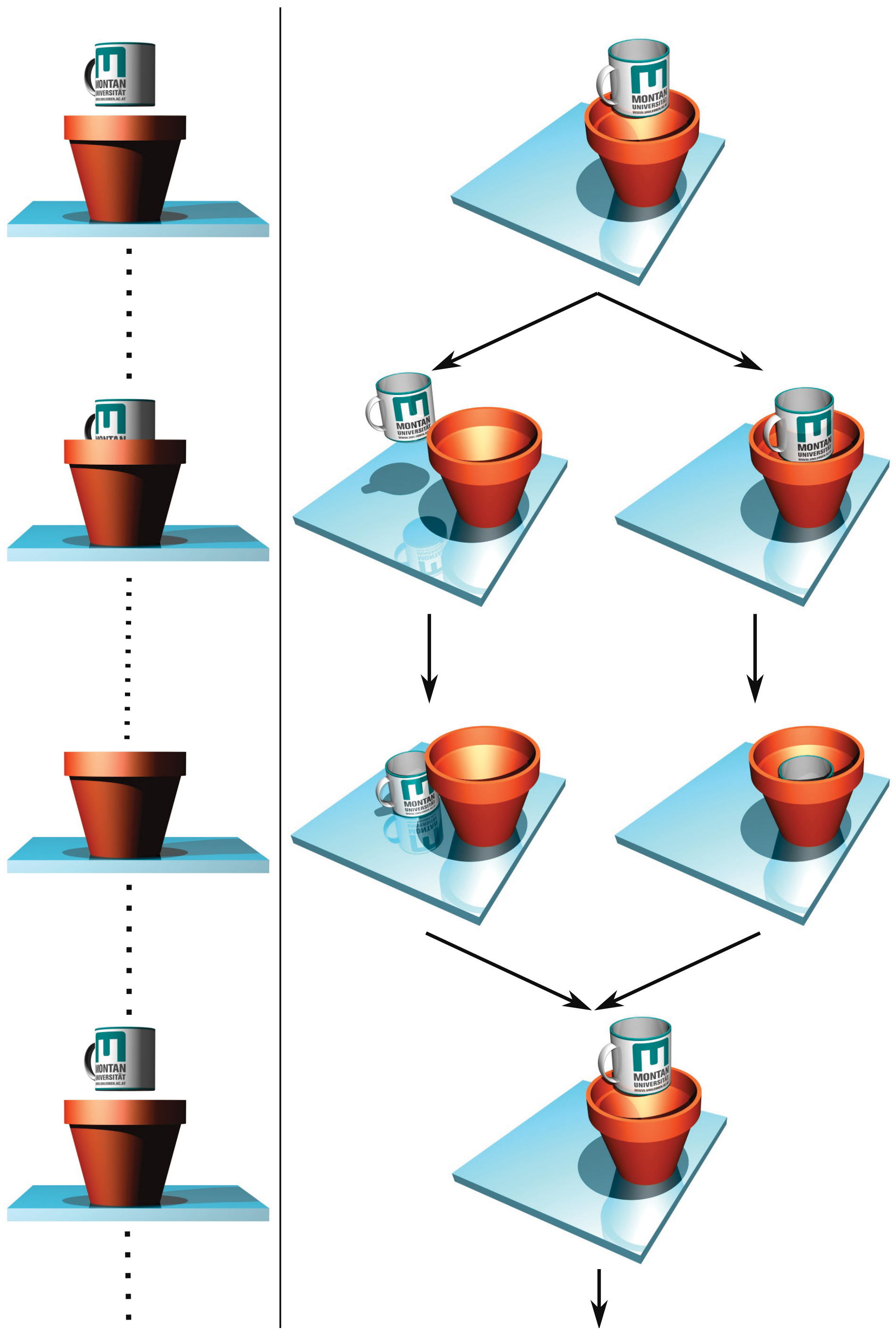


Figure 2.9: A mug disappears and reappears on top of a pot. The input images are shown on the left side from top to bottom, the corresponding part of the hypothesis graph is shown on the right side. The sequence has two possible interpretations and none can be falsified. Therefore, the last hypothesis in the hypothesis graph has two parent hypotheses.

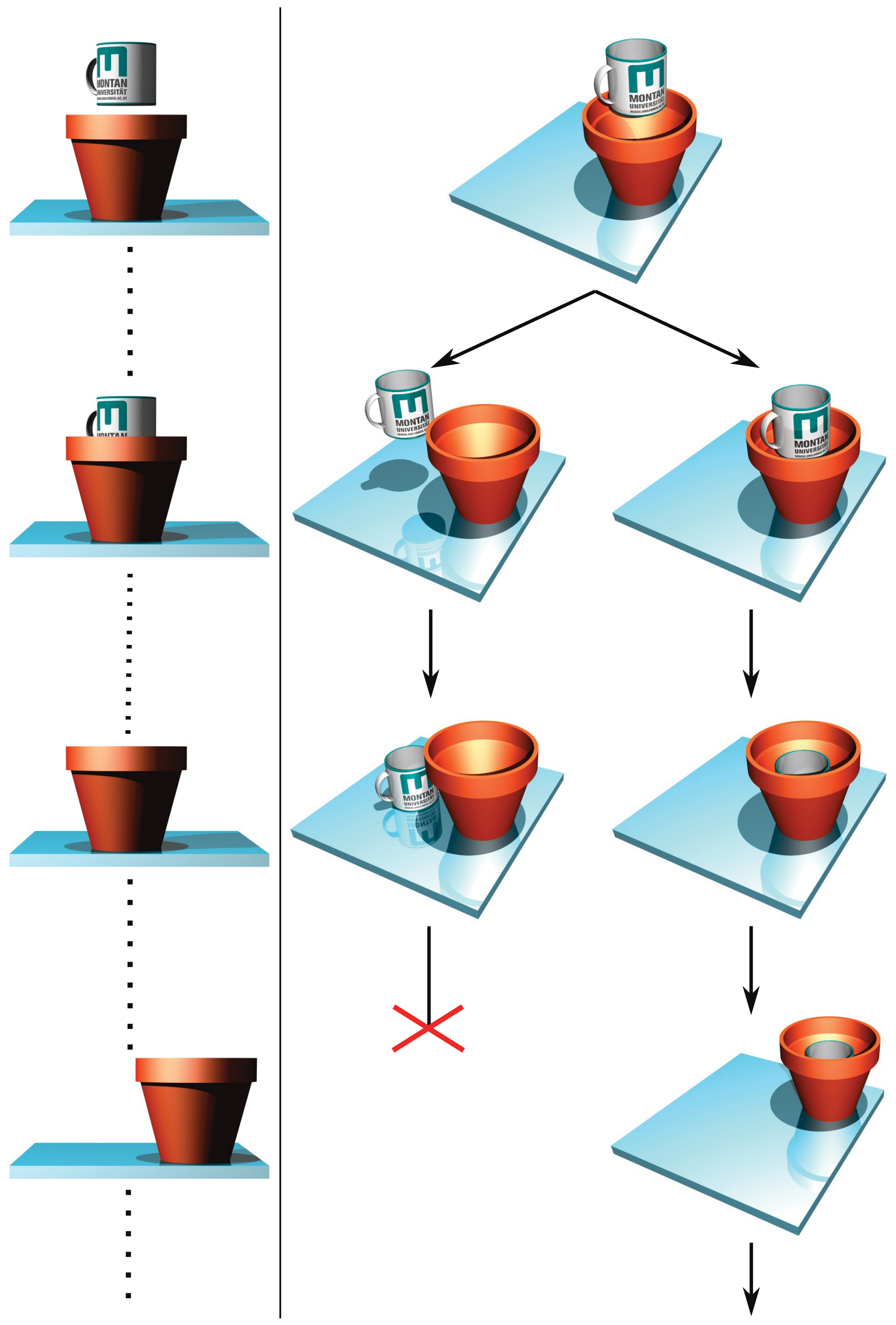


Figure 2.10: A mug disappears and then the pot is moved to a new location. The high-level reasoning has additional data compared to Figure 2.9. In this situation, the right path of the hypothesis graph is more likely than the left path. Further processing of the left path is discontinued by the high-level reasoning (marked as a red cross).

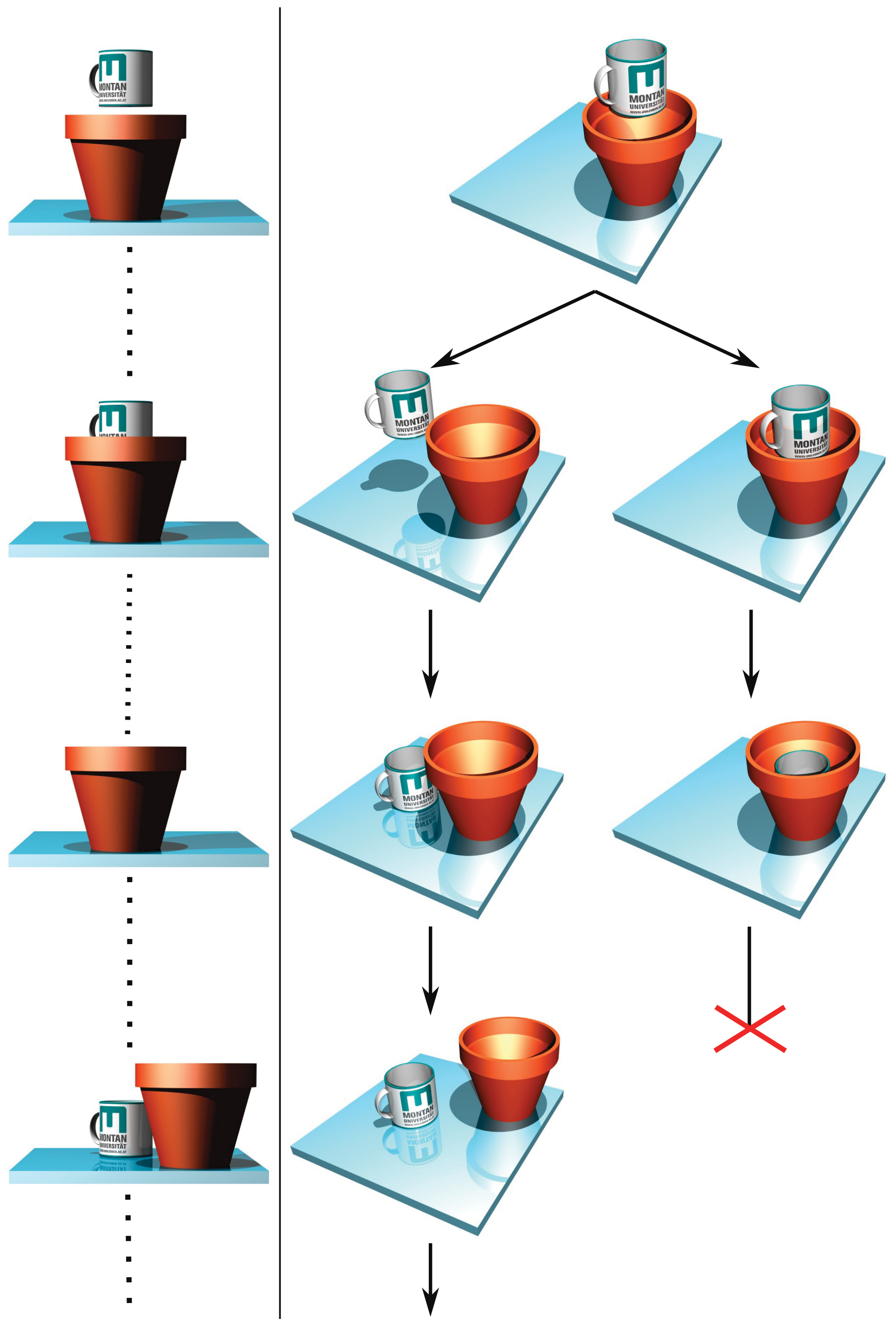


Figure 2.11: A mug disappears. It reappears after the pot is moved to a new location. The high-level reasoning receives different data compared to Figure 2.10. Therefore, the left path of the hypothesis graph is more likely.

### Object States

The above examples also show that hypotheses need to store a state for each object. An object can have the following states: *visible*, *behind*, and *attached*. State *visible* is for visible objects, and the reasoning system can use the results from the low-level vision components. State *behind* indicates that an object is partially or fully occluded by other objects. Depending on the occlusion the reasoning system has to estimate the locations of objects and cannot use the output from the low-level vision component. State *attached* is used if an object is linked to another object. An object which is attached to another object can reappear at locations nearby the attached object. The reasoning system uses the state *attached* if it assumes that two objects move jointly, e.g. when an object is placed or wrapped in another object.

### Object Hierarchies

A simple description with states is not sufficient if the system wants to determine the visible area of objects and more than two objects are observed. The system stores a depth ordering for each hypothesis, called object hierarchies.

Let us assume we have three objects labeled as A, B, and C. If object A partially occludes the two other objects, then there are three possible interpretations:

1. Object C is behind object B and object B is behind object A.
2. Object B is behind object C and object C is behind object A.
3. Object B is behind object A and object C is behind object A.

If hypotheses maintain hierarchies of occluded objects then these possible interpretations can be described. Additionally, these hierarchies give the system the capability to estimate the visual areas of each occluded object in an image. Later, the estimated visual areas are used to reject inconsistent hypotheses and to improve visual appearance models.

Summarized, a hypothesis graph $H$ consists of hypotheses. Each hypothesis describes the content of a particular frame of the video, and it is linked to plausible hypotheses for the previous as well as the next frame. A hypothesis is an assumption about the states of all relevant objects. A hypothesis consists of a set of object descriptions. Visible objects are stored directly in such a hypothesis. Occluded objects, with state behind, are stored within the description of the occluding object. The same applies to the state attach. Attached objects are linked to the description of its parent.

### 2.4.2 Vision Controller

The vision controller makes use of the information from the hypothesis graph to control the low-level vision components. The controller generates different commands depending on the objects' states. Additionally, the controller is used in different working steps of the reasoning component. In the first step, the controller gives a search window and appearance model to the low-level vision component. The rules for generating the commands are given below.

#### Rules for Generation of Search Commands

1. If an object has state *visible* then the low-level vision component obtains its appearance model from the previous hypothesis and a search window. A search window defines the range where the object center $\mathbf{x} = (x_1, x_2)$ is expected in the new image. The system defines a quadratic search window around the previous position of the object[3].

2. If an object is occluded and has state *behind* then the low-level vision component has to process two search commands. First, the controller uses rule 1 to define a search window. Second, the system query explicitly at the last visible position that was known before the occlusion. The vision controller assumes in this case that the object does not move. In both commands, the appearance model before the occlusion event started is used as model.

3. If an object is attached to a parent object then an extended search window is defined. All positions of the parent object are recorded from the current hypothesis to the hypothesis where the object was attached to the parent object. The recorded positions are the trajectory of the parent object. The search window is defined by the maximum and minimum coordinates of the trajectory. Finally, the search window is enlarged in every direction by $v_{max}$ pixels.[4]

[3]It assumes a maximum velocity $v_{max}$ of 50 pixels.

[4]A more elegant solution would be to search only along the trajectory.

The low-level vision components process the search instructions and give the result to the hypothesis generator. The hypothesis generator constructs and adds new hypotheses to the hypothesis graph. After that, the vision controller updates the appearance models. The rule for improving the objects' appearance models is summarized below.

#### Rule for Improving the Object Models

One simple rule is used for improving the object models. This rule states that if an object is visible and reliably detected, then the system updates the appearance model of the low-level vision component. The assumption is that model updates of reliably detected objects are correct.

Although, the system estimates the visible object area of each object, we decided to ignore use this information and do no partial updates of object models, because an incorrectly estimated visible object area and corresponding incorrect model update would have negative effects on the detection performance.

### 2.4.3 Hypothesis Generator

The hypothesis generator constructs new plausible hypotheses and adds them to the hypothesis graph. It uses the information from the previous hypotheses $H_{t-1}$ and the input from the low-level vision components for image $I_t$. The hypothesis generator splits a previous hypothesis into two or more plausible hypotheses, if the correct hypothesis cannot be inferred using the provided data from the low-level vision components. What can be inferred mainly depends on what can be observed. Our setup consists of one static camera which is not calibrated. Thus, the distance of objects from the camera cannot be measured. Additionally, our generic interface to the low-level vision components does not permit us to examine certain pixel regions. This is the drawback of a generic interface as mentioned in Sec. 2.1.2. Therefore, if the contours of two objects intersects, it cannot be decided immediately which object is in front of the other. However, we show that our high-level reasoning can infer the correct depth ordering by using the hypothesis inspector (Sec. 2.4.4). The system uses the following rules for hypothesis construction:

**Construction Rules**

1. If the contours of two objects intersect, then split the previous hypothesis into two. One hypothesis states that the first object is in front, the other hypothesis states that the second object is in front. If the contours of more than two objects intersect simultaneously, then the system calculates every permutation.
2. If an object is occluded by another object (occluder), then the system generates two hypotheses. One hypothesis states that the object does not move (object's state: *behind*) and reappears at the old position. The other hypothesis states that the occluded object can reappear along the trajectory of the occluder (object's state: *attached*).
3. If the low-level vision component returns two or more possible object positions with similar confidence for an object, then generate one hypothesis for each plausible object position.
4. If no occlusion is detected or if the object is found along the trajectory of an occluder, then the system sets the object's state to *visible*.

The system stores one or more hypotheses after processing the construction rules. Each generated hypothesis is evaluated by the hypothesis inspector.

### 2.4.4 Hypothesis Inspector

As shown in the previous sections, there are several hypothesis paths in the hypothesis graph. The number of hypothesis paths depends on the complexity of the observed scene. The most likely hypothesis path—which explains the video sequence—must be determined by the hypothesis inspector. Thus, the hypothesis inspector evaluates each hypothesis path using an evaluation function $Q(\mathcal{H}|\mathcal{I})$. It is a quality measure of a hypothesis path compared to other paths. The hypothesis path $\mathcal{H} = \langle h_1, \ldots, h_m \rangle$ is a selected path from the hypothesis graph. The hypothesis path starts from the root hypothesis $h_1$. The hypothesis path ends at a hypothesis for the current image $I_m$. The corresponding sequence of images is defined as $\mathcal{I} = \langle I_1, \ldots I_m \rangle$. The result of the evaluation function $Q(\mathcal{H}|\mathcal{I})$ is a number between zero and one. The value from $Q(\mathcal{H}|\mathcal{I})$ indicates how well the hypotheses from the hypothesis path fit to the observed images. A value of one is a perfect fit.

### Evaluation Function

We use the mean quality of the hypotheses as evaluation function for a path. The quality measures $q$ for all hypotheses on the path are summed, divided by the number of hypotheses considered,

$$Q(\mathcal{H}|\mathcal{I}) = \frac{1}{m}\sum_{i=1}^{m} q(h_i|I_i). \tag{2.24}$$

We calculate a normalized plausibility value for each object of interest. The normalized plausibility value is a mapping of the confidence value of the low-level vision component and the current state of the hypothesis $h_i$. The quality measure $q(h_i|I_t)$ for a hypothesis $h_i$ of image $I_t$ is then the mean of all normalized plausibility values,

$$q(h_i|I_t) = \frac{1}{N}\sum_{j=1}^{N} p_{norm}(o_j|h_i, I_t), \tag{2.25}$$

where $N$ is the number of tracked objects. The range of the normalized plausibility is from zero to one. A value of one means that the state of the object $o_j$ and the confidence value of the low-level vision component is the most convincing value pair:

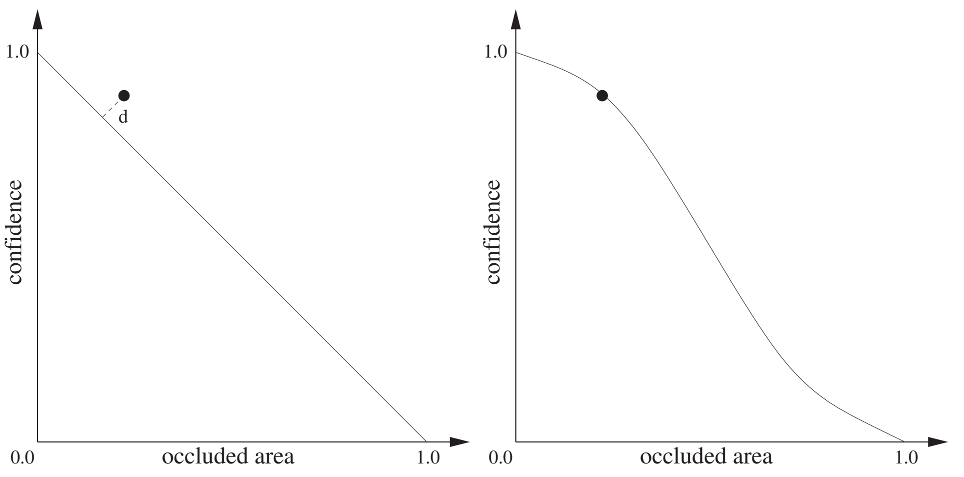
$$p_{norm}(o_j|h_i, I_t) = \frac{p(o_j|h_i, I_t)}{\max_{h \in H_t} p(o_j|h, I_t)}, \tag{2.26}$$

where $H_t$ is the set of all hypotheses for image $I_t$.

The plausibility value uses an inconsistency value $\iota(o_j|h_i, I_t)$ which defines a mapping between the hypothesis state for an object and the confidence value of the low-level vision component. For objects which are not totally occluded we define

$$p(o_j|h_i, I_t) := 1 - \iota(o_j|h_i, I_t). \tag{2.27}$$

The inconsistency function $\iota : \mathbb{R}^2 \to \mathbb{R}$ maps the confidence value of a detector $\mathit{conf}\,(o_j|h_i, I_t)$ and the relative occluded area $area_{occ}\,(o_j|h_i)$ given a hypothesis $h_i$ to an inconsistency value. The inconsistency value is between zero and one, where a value of zero means that the result from the detector and the hypothesis state has zero inconsistency.



(a) linear (b) nonlinear

Figure 2.12: Two possible inconsistency mappings are shown. In the linear mapping the inconsistency value is zero on the line. The nonlinear mapping defines zero inconsistency on the curve. Therefore, the same confidence and occluded area values have different inconsistency values. A confidence and occluded area pair is depicted as a point in both figures. In the linear case $\iota(o_j|h_i, I_t) = d$ and in the nonlinear mapping it is zero.

The inconsistency function can incorporate additional knowledge about low-level vision components. The results of the detector can be measured depending on the occlusion of the target object. Using that measurements the inconsistency value can be calibrated by an operator. Figure 2.12 shows two possible inconsistency mappings. On the curve the inconsistency value $\iota$ is zero. If a value pair of $conf\,(o_j|h_i, I_t)$ and $area_{occ}\,(o_j|h_i)$ does not lie on the curve than $\iota(o_j|h_i, I_t)$ will have a value between $(0.0, 1.0]$.

These mappings can be useful for the following circumstances. First, a low-level vision component can react differently to small occlusions than to large occlusion. If this is the case, it can be taken into account with a special inconsistency mapping for all objects as seen in Figure 2.12(b). Second, if low-level vision components using different visual properties are used for detections. These various low-level vision components may need individual mappings such that same inconsistency values between various low-level vision components have equal meaning. This is a calibration process which can be done before the system is used.

We assume in all experiments a linear behavior between occlusion and detection results $conf\,(o_j|h_i, I_t)$ as in Figure 2.12(a). Therefore, our inconsistency function $\iota$ is defined as

$$\iota(o_j|h_i, I_t) = |conf\,(o_j|h_i, I_t) + area_{occ}\,(o_j|h_i) - 1|\,. \tag{2.28}$$

A different definition of plausibility is needed if an object is totally occluded. It would be misleading if a plausibility value of 1.0 were assigned to totally occluded objects, since a visible object will not typically receive a plausibility value of 1.0 due to imperfect recognition by the low-level vision component. Therefore, we bound the plausibility of a total occluded object by its maximal plausibility when it is visible:

$$p(o|h_i, I_t) = \max_{\substack{h \in H_t \\ o \text{ is visible}|h}} p(o|h, I_t) \tag{2.29}$$

If there does not exists a hypothesis $h \in H_t$ such that object $o$ is visible, then the plausibility value $p(o|h_i, I_t)$ is set to one. This is the case where all hypotheses assume that the object $o$ is totally occluded.

### Pruning Rules

Pruning strategies are implemented as rules and use the results of the evaluation function. The system rejects a hypothesis if one of the following pruning rules match:

1. Reject a hypothesis if an object has the state *visible* and very low confidence over the last few frames. (A visible object must have at least a minimum confidence value and should be reliably detected over a time period, to be valid.)

2. Reject a hypothesis if an object's state is 'attached', the occluded area is very small, and the confidence is low. (If an occluded object reappears, the low-level vision should recognize the object and provide an appropriate confidence value.)

3. If hypotheses describe the same state then the system merges these hypotheses.

```
hypothesis      = "hypothesis", id, "for frame", number
                  "total confidence", decimal number,
                  "parent hypotheses", hyp-list,
                  "(", {visible objects}, ");";
visible objects = "(", "obj", id, "state", "visible",
                   {other objects},
                   reasoning data, vision data, ")";
other objects   = "(", "obj", id, "state", state,
                   {other objects},
                   reasoning data, vision data, ")";
reasoning data  = "conf", decimal number,
                  "visible area", decimal number;
vision data     = "vision component", id,
                  "vision conf", decimal number,
                  "contour pixels", "[", {pixel}, "]";
state           = "behind" | "attached";
hyp-list        = "none" | id, {",", id};
id              = number;
pixel           = "(", number, ",", number, ")";
decimal number  = "0.", number | "1.0";
number          = digit, {digit};
digit           = "0" | "1" | "2" | "3" | "4" |
                  "5" | "6" | "7" | "8" | "9";
```

Figure 2.13: The grammar in Extended Backus-Naur Form notation used by the hypothesis visualizer to export the hypothesis graph.

4. If the number of hypotheses exceeds a threshold then the system rejects hypothesis paths with low $Q(\mathcal{H}|\mathcal{I})$. The $Q(\mathcal{H}|\mathcal{I})$ has to be lower than the arithmetic mean $\overline{Q}(\mathcal{H}|\mathcal{I})$ multiplied by some $\epsilon \in [0, 1)$.

The thresholds for these rules were selected in a very conservative manner, such that it is very unlikely—across a wide range of videos—that a pruning rule deletes a correct hypothesis. Pruning is used only for efficiency to reduce the number of nodes in the hypotheses graph.

### 2.4.5 Hypothesis Visualizer

Each hypothesis of the graph can be visualized on a screen or exported as a description. The hypothesis visualizer exports the generated data using the description language in Figure 2.13. The description language is expressed

```
hypothesis 91 for frame 30
  total confidence 0.905
  parent hypotheses 89
  (
      ( obj 1 state visible
          ( obj 2 state behind
              conf             1.0
              visible area     0.0
              vision component 8
              vision conf      0.0
              contour pixels   [ (53,123) (54,123) ... ]
          )
          conf             0.975
          visible area     1.0
          vision component 7
          vision conf      0.950
          contour pixels   [ (40, 105) (41, 105) ... ]
      )
  );
```

Figure 2.14: The description language applied on the last hypothesis from the left hypothesis path of Figure 2.10.

by an extension of the Backus-Naur Form [8] (EBNF) and it is defined in ISO/IEC 14977:1996 [51]. The important rules of the EBNF notation are: a comma is a concatenation operator, a vertical bar is used for choices of one from many, the construct {`c`} indicates zero or more repetitions of the enclosed construct `c`, and a semicolon terminates a construct.

Additionally, Figure 2.14 shows a description of a hypothesis using the description language. It is the last hypothesis of Figure 2.10 (left hypothesis path, third row). The input image and hypothesis can be found in the third line. Additional line breaks and indentations are used to make the description more readable to the human eye.

The description of the hypothesis gives us the following information: In the first line, the hypothesis number 91 is shown. Every hypothesis is assigned a unique number and the hypothesis was generated for frame 30. The total confidence of the hypothesis path was evaluated as 0.905 by the evaluation function. The hypothesis path contains all hypotheses from the first hypothesis to the $91^{th}$ hypothesis. Therefore, all image data from the first image to the current image are taken into account. The value from the evaluation function is used to decide the most likely hypothesis path. The next line includes the identifier of the parent hypothesis.

After that, the object descriptions of the visible objects are shown between the first round parenthesis and the last round parenthesis. The following descriptions of the visual objects state that only object 1 is visible. Furthermore, object 1 totally occludes object 2. For each object its confidence values, relative size of its visible area, the associated visual component, the confidence of the visual component, and the contour of the object is given. The description of the occluded object is nested into the description of the occluding object. The detection was unsuccessful, thus the visual confidence value is 0.0 and the confidence value was set to be 1.0 by the high-level reasoning module. The contour pixels of object 2 are estimated by the high-level reasoning module, because of the lake of visual evidence.

This page intentionally contains only this sentence.

Chapter 3

# Experimental Evaluation

## 3.1 Test Scenario I

We evaluate the performance of the proposed system with video sequences of different levels of difficulty. Videos are recorded with a static camera observing a scene with a desktop. Cups are placed on the desktop and a human moves these cups. In the first video sequence two cups are moved. As a result of the setup a cup can be partially occluded by the hands of the human and/or by the other cup. Figure 3.1 from (a) to (c) shows three frames from the first video. In Figure 3.1(b) the yellow cup is highly occluded by the hands of the human and the cup with colored stripes. The cup with colored stripes is slightly occluded by one hand. Even in this simple setup some object trackers will fail to track the yellow cup reliably. However, if we find an object tracker which tracks the yellow cup reliably then it will fail on our next video. We increase the difficulty by adding another big cup which can hide other cups. In the second video the human hides the cup with colored stripes inside the white cup. Figure 3.1(d)–(f) shows three frames from the second video. A cup can be inside another cup and can therefore reappear elsewhere. Additionally, sometimes one cup is total hidden as a result of a cup which is in front and due to one or more hands.

### 3.1.1 Used Vision Components

Initially, we tested different trackers on the videos. In our test videos target objects have similar color distributions with some background regions. Therefore, trackers using color histograms (e.g. [20, 84]) fail. Keypoint based

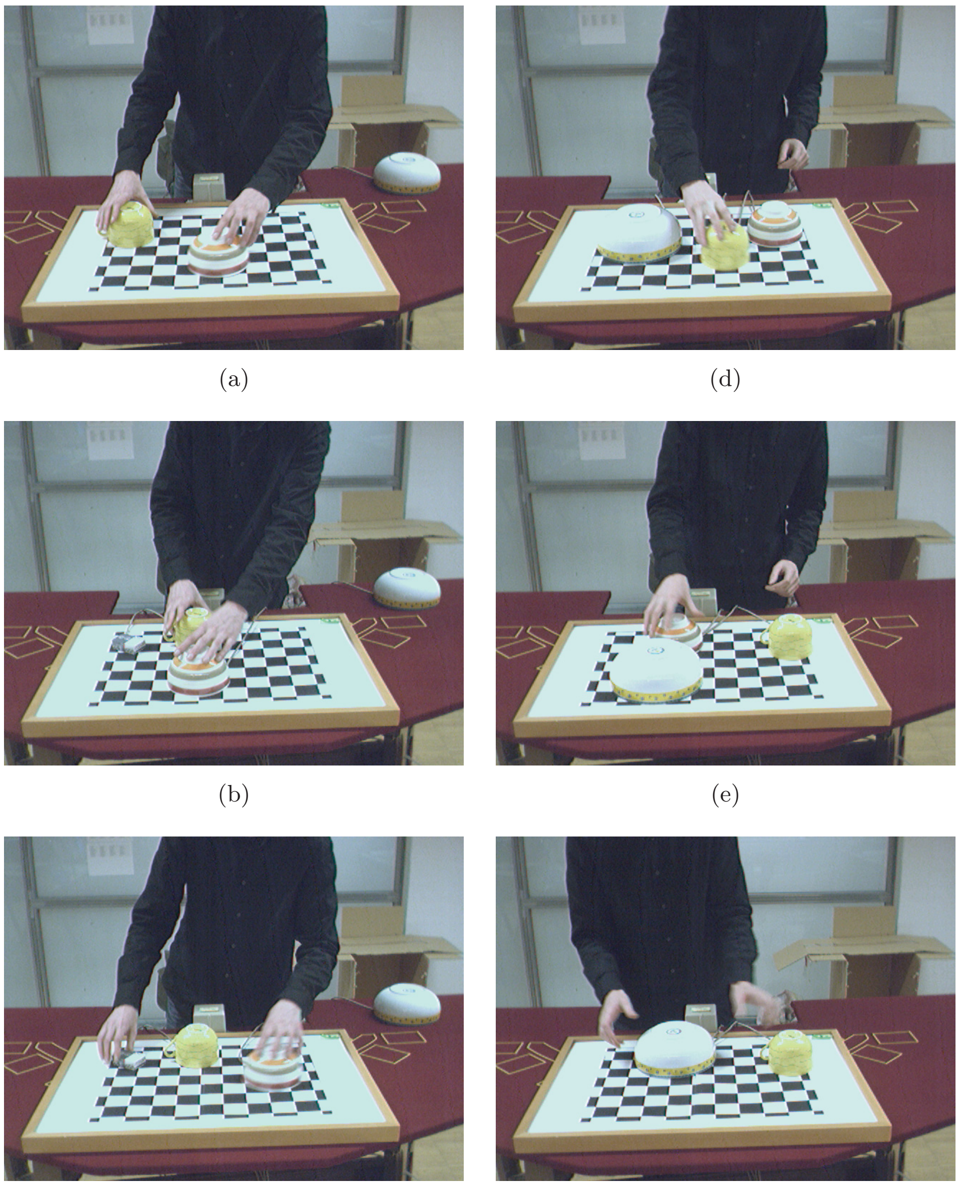


Figure 3.1: Shows three frames of two test videos from test scenario I. Figure 3.1(a)–(c): Video sequence cup 1 with partial occlusions of two cups. Figure 3.1(d)–(f): Video sequence cup 2 where a smaller cup is hidden inside a larger cup.

trackers [70, 62] have problems if they cannot find a suitable number of stable keypoints between two successively frames. The videos have a high signal-to-noise and only a few stable keypoints are found on the target objects. Movements of the target objects introduce a significantly amount of motion blur that removes almost all stable keypoints. Therefore, these types of trackers are not suitable for our videos. On the other hand, template trackers from Sec. 2.3.1 performed well and are used in this set of tests.

We use the OpenCV implementation of the fast matching algorithm from [64]. The algorithm computes the normalized cross-correlation in the frequency domain. The author Lewis in [64] showed that the normalization values can be calculated by using pre-calculated tables (integral images) which yields to a fast matching algorithm.

The Eq. (2.3)–(2.5) on page 17 define the normalized cross-correlation $c^{x,y}$ for a gray scale image. The normalized cross-correlation for each channel (RGB) are averaged and scaled to give a value in $[0, 1]$. Therefore, the input value for the high-level reasoning module is

$$conf\,(o|h_i, I_t) = \frac{1}{6}\Big(c_R^{x,y} + c_G^{x,y} + c_B^{x,y} + 3\Big). \tag{3.1}$$

We use the hand detector from Sec. 2.3.4. The thresholds are adjusted to match the lighting conditions of the scene. Every detection result of the hand detector is represented in our system with confidence value of one.

## 3.1.2 Visual Display of Results

The results of our vision system can be visually inspected, see Figure 3.2 - 3.4. The green/white bounding boxes show the location of the objects according to the most likely hypothesis. The object label is written inside the corresponding bounding box. The line position of the label indicates the relative depth between the objects. If the label is written near the upper bounding box line, then the object is in front of all the other intersecting boxes. The label is bracketed if the occlusion reasoning does not use the low-level vision input. Question marks indicate that the reasoning has assumed a low-level vision failure. The best low-level vision input is drawn with a dashed box, if the reasoning supposes an error.

Figure 3.3 shows four frames from the video. The hypothesis for frame 486 shows that the reasoning component does not trust the low-level vision com-

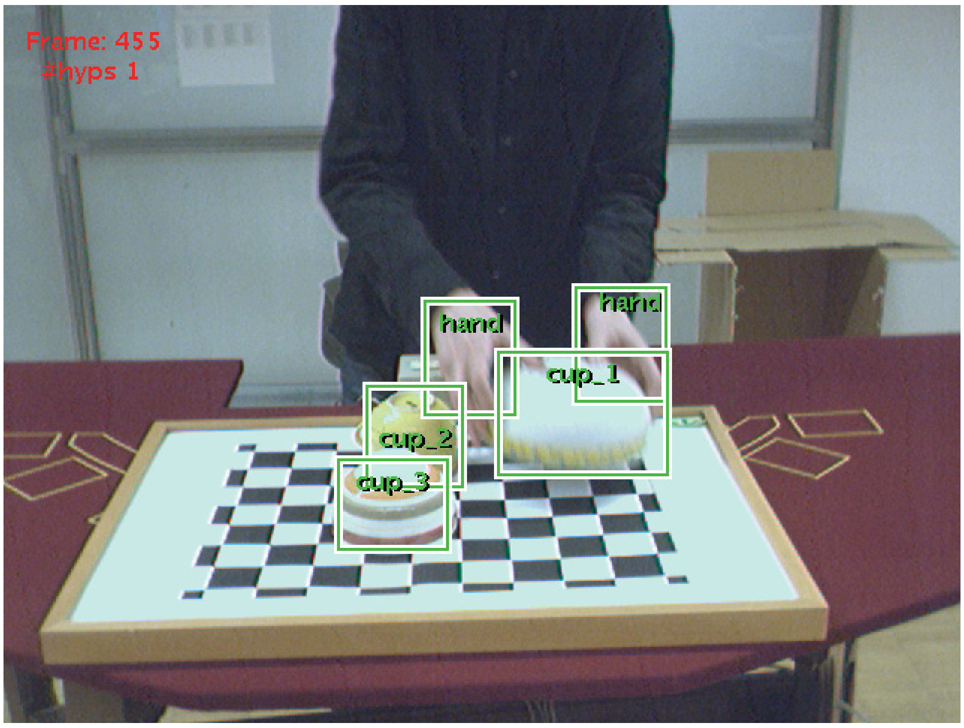
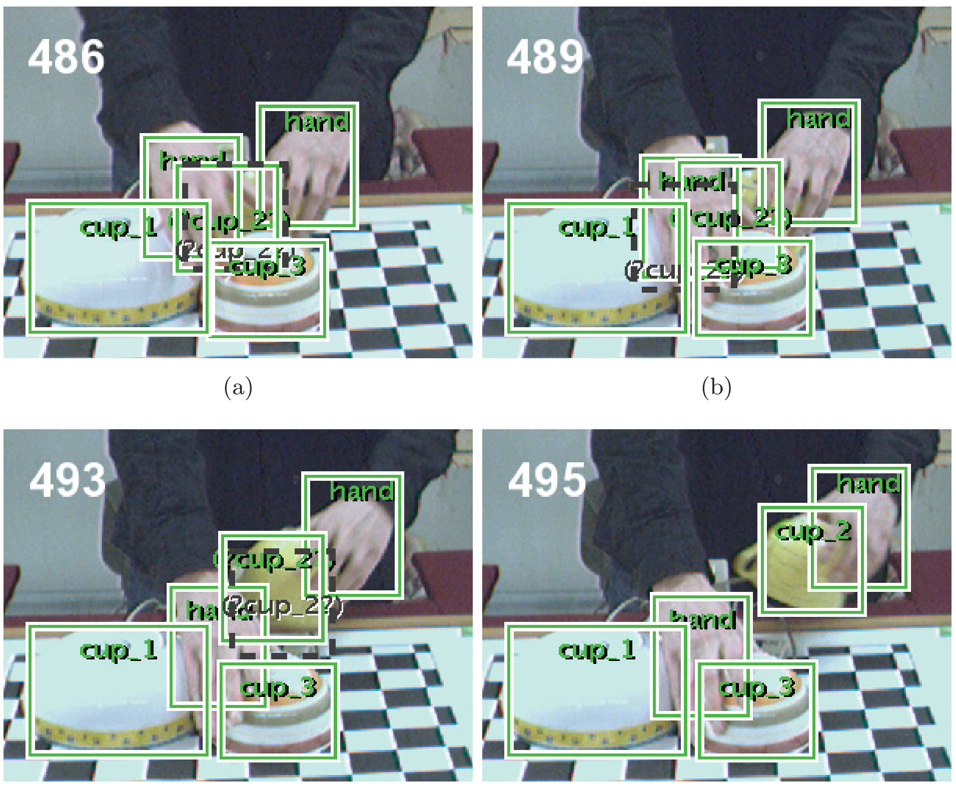


Figure 3.2: A test sequence with three cups. The best hypothesis assumes that 'cup 2' is behind 'cup 3'.

ponent for cup 2, because of the high occlusion. The relative depth between the cups is correct. The next picture (frame 489) shows that the tracker is confused by the hand. The best detection result is on the right hand. The hypothesis assumes that the yellow cup is attached to the left hand, therefore the tracker search window is set along the trajectory of the left hand. In frame 493 the tracker and hypothesis position of cup 2 converge to the same position. Two frames later, the reasoning system believes that the tracker of cup 2 is once again trustworthy, and therefore the system uses the position of the vision component. The video sequence in Figure 3.4 shows a total occlusion and re-detection event. The big cup 1 hides cup 3. After that the two cups are moved together; during that situation the system shows the estimated position of cup 2. The human wants to trick the system and moves the yellow cup in front of cup 1. The hypothesis assumes the following relative depth order: cup 2, cup 1, cup 3. From frame 605 to 620 cup 3 is released, but due to the high occlusion there exists very little visual evidence for the fact that cup 3 is behind the yellow cup. Therefore,



(a) (b)

(c) (d)

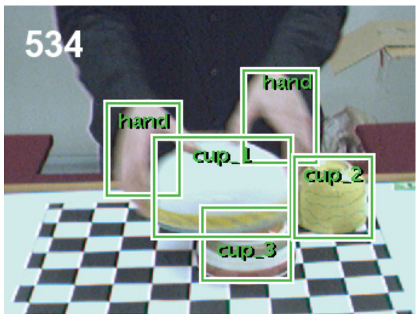
Figure 3.3: Sequence with occluded cup and low-level vision failure.

the best hypothesis assumes that cup 3 is inside cup 1. After the actor moves cup 2 away from cup 3, the low-level vision component provides good confidence values at a previous position of cup 1. The best hypothesis with correct depth values is shown in the last picture.

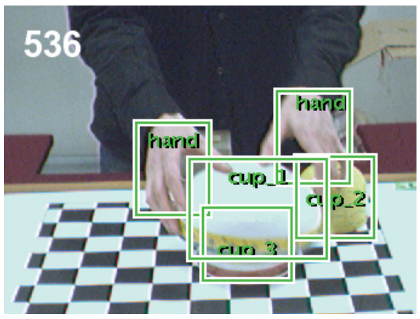
## 3.2 Test Scenario II

We show with the second data set that the system is modular and not tailored to a specific type of trackers. The system can be easily adapted if the constrains of the tracking task are changing. For this we allow that objects have different scale and are rotated during the video sequence.

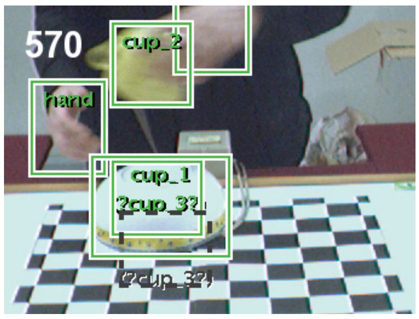
First, experiments with the adaptive template tracker introduced in Sec. 2.3.1 were done. The detection results were, as expected, weak. Our second set of test videos cannot be solved with the adaptive template tracker.



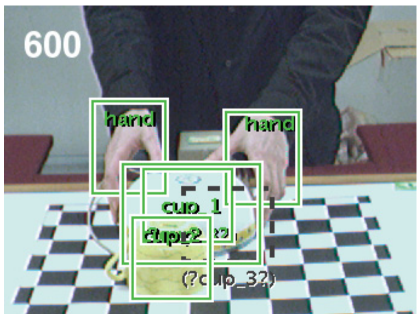
(a)



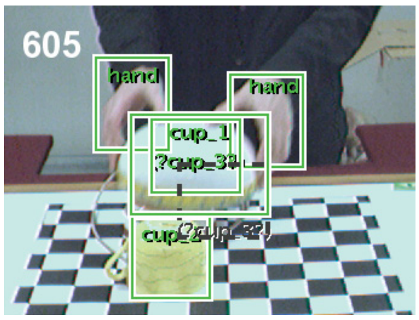
(b)



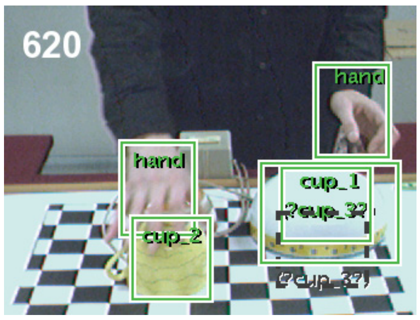
(c)



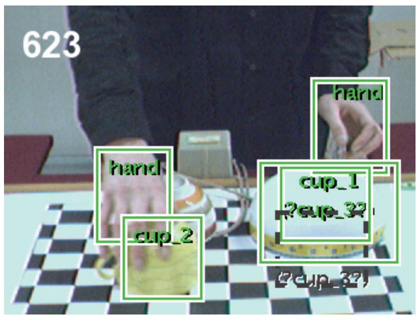
(d)



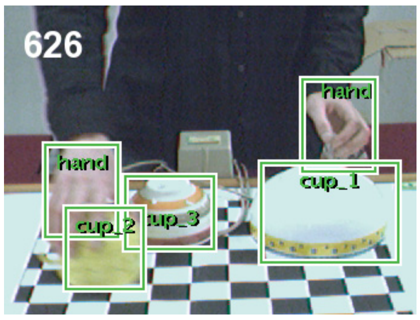
(e)



(f)

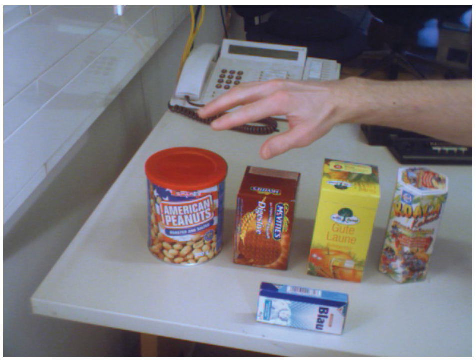


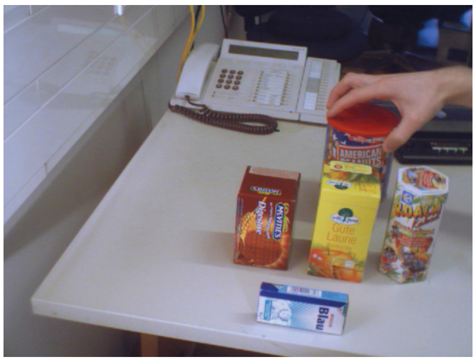
(g)

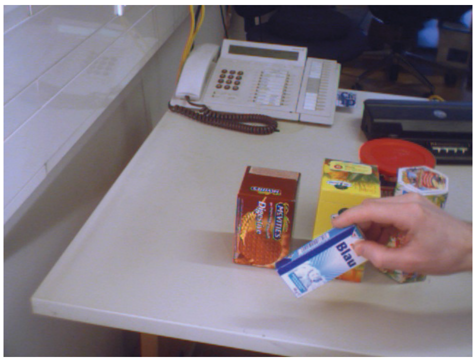


(h)

Figure 3.4: Sequence with an occlusion and re-detection event from video sequence cup 2.


(a) Frame 120


(b) Frame 280


(c) Frame 460

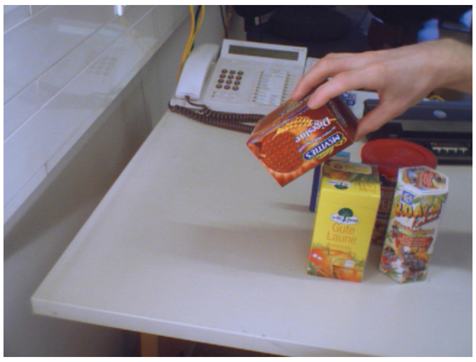

(d) Frame 815

Figure 3.5: Shows the box 1 sequence from the second test scenario. (a) Shows the initial positions of the five target objects from video box 1. (b) One target object is placed behind other target objects. (c) Three target objects are only partially visible. The hand and another target object occlude them. (d) Two objects are hidden and a third one is rotated.

The objects are rotated by 90 degrees and scaled by a factor of 1.35. These appearance changes cannot be handled by the template tracker. Therefore, we use a tracker which stores small local parts of an object, introduced in Sec. 2.3.2. This type of tracker can handle scale changes and rotation.

Figure 3.5 shows four frames from the first video sequence from the second test scenario. The video sequence is called box 1. Figure 3.5(a) shows five target objects placed on a desktop. In the next frames, a person moves, rotates, and hides objects. Figure 3.5(b) shows one target object is moved behind other target objects and it is placed behind two other target objects in Figure 3.5(c). In the third image the hand and one target object hide three target objects. The fourth image shows that two target objects are partially visible and another one is rotated. It can be seen that the occlusion of some target objects is high. The human visual system with its marvelous performance can detect the position of the target objects from the shown frames. A detector, using a model with local parts, fails if only a few parts of the model can be detected. The results shows that the detector combined with the high-level reasoning module is robust against partial occlusion of the target object.

Figure 3.6 shows from the test videos box 2, box 3, and cup 3 two frames. In the box 1 sequence, objects are only hidden behind other objects. The second test video, box 2 sequence, introduces another difficulty: smaller objects reappear at different positions. They are placed into a big box and the big box is moved. After the big box is moved, the smaller objects are taken out of the big box.

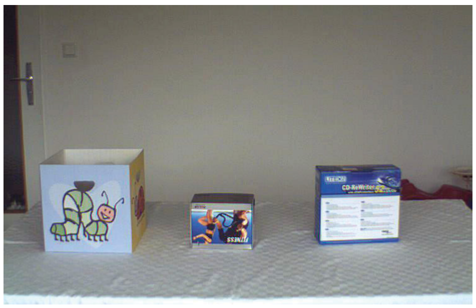
In the box 3 sequence, the human tries to trick the system. The two small boxes are covered by the hands, removing almost all visual evidence, and after that they are moved by the human, e.g. see frame 280 in Figure 3.6(c). However, even this situations can be handled by the system.

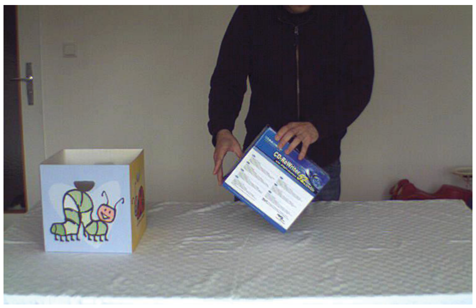
The cup 3 sequence in Figure 3.6(e)–(f) tests the system with extreme visual input data for the part-based tracker. The colored cups have less texture and only a few local parts can be found inside the boundaries. Therefore, detections are unstable and objects are not detected continuously through the video. The overall system can overcome this problems. Additionally, the cup sequence shows the capabilities of the system to cope with hierarchic composition.

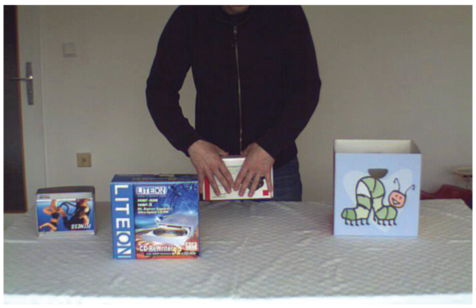
### 3.2.1 Used Vision Components

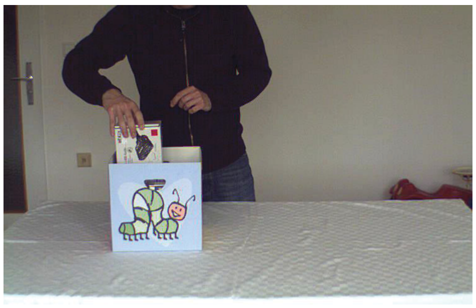
Detection of the objects of interest is done with the detector using a part based model from Sec. 2.3.2. The detector stores the number of parts used in the model. During detection, the detector counts the number of SIFTs which vote for an object model. One definition for a confidence value $conf\ (o|h_i, I_t)$ is the ratio of SIFTs which vote for a model $n_{voted}$ to the number of parts in the model $n_{parts}$. However, this definition assumes that the interest-point detector detects all parts in every frame. In reality, this assumption does not hold; therefore, the above definition includes the systematic error of the interest-point detector.

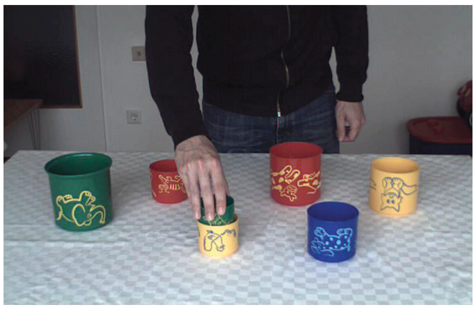
In our experiments, we use a confidence value definition which uses the


(a) box 2: frame 1


(b) box 2: frame 600


(c) box 3: frame 280


(d) box 3: frame 1294


(e) cup 3: frame 350

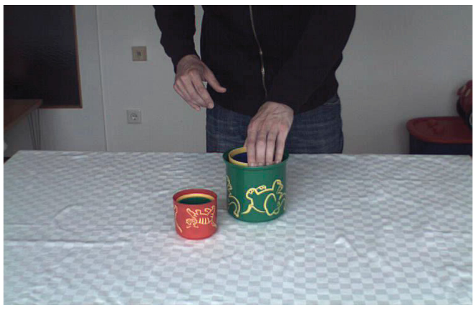

(f) cup 3: frame 1335

Figure 3.6: Figure 3.6(a) - 3.6(b) show two frames from the box 2 sequence. All three objects are moved and the two smaller boxes are placed into the right box. Figure 3.6(c) - 3.6(d) are from the box 3 sequence. In this sequence four objects are used. Additionally, the human tries to trick the system. He covers small objects by hand and moves them. An example of such an action can be seen in Figure 3.6(c). Frames of the cup 3 sequence are shown in Figure 3.6(e) - 3.6(f). Simple textured, colored cups are nested to show the ability of the system to deal with hierarchic composition.

number of detected interest points $n_{detected}$ in the denominator. This reduces the influence from missing parts due to interest-point detector errors. Therefore, the confidence value is defined as

$$conf\left(o|h_i, I_t\right) = \frac{n_{voted}}{n_{detected}}. \tag{3.2}$$

Detection failures of the interest-point detector are not considered. Only detected parts which cannot be matched to the object model have an effect on the confidence $conf\left(o|h_i, I_t\right)$.

The hand detector from Sec. 2.3.4 is applied. The system assigns the detection result of the hand detector a confidence value of 1.0.

### 3.2.2 Results of Model Updates

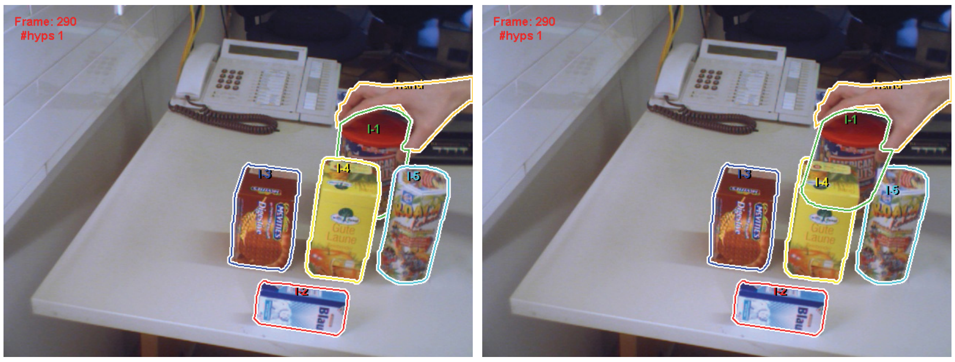
In our first video sequence we arranged the objects within three layers. A person moves the objects between the layers and the objects are partially or totally occluded.

The system has to interpret the sequence correctly, such that it can update the object models accordingly. This is important, because wrong model updates can accumulate to a wrong model, which can lead to false detection results. On the other hand – without updating the object models – the system will not be able to detect the objects reliably. Small objects are usually not reliably detected during motion and rotation. In our first test sequence the object with id 'I-2' is an example of such a small object.

Figure 3.7 shows an occlusion event and two possible interpretation of the detection results. In Figure 3.7(a) the system concludes that the object with id 'I-1' is occluded by the two objects 'I-4' and 'I-5'. It therefore can update the model for 'I-1' accordingly. The system draws the visible boundary for the object 'I-1'.

In Figure 3.7(b) the system tries to explain the scene with a different object ordering. This leads to a different and wrong object model for object 'I-1'. The new learned model for object 'I-1' gives good detection result, because the system learned additional keypoints from the surrounding objects, but the overall hypothesis cannot explain the scene properly. The contradiction is that all detectors give good detection results even under occlusions.

The evaluation function can detect such a contradiction, the correspond-



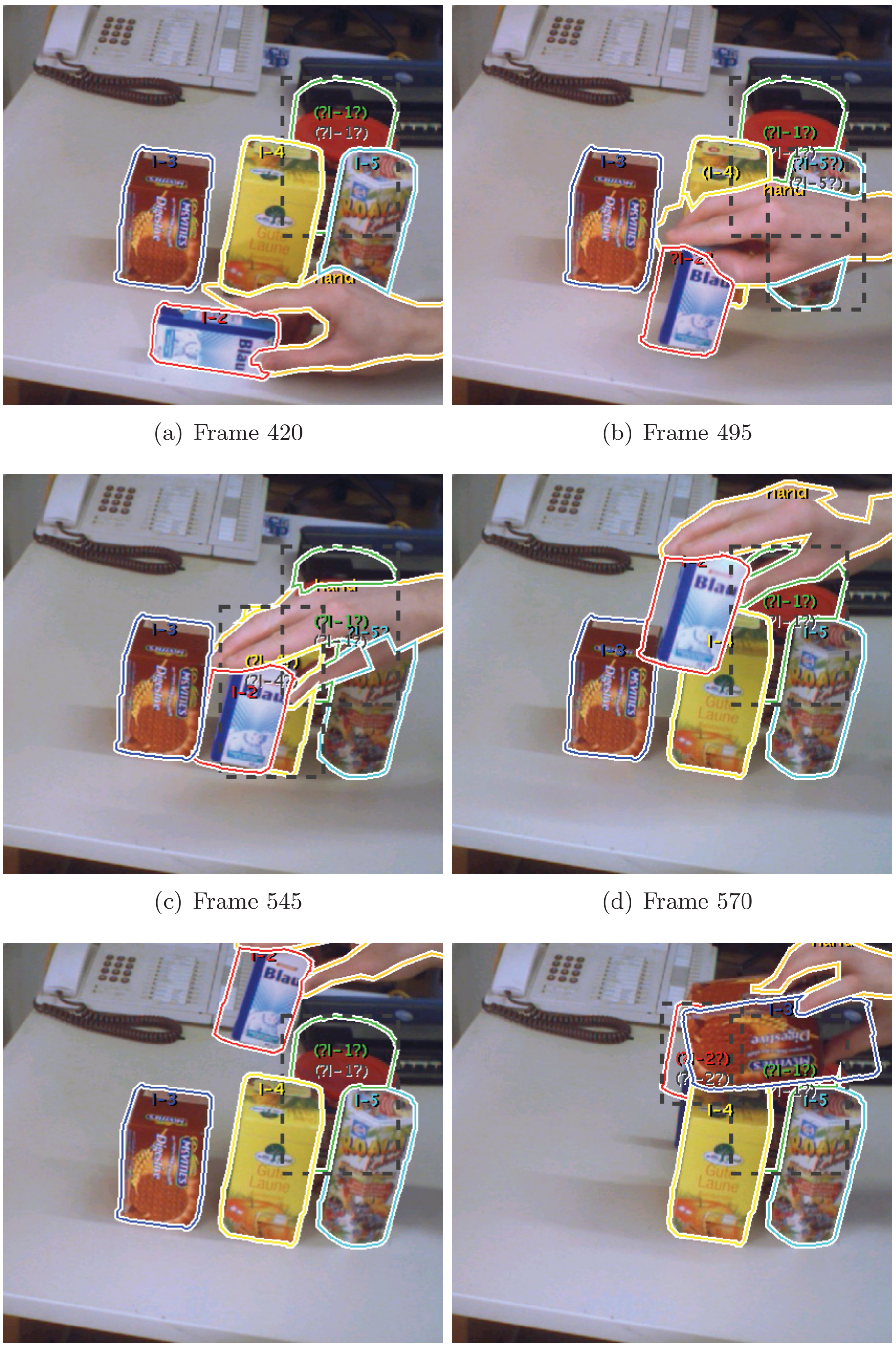
(a) (b)

Figure 3.7: Two possible interpretation of an observed scene are shown. Figure 3.7(a) shows the best interpretation of the frame 290, which is the correct one. The confidence value of this hypothesis path is 0.9669, compared to the other interpretation in Figure 3.7(b), which is only 0.8913. The system explained the occlusion event correctly.

ing hypothesis of Figure 3.7(a) has a confidence value of 0.9669, which is higher than the confidence value 0.8913 of the hypothesis from Figure 3.7(b). Pruning rules may remove the wrong interpretation in later processing steps.

### 3.2.3 Visual Display of Results

We tested our system with four video sequences. In Figure 3.8 a more complex arrangement of objects is shown from the first sequence. Every object has an id label and a boundary. The system draws the estimated visible boundary, therefore the images show the estimated relative depth between objects. The id label, within the boundary, is bracketed if the reasoning does not use the detection result. Question marks indicate that the reasoning has assumed a detection failure. A dashed box around the last known object position is drawn, if the reasoning has not enough evidence for an exact object position. In frame 420 (Figure 3.8(a)) the object with id 'I-1' is placed between the objects 'I-4' and 'I-5'. The relative depth is correctly estimated. The next frame (Figure 3.8(b)) shows nearly total occlusion of object 'I-1', 'I-4', and 'I-5'. In Figure 3.8(c) detection results are used for objects 'I-2' and 'I-3', but not for object 'I-4' which is totally occluded by the hand and object 'I-2'. In the frames 570 and 595 (Figure 3.8(d) - 3.8(e)) there is less occlusion and the reasoning uses the detection results (bottom-

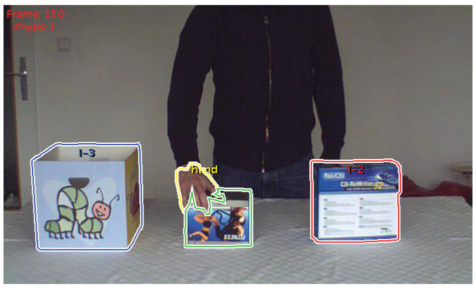


(a) Frame 420 (b) Frame 495

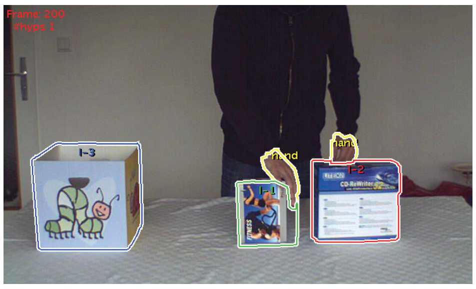
(c) Frame 545 (d) Frame 570

(e) Frame 595 (f) Frame 845

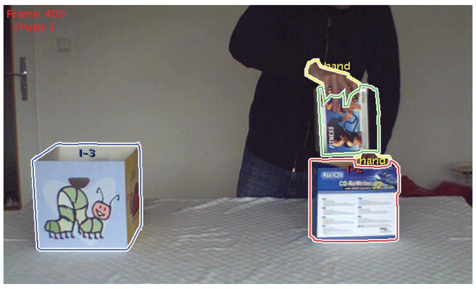
Figure 3.8: The first box sequence showing the various occlusion events. The relative depth between the objects is correct in every frame.



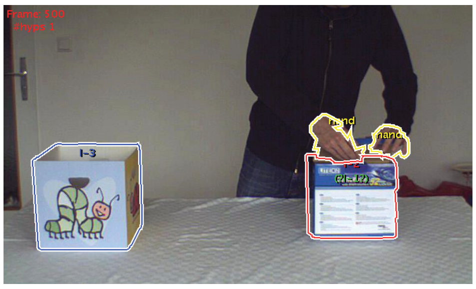
(a) Frame 150



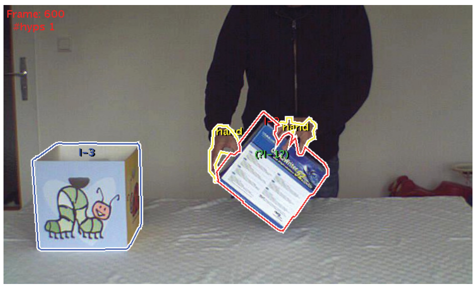
(b) Frame 200



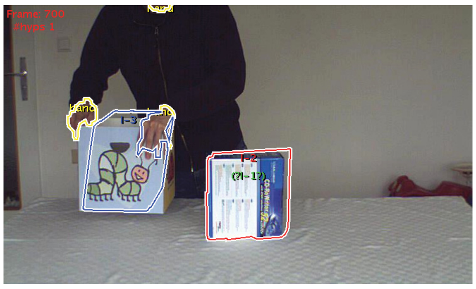
(c) Frame 400



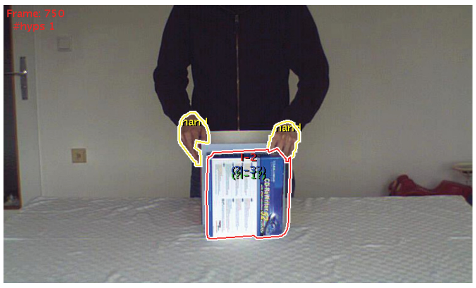
(d) Frame 500



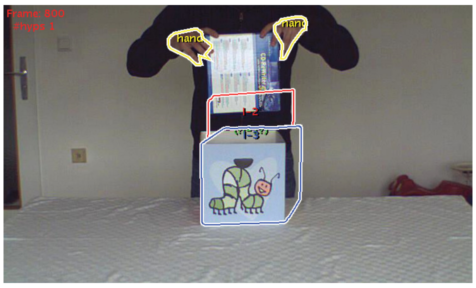
(e) Frame 600



(f) Frame 700

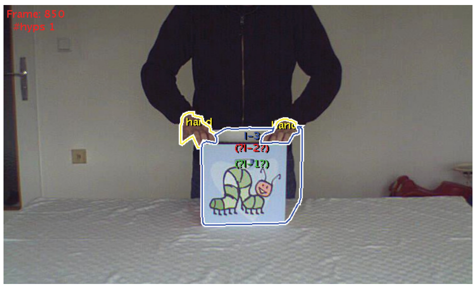


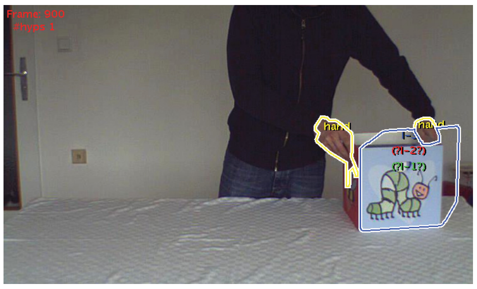
(g) Frame 750

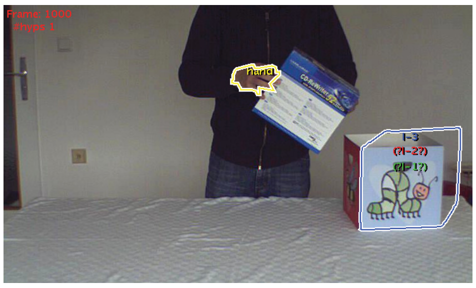


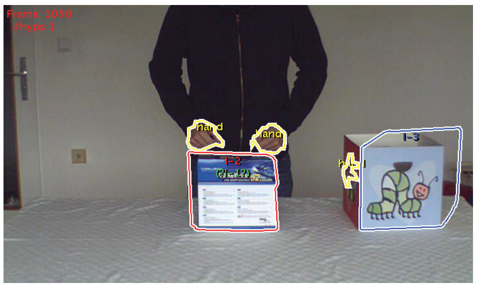
(h) Frame 800

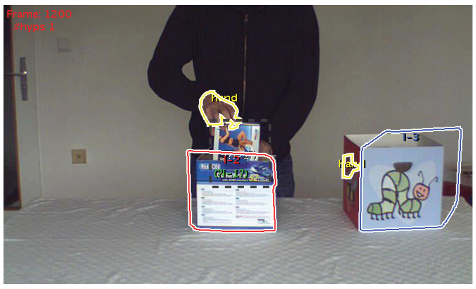
Figure 3.9: Results of the box 2 scene

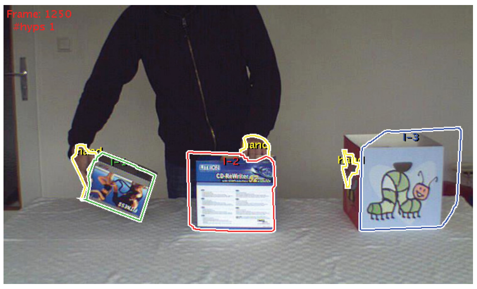

(i) Frame 850

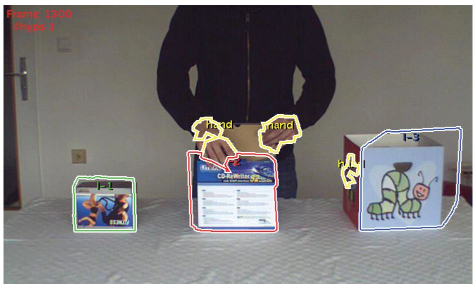

(j) Frame 900


(k) Frame 1000


(l) Frame 1050


(m) Frame 1200


(n) Frame 1250


(o) Frame 1300

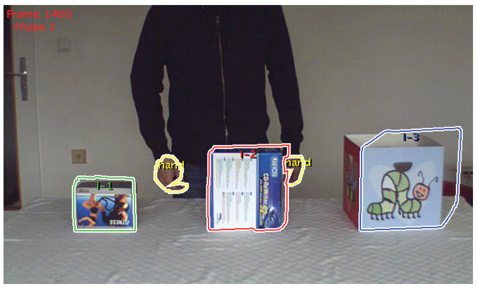
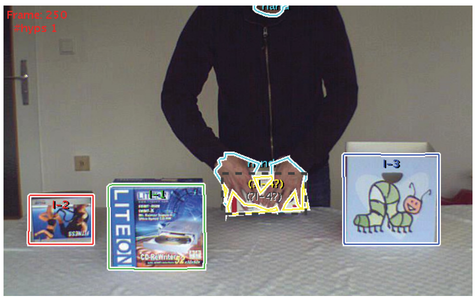
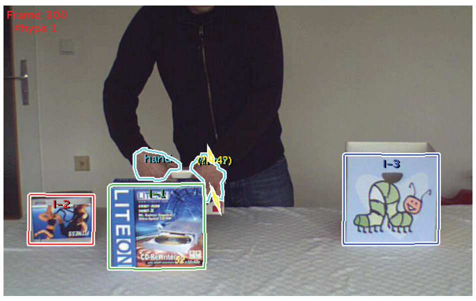

(p) Frame 1400

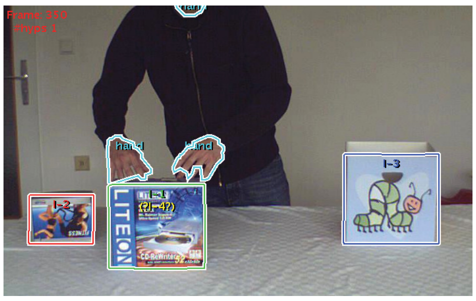
Figure 3.9: Results of the box 2 scene (continued)



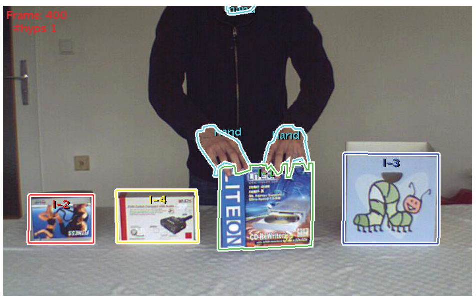
(a) Frame 250



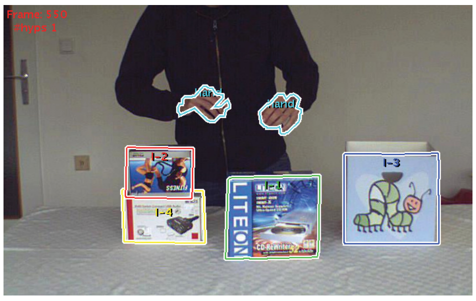
(b) Frame 300



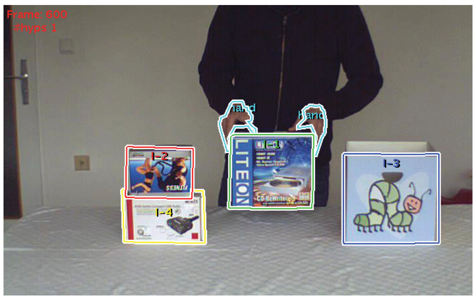
(c) Frame 350



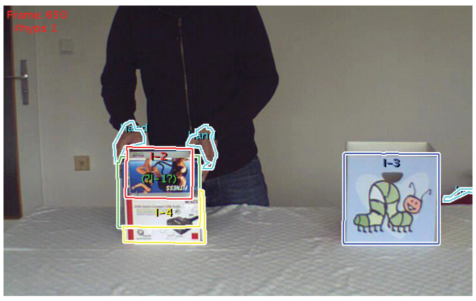
(d) Frame 400



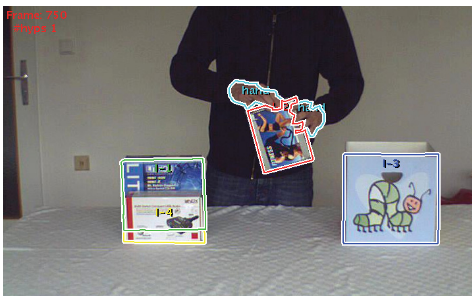
(e) Frame 550



(f) Frame 600

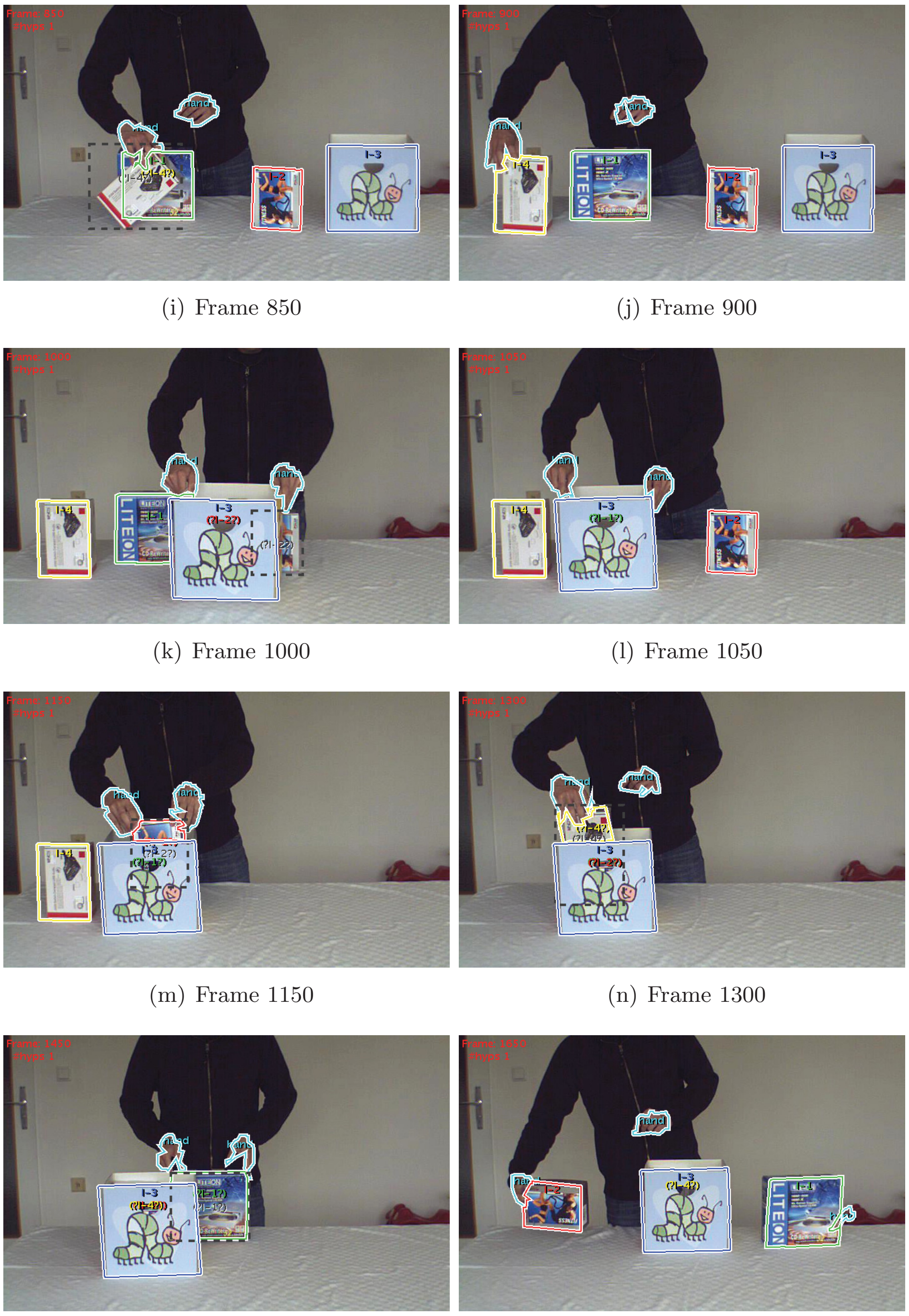


(g) Frame 650



(h) Frame 750

Figure 3.10: Results of the box 3 scene



(i) Frame 850

(j) Frame 900

(k) Frame 1000

(l) Frame 1050

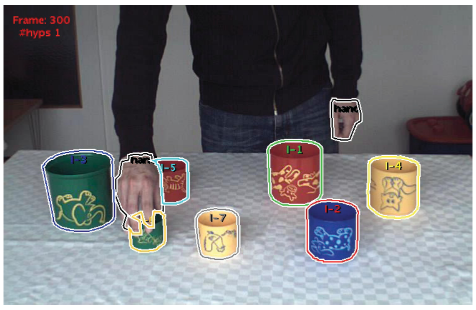
(m) Frame 1150

(n) Frame 1300

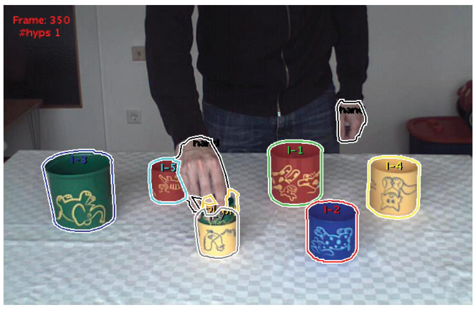
(o) Frame 1450

(p) Frame 1650

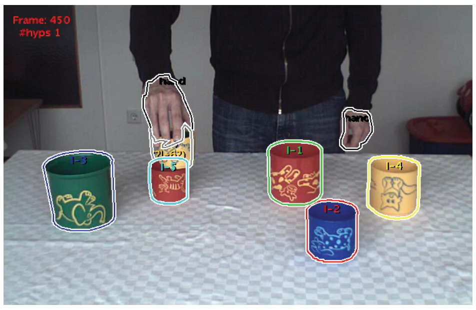
Figure 3.10: Results of the box 3 scene (continued)



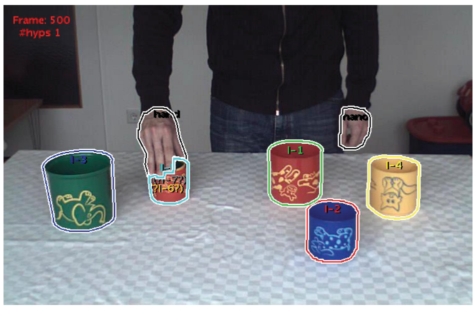
(a) Frame 300



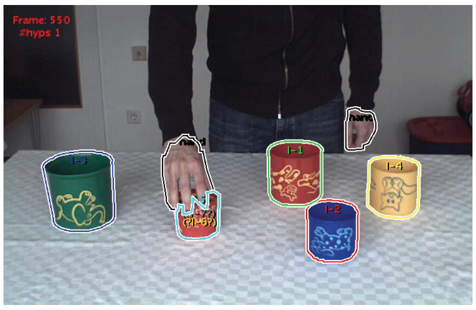
(b) Frame 350



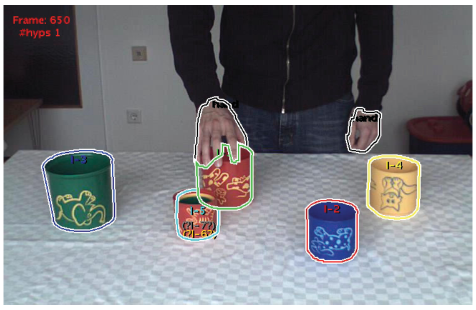
(c) Frame 450



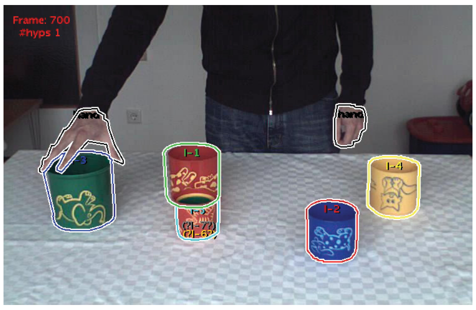
(d) Frame 500



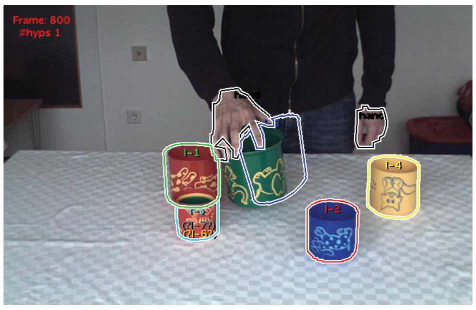
(e) Frame 550



(f) Frame 650

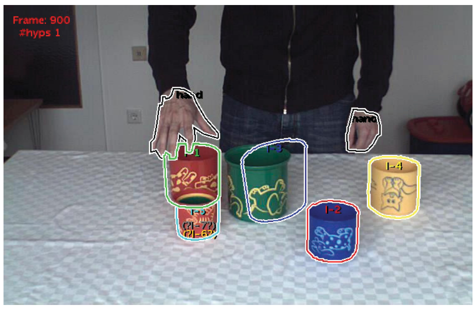


(g) Frame 700

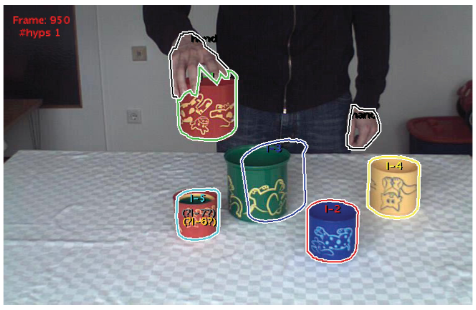


(h) Frame 800

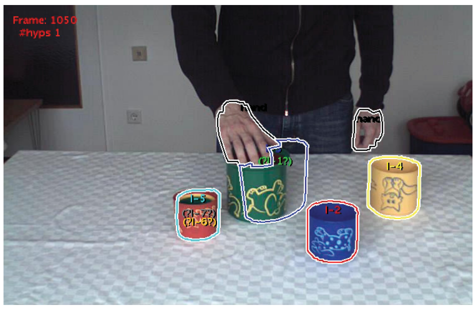
Figure 3.11: Results of the cup 3 scene



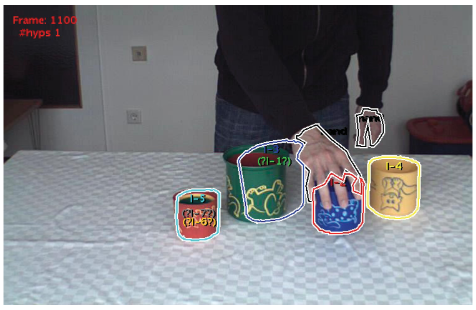
(i) Frame 900



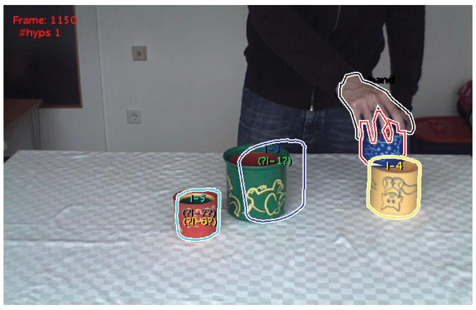
(j) Frame 950



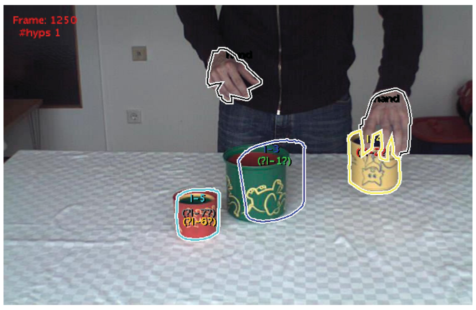
(k) Frame 1050



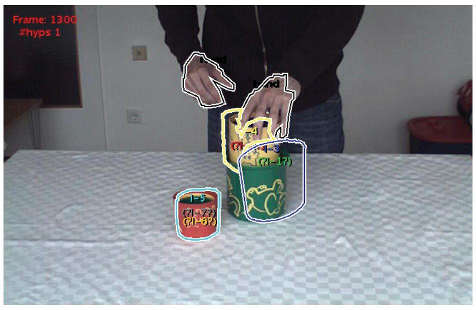
(l) Frame 1100



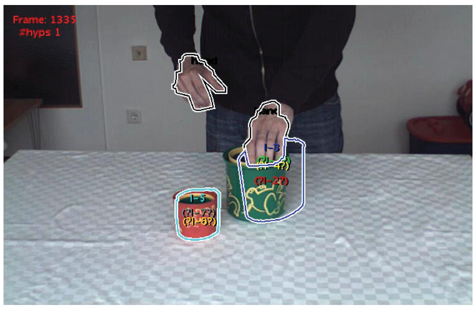
(m) Frame 1150



(n) Frame 1250



(o) Frame 1300



(p) Frame 1335

Figure 3.11: Results of the cup 3 scene (continued)

up processing). In the last frame the object ’I-3’ is rotated and placed on top of the object ’I-4’. The object ’I-1’ and ’I-2’ are totally occluded by the other objects in the scene.

Figure 3.9 - 3.11 show the results of the other test videos. In Figure 3.9 the box 2 sequence is shown. In this video three boxes of different size are moved and rotated. The two smaller boxes are packed into the largest box and the largest box is moved to a different location. After that, the human unpacks it and the system redetects the two smaller boxes again.

A box that occludes or contains other boxes, has the id labels of the hidden boxes below its own id label. The system does not use the detector results of hidden boxes; therefore, the id labels are bracketed. The correct depth ordering is shown in the images, e.g. Figure 3.9(i) shows that the largest box (marked blue) contains the box with the label ’I-2’. And the most likely hypothesis is that the smallest box with label ’I-1’ is packed into box with label ’I-2’.

Figure 3.10 shows the box 3 sequence. The system tracks four boxes and the human tries to trick the system. He covers small objects by hand and moves them; this is shown in Figure 3.10(a) - 3.10(b). The tracking result are reliable and the depth-ordering is almost correct.

Finally, in Figure 3.11 the cup 3 sequence is shown. In this sequence simple textured, colored cups are nested into other cups. The correct depth order is shown with the bracked id labels in the correct order. The objects in this video sequence have only a few stable keypoints on the cups. Therefore, the interest-point detector fails if model updates are disabled, as will be shown in the next section. Figure 3.11 shows the results with learning enabled. The high-level reasoning module provides additional information for the interest-point detector (top-down processing). Hence, the detection results are improved and the correct depth ordering is extracted.

## 3.3 Static Models vs Dynamic Models

In this section we compare the performance of our system with continuous learning enabled or disabled. If continuous learning is disabled then we only use the learned model from the first frame. The model does not change over time and therefore the results are denoted as *static model.* In the other case we use a model that may change over time, through our proposed top-down

| video | static model | | dynamic model | |
|---|---|---|---|---|
| sequence | detection rate | depth order | detection rate | depth order |
| cup 1 | 55.25% | 51.50% | 100.0% | 92.13% |
| cup 2 | 49.67% | 42.94% | 99.78% | 88.05% |

Table 3.1: Performance of the template tracker using static models versus dynamic models.

reasoning. The results are in the columns called *dynamic model*.

There are perhaps two main objections against updating the models during tracking. First, a good tracker may not need model updates, in particular for relatively simple videos. We show that this is not true for our used videos.

The second objection is that model updates during tracking are risky and can lead to wrong models. As a consequence, the performance can be lower than without updates.

The experiments show that it is important to update the models to obtain a good performance. Depending on the difficulty of the video sequence, the difference of the detection rate can be more than 67%. For each tracker we compare the mean detection rate with and without model updates. A detection is counted as correct if the system has found the correct localization and scale of the target objects. The localization is wrong if the projected bounding box is shifted more than five pixels away from the true bounding box. In our experiments we found out that the scale is always correct if the detector find the correct location.

### 3.3.1 Template Tracker

Template trackers with static models work in simple videos. However, our test videos do not fall in this category. Tab. 3.1 shows the performance of the system using a static model versus a dynamic model. The first column is the detection rate for all visible objects. The second column indicates how well the reasoning can infer the correct depth ordering.

The template tracker which uses a static model cannot track the object reliable. It fails if the visual appearance changes because of illumination changes, motion blur, and scale variations. All these effects are present in our videos. The human produces shadow with its hands, moves the cups quickly, and places cups near or far away from the camera. About 50%

| video | static model | | dynamic model | |
|---|---|---|---|---|
| sequence | detection rate | depth order | detection rate | depth order |
| box 1 | 95.90% | 88.86% | 97.81% | 85.05% |
| box 2 | 74.31% | 66.43% | 98.71% | 97.84% |
| box 3 | 81.43% | 74.80% | 99.07% | 91.24% |
| cup 3 | 32.86% | 21.40% | 99.95% | 93.21% |

Table 3.2: Performance of the interest-point based tracker using static models versus dynamic models.

percent of the video the tracker with a static model fails.

Continuous learning using a dynamic model is important for our demanding videos. The gradually illumination change, the motion blur, and small changes in the visual appearance through scale variations are processed by the template tracker with a dynamic model. Good tracking results can be used to even extracted the correct depth order of the objects by the high-level reasoning module most of the time.

### 3.3.2 Interest-point Based Tracker

In Tab. 3.2 contains the results of our system with the interest-point based tracker. As can be seen from Tab. 3.2, our dynamic model from Sec. 2.3.2 improves the detection results, compared to the static model. Additionally, the reasoning component can prevent the system to learn a wrong model even under occlusion events. The cup 3 sequence is only solved with learning enabled (top-down processing). The simple bottom-up processing (static model) gives poor results.

## 3.4 Surveillance Camera

The surveillance camera example in Figure 3.12 was shown in the first chapter as an introductory example. The real-world sequence allows us to evaluate the system performance in realistic, not carefully staged scenarios. The sequence is a car-parking scene with a person entering a car and getting out after crashing the car.

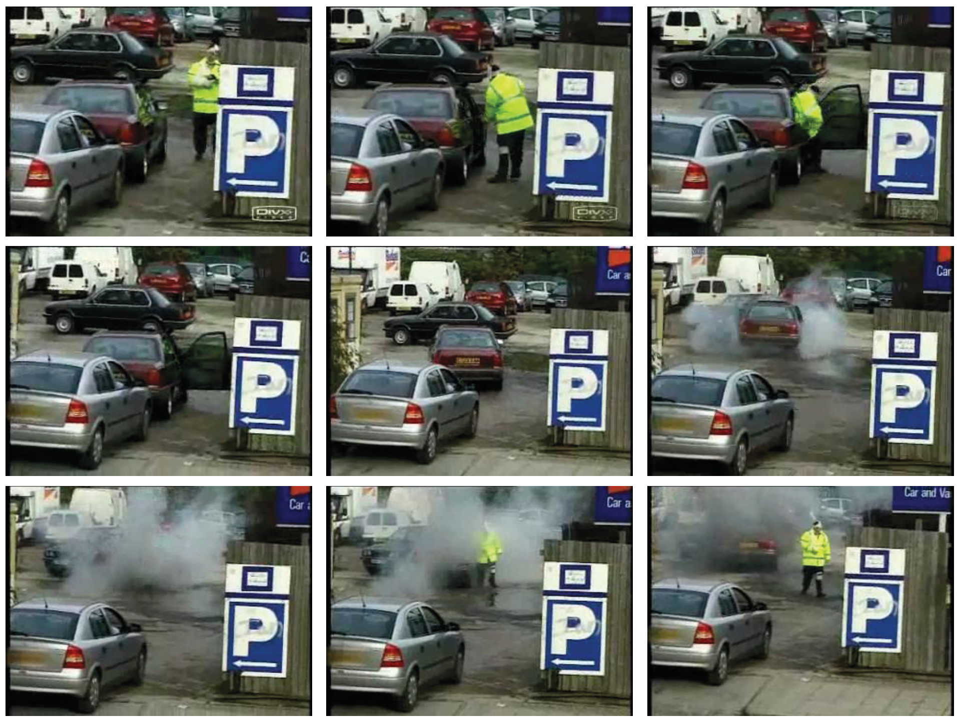


Figure 3.12: Images of a surveillance camera: A person enters a car and crashes this car into another car. In some frames there is no visual evidence of the person and the car.

### 3.4.1 Used Vision Components

The example video is processed with the mean-shift tracker algorithm with random subsambling from Leung and Gong [63] which was introduced in Sec. 2.3.3. We tested other vision components, however the template tracker and interest-point detector failed to reliably detect the objects. The main reason for the poor performance of the template tracker is that it cannot deal with the scale changes of the target objects. The interest-point detector has too few stable keypoints on the target objects which is the reason why it failed. Thus, we use a mean-shift tracking algorithm. The object model of the mean-shift tracker is a color histogram.

The confidence value $conf\ (o|h_i, I_t)$ for the high-level reasoning module is the Bhattacharyya coefficient $\hat{\rho}$. The Bhattacharyya coefficient in Eq. (2.19) on page 27 has a geometric interpretation, because it is the cosine of the angle between the unit vectors $(\sqrt{\hat{q}_1}, \ldots, \sqrt{\hat{q}_m})^T$ and $(\sqrt{\hat{p}_1(\mathbf{x})}, \ldots, \sqrt{\hat{p}_m(\mathbf{x})})^T$. In our case $\hat{\rho}$ will range from 0 to 1, since the values of $\hat{p}_u(\mathbf{x})$ and $\hat{q}_u$ are positive.

The Bhattacharyya coefficient $\hat{\rho}$ is 1 if there is a perfect match between the pdf of a candidate region and the pdf of the object. See [55, 38] for additional properties of the Bhattacharyya coefficient. The Bhattacharyya coefficient $\hat{\rho}(\mathbf{x})$ fulfills our requirements for the confidence value.

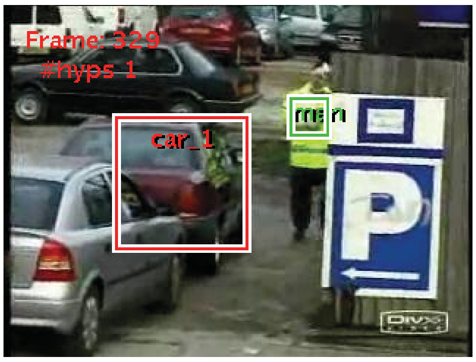
The mean-shift tracker has parameters which have to be set before it can be used. We set the parameters using the first frames such that the tracking is reliable and fast. In our experiment we used 300 random samples in each iteration. The other parameters in the experiment were: $\epsilon$ was set to 1, the number of bins for each RGB channel was set to 8, yielding a feature space of $8 \times 8 \times 8 = 512$, and the number of iteration $it_{max}$ until the optimization stops was set to 25. We observed that the mean-shift optimization usually converged after 4–13 steps and $it_{max}$ was never reached.

### 3.4.2 Results

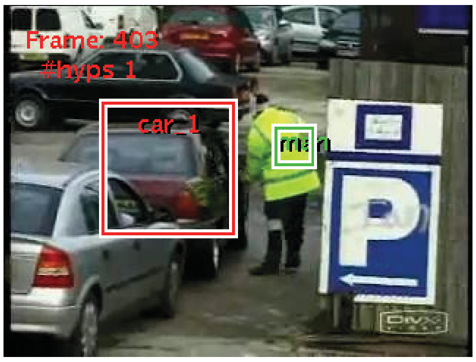
In Figure 3.13 the object of interest are the car and the observed person. They are surrounded by a bounding box with a label. The label, within the boundary, is bracketed if the reasoning does not use a tracker result due to occlusion. Question marks indicate that the reasoning has assumed a tracker failure. A dashed box around the best detection position is drawn, if the reasoning has not enough evidence for an exact object position.

As shown in first row of Figure 3.13, the person opens the car door and enters the car while being partially occluded. From the second row to the third row, the person enters the car completely and closes the door. The person then starts the car which goes into the parking lot. Our reasoning algorithm correctly determines that the person is totally occluded by the car and the person is associated with the car. In the fourth row, the tracked car crashes into another car and smoke begins to come out. After that, the person tries to get out of the car. The tracker is re-initialized by the system when the person is detected reliably.

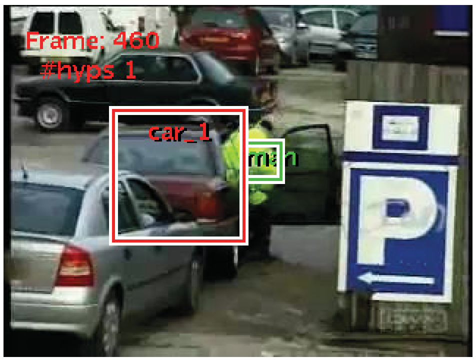
To conclude, our framework addresses the occlusion problem for mean-shift tracking. The problem of incorrect convergence to wrong neighboring local maxima due to occlusion of the target object is solved by the high-level reasoning module. It is shown that the robustness of the mean-shift tracker is improved by considering the states of all relevant objects. Results are demonstrated with a real-world sequence with severe occlusions.



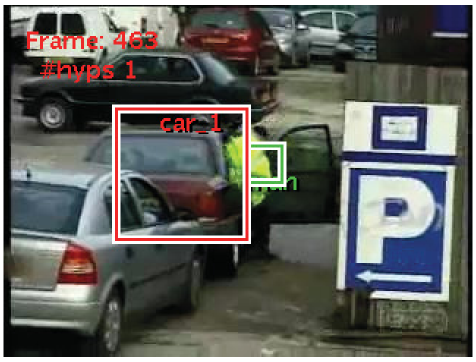
(a) Frame 329



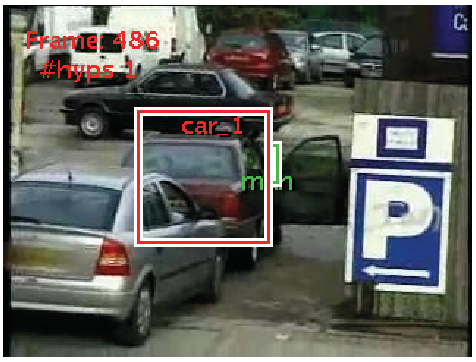
(b) Frame 403



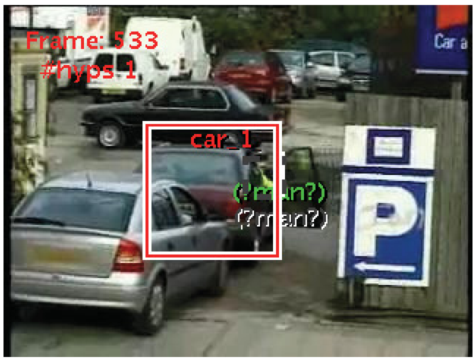
(c) Frame 460



(d) Frame 463

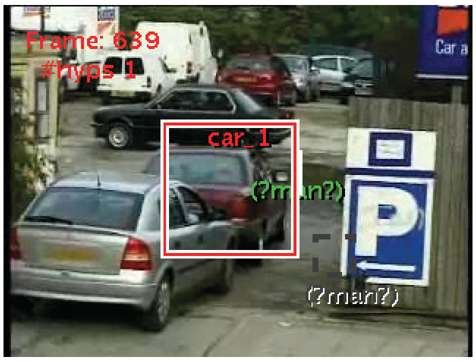


(e) Frame 486

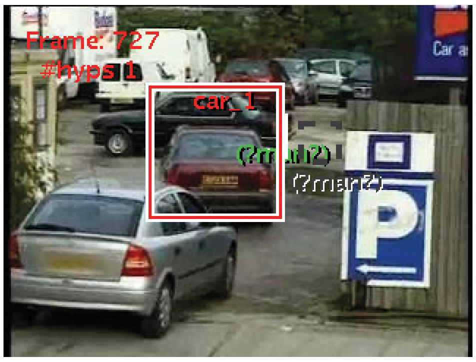


(f) Frame 533

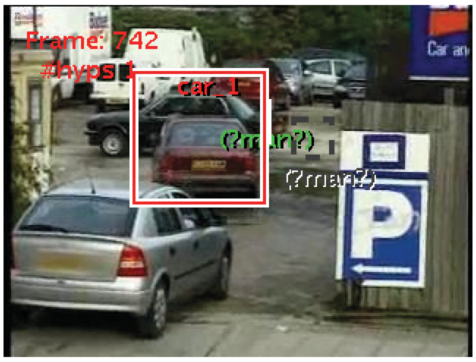
Figure 3.13: Results of the car crash scene. Figure 3.13(a) - 3.13(d) The person goes to the car and opens the car door.



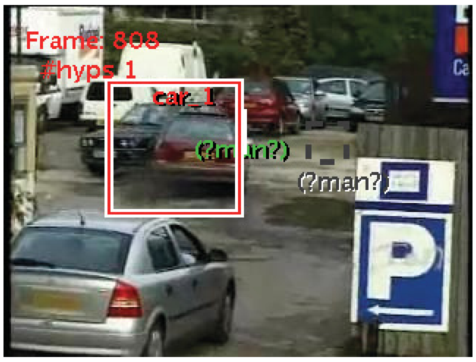
(g) Frame 639



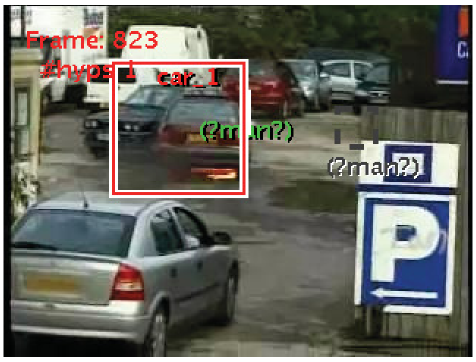
(h) Frame 727



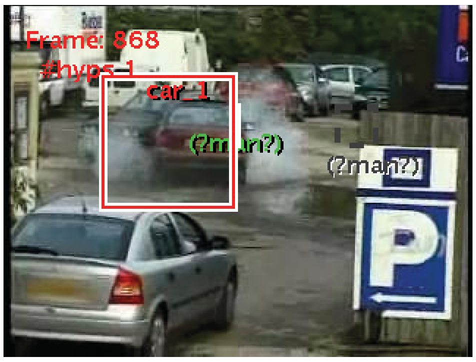
(i) Frame 742



(j) Frame 808

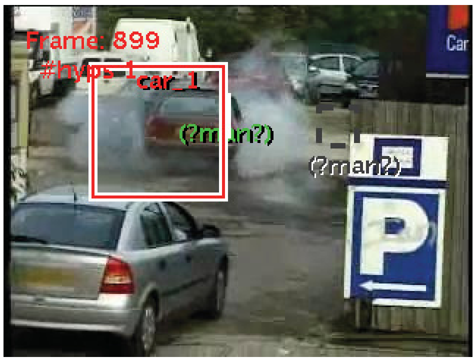


(k) Frame 823

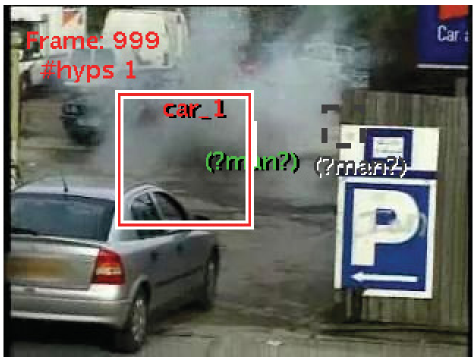


(l) Frame 868

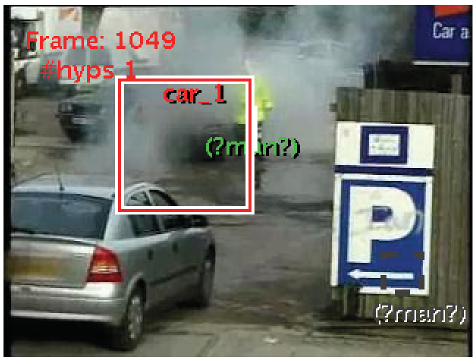
Figure 3.13: Results of the car crash scene (continued).



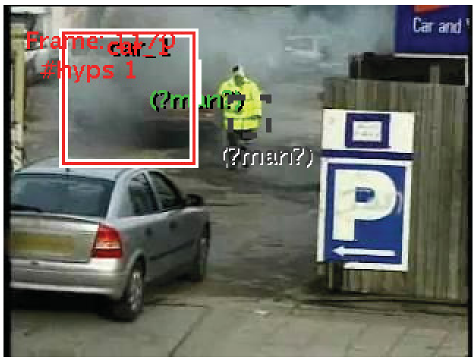
(m) Frame 899



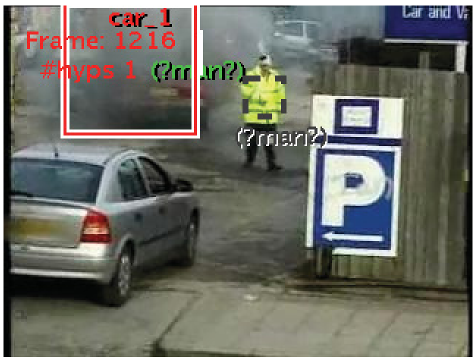
(n) Frame 999



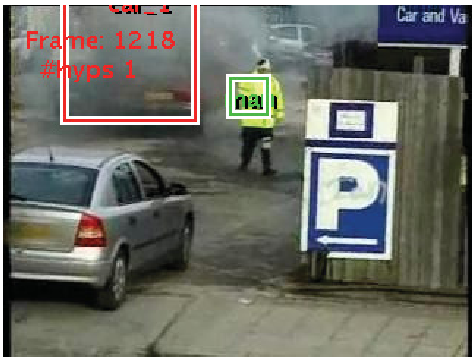
(o) Frame 1049



(p) Frame 1170



(q) Frame 1216



(r) Frame 1218

Figure 3.13: Results of the car crash scene (continued).

## 3.5 Adding Task Specific Knowledge

In the previous sections, we showed that the system is adjustable. In accordance with the requirements of a video sequence, the right low-level vision components can be selected and integrated into the system. In this section, we show that our system can also address shortcomings of the low-level vision components by adding additional task specific knowledge. Due to the modular structure of the system, task specific knowledge can be integrated easily without changing the system. The previously introduced mean-shift tracking algorithm with subsampling is used in our experiments. It is known that the mean-shift tracking algorithm can converged to the wrong local maxima of the Bhattacharyya coefficient if the velocity of the target object is high. There are two possible solution to this problem. First, a camera with higher frame rates can be used. Then the original mean-shift tracker is applicable. However, a higher frame rate means more data to be processed and therefore the processing requires more computing power. In addition, cameras with high frame rates are expensive. The second solution is to add task specific knowledge, in our case we add different motion models. In this case, we do not need specialized and expensive hardware to solve the problem. The next section describes the motion models and the incorporation of the motion models into the existing system.

### 3.5.1 Motion Models

The hypothesis generator will generate additional hypotheses, if the velocity of an object changed too fast in the last frames (motion prediction). The generator chooses from three motion models depending on the previous possible object locations. Currently, the system uses zero velocity, constant velocity, and constant acceleration motion models. For every new generated hypothesis, the reasoning component gives the mean-shift tracker different initial positions.

1. **Velocity and acceleration are zero:** In our first motion model both values are zero. Therefore, the mean-shift tracker is initialized at the old object location. This is usually the default usage of the mean-shift tracker and works quit well if the movement of the target object is small between two consecutively frames.

2. **Constant velocity:** The second motion model assumes a constant velocity of the target object. The system calculates a velocity $\hat{\mathbf{v}}$ from the previous frames and predicts the new position $\mathbf{x}(t)$ of the target object with
$$\mathbf{x}(t) = \hat{\mathbf{v}}t + \hat{\mathbf{x}}. \tag{3.3}$$
The predicted position $\mathbf{x}(t)$ is used as initial position for the mean-shift tracker.

3. **Constant acceleration:** The third motion model assumes that the target object has constant acceleration. The previous frames are used to evaluate velocity $\hat{\mathbf{v}}$ and acceleration $\hat{\mathbf{a}}$ of the target object. The predicted position in this case is therefore
$$\mathbf{x}(t) = \frac{\hat{\mathbf{a}}t^2}{2} + \hat{\mathbf{v}}t + \hat{\mathbf{x}}. \tag{3.4}$$

In Eq. (3.3)–(3.4) $\hat{\mathbf{x}}$ is the estimated position from the previous frame of the target object and $t$ depends on the frame rate of the camera.

### 3.5.2 Results

We use two video sequences where the standard mean-shift tracker fails. The velocity or acceleration is very high such that the tracker converges to a wrong local maxima. The use of the motion models gives additional input to the reasoning system. The system decide automatically the best fitting motion model using the evaluation function.

#### Coin Sequence 1

In this video sequence, a person plays a shell game with a bottle cap and a coin. The results are shown in Figure 3.14. The person hides a coin with a bottle cap, moves the cap around in rapid movements and the coin reappears. At times, the coin is totally occluded by the bottle cap. Additionally, the person accelerates the bottle cap very rapidly. It is shown that the system is able to track the bottle cap and the coin with rapid movements and total occlusions. As we can see in the first row of the figure, the cap covers the coin and our system reasons correctly that the coin is occluded by the cover. The last position reported by the tracker is used during the occlusion of the coin, the reasoning updates that position depending on the movement of the occluder. Then, the cap and the coin are moved around. In the third row, the coin is released from the cap and the coin is detected and tracked again. After that, it is occluded again. In the fourth row, the bottle cap is moved around with the coin.

### Coin Sequence 2

Figure 3.15 shows the second coin sequence. In this video three objects of interest are tracked: a coin, a cap, and a hand. The system can track all the objects even under full occlusion and generates a correct depth ordering of the objects. In the first row the cap is partially and the coin is totally occluded. After that, the coin reappears on the right side. The second row shows that the cap has changed its position with the coin. Then the cap is totally occluded by the hand and is released near the coin. The third row shows that the coin is covered by the cap and than the cap is occluded by the hand. About 300 frames the user tries to trick the system with fast movements, until he releases the cap. A likely interpretation of the last frame is that the coin is still under the cap.

## 3.6 Conclusion

We showed that our system can update object models in a reliable manner. This was done for an adaptive template tracker and an interest-point based tracker based on SIFT features. The high-level reasoning module has a representation of the entire scene and can therefore trigger the correct model updates for the low-level vision components.

In the last section, we showed that our unified framework addresses two problems of the mean-shift tracker. First, the mean-shift tracking may converges to wrong local maxima, because of rapid movements of the target object. Second, it cannot deal with occlusion events. Additional domain knowledge is added to the high-level reasoning module. The domain knowledge, defined as different motion models, solves the convergence problem. It is shown that the robustness of the mean-shift tracker is greatly improved by considering these two problems together in a unified fashion, while our reasoning algorithm selects the best globally consistent scene interpretation.

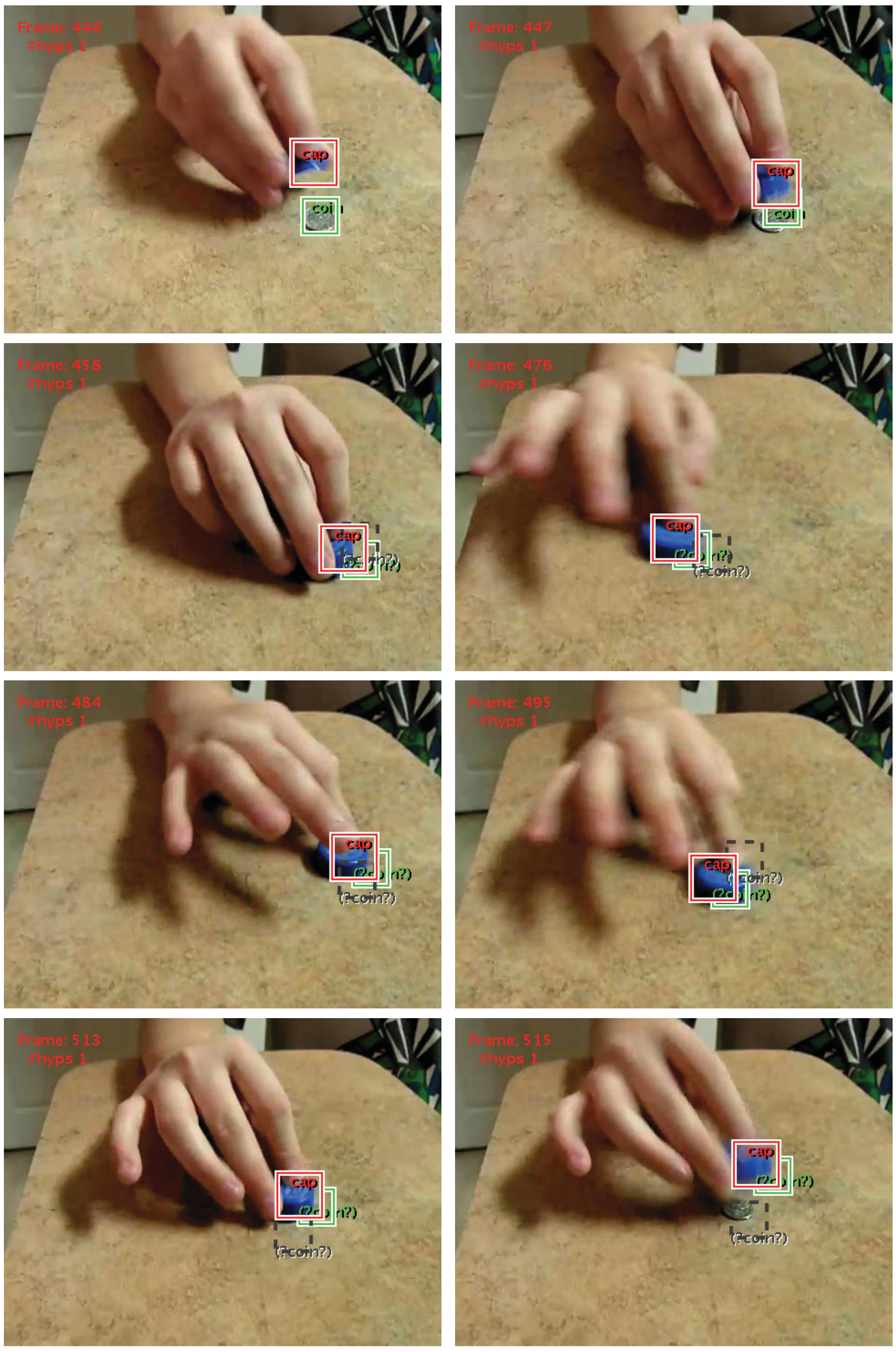


Figure 3.14: Results of coin sequence 1

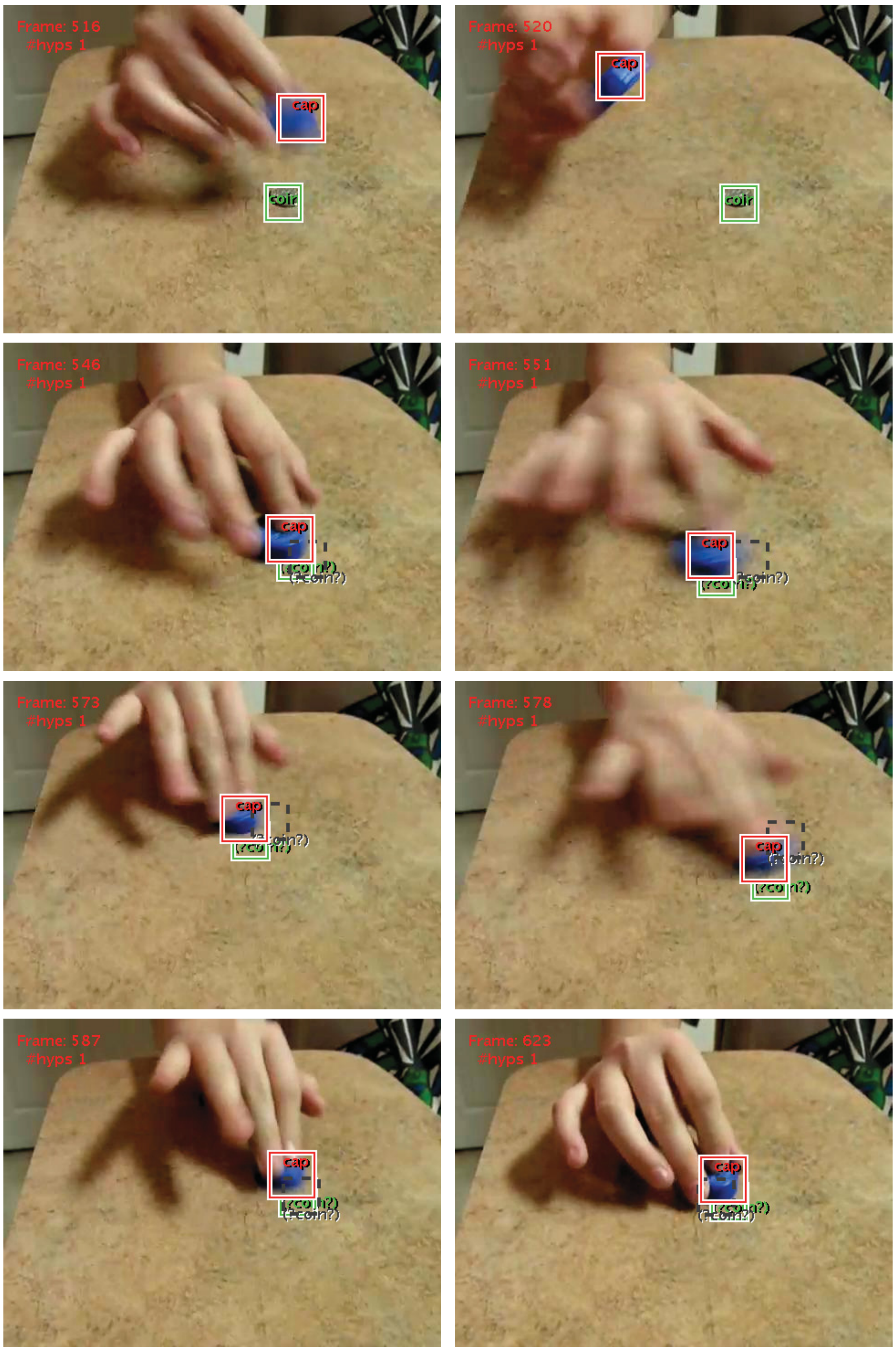


Figure 3.14: Results of coin sequence 1 (continued)

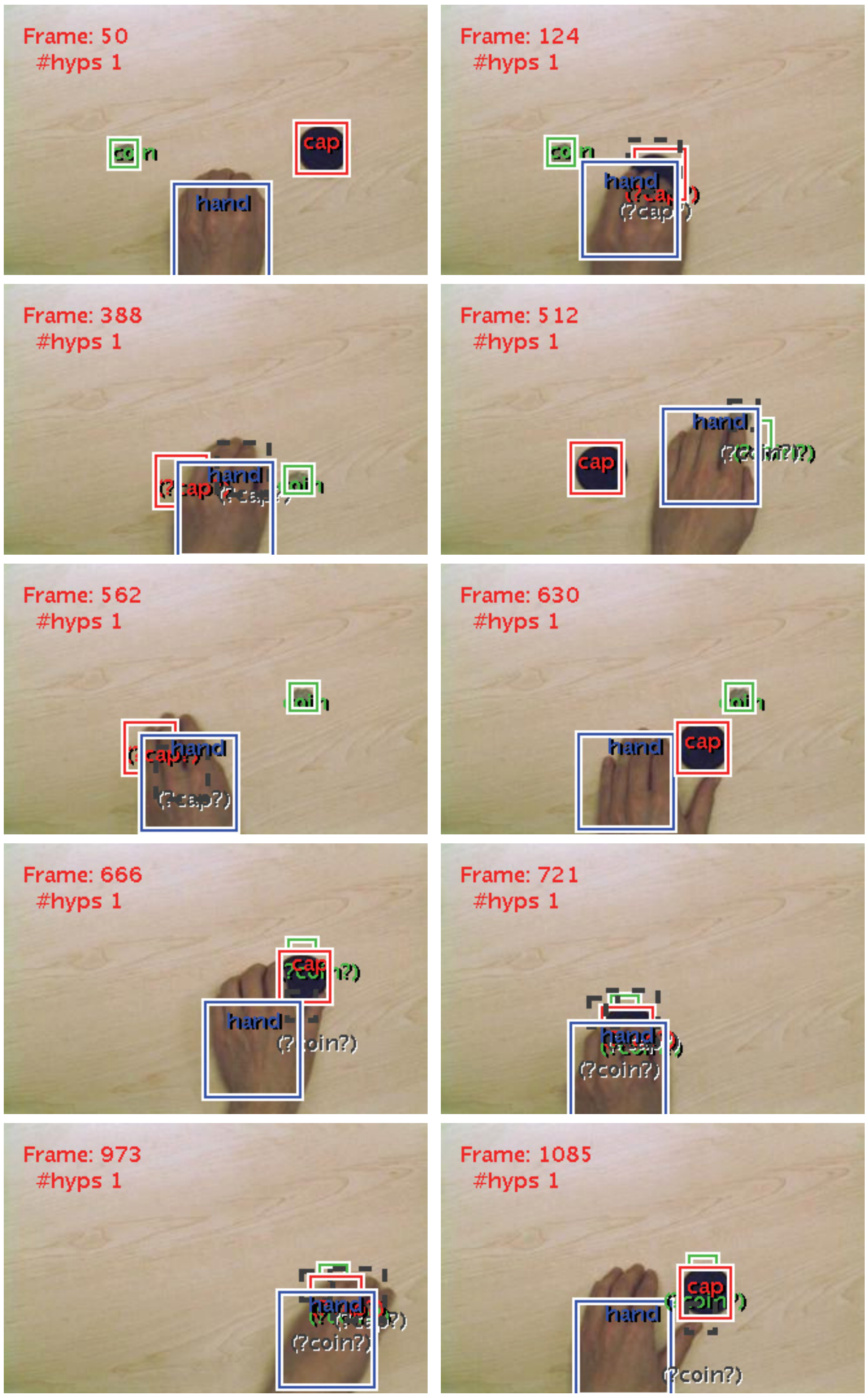


Figure 3.15: Results of coin sequence 2

Chapter 4

# Combining Detectors with Low-Level Features

In the previous chapter we introduced the overall system. In the following chapter we will focus on improvements for the proposed system. One drawback is that the presented low-level vision components are using only one type of visual feature (e.g. SIFTs). Therefore, reliable detection or tracking of the target objects is only possible in restricted environments. Additionally, it is obvious that such a cognitive system cannot accomplish its objectives if it is used in a new environment where the pre-selected feature type is meaningless, e.g. all objects of interest have the same color and different shapes but the pre-selected feature type is a normalized color histogram. Depending on the given setting, a real system should learn and select suitable features from a sufficiently large pool.

Another drawback is that information between low-level vision components is not shared and utilized. For example, if the system has four low-level vision components—one for detecting a cup, a car, a desk, and for detecting roadways—and the first two detectors response with high confidence that they found a cup and a car in an image frame, then our system trusts both detectors even if there is evidence that the car detector is wrong, for example if the desk detector finds a desk but the road detector cannot find a road. It is more likely to find a car on a road, than a car beside a desk with a cup on it and no road in sight. We call this class relationship information. Such class relationship is ignored in the current system but it can be used to improve the overall performance.

In this chapter we introduce a novel approach to deal with both mentioned drawbacks in a unified way: First, we show how a system can learn suitable feature types for an object category; second, how the system can use results from other object categories to improve the overall performance for each category. To develop our approach, we focus on object classification instead of videos sequences. This allows to compare our results on standard image databases with other published results. In addition, we can use results from other research groups and improve their results with our novel approach.

## 4.1 Introduction

Our main idea is that we use a two step learning approach. In the first learning step classifiers are learned for each object category and for each feature type, e.g. if we have $N$ object categories and $M$ different feature types we learn $N \times M$ classifiers. These classifiers from the first step are named *base classifiers*. In the second learning step we combine the output of the base classifiers. This is done to gather additional information from the data which a base classifier could not extract easily. We want to extract how well each pair of feature type and classifier performs. Some feature types may have a poor performance because they are not very suitable for a given object category. Other feature types perform superior, therefore we want to have a higher influence from better performing feature types for the final prediction. Additionally, we want to use class relationship information. Class relationship information can be programmed by hand, but this can be inconvenient if a large number of object categories have to be considered. Therefore, our approach learns relevant class relationships during the second learning step.

We use the standard image database for object detection from the PASCAL Visual Object Classes Challenge [34] and we apply the evaluation protocol of the challenge. The evaluation protocol prescripts every important step, from the splitting of the data to the final evaluation formula.

The idea of combining classifiers in order to obtain improved prediction accuracy has been considered by many researchers (see e.g. [27, 59] for an overview and further references) and has sparked the development of seminal methods such as *boosting* [37, 100]. Here we consider *multilabel* learning problems where each instance may have several labels (unlike in *multiclass* problems where each instance is assigned to a unique class, i.e., has a single label). We present a very simple method for multilabel learning based on the combination of binary base classifiers. That is, we propose to train for each label a binary base classifier. Then we feed the output (i.e., the confidence scores) of these binary base learners into a support vector machine (SVM) [115, 22] in order to improve prediction accuracy.

The basic concept underlying this simple approach resembles *stacking* [118] methods: In the stacking framework the output of several (distinct) base classifiers is combined by a meta-level learner to give an improvement in (binary or multiclass) classification. In the multiclass case usually multiclass classifiers are used as base learners and as meta-level learners (cf. [29]). While the idea behind stacking is that the metalearner shall be able to combine the base learners in a more sophisticated way than doing simple voting or cross-validation [118], in our method the meta-level learner shall grasp interdependencies between the single classes that the base learners have not properly captured.

There has been some discussion whether stacking really gives any improvement over choosing the best base classifier [29] (see also [96] for a related discussion). In any case, it is known that the success of stacking methods depends on the choice of the meta-level learner as well as the kind of input this learner takes from the base learners [112]. Recent suggestions (see [29] for an overview) usually use the probability distributions predicted by the base learners (in some form) as input for the meta-level learner. Similarly, we use the base learners' confidence scores, which is usually simpler than working with probability distributions, as obtaining the latter requires additional optimization methods [88]. The choice for the meta-level learners in stacking approaches ranges from nearest neighbor [77] to tree methods [29]. SVMs have been used as metalearners as well [1, 28, 60].

We tested our algorithm on the popular image classification databases, provided for the VOC challenges in 2006, 2007, and 2009 [33, 30, 32, 34]. Instead of combining essentially different base learners, we kept the base learning algorithm fixed, while using different features for describing the image data. Results show that the combined classifiers outperform the base classifiers in every experiment, which indicates that the combined classifiers are indeed able to extract interdependencies between the individual classes and also the individual classifiers.

## 4.2 General Considerations

In multilabel classification problems there are usually interdependencies between classes. For example, when labeling images it is rather unlikely that an image containing a sheep also shows an aeroplane. On the other hand, images containing cars will usually also contain roads. The art of multilabel classification lies in reliably detecting and exploiting these interdependencies. A base classifier that is trained on enough examples will also learn these interdependencies. However, in the common case where the training set is not sufficiently large, a base learner for cars may not fully grasp the interdependency between cars and roads. This defect can be corrected by having a base learner for roads that will be able to learn roads from more training examples than just those where also cars appear. That way, combining the road classifier with the car classifier in a suitable way will also improve the performance for classification of cars.

Our algorithm uses binary base classifiers that are trained to recognize single classes. In order to better capture interdependencies between the individual classes we propose combining the confidence scores of the different base classifiers by simply feeding them into a support vector machine (SVM) [115, 22].

## 4.3 The Basic Algorithm

We consider the following *multilabel* problem: Given is a set of training examples $\{x_1, \dots, x_n\} \subset X$ together with labels for $N$ different classes $\{C_1, \dots, C_N\}$ (where each $C_k \subseteq X$). That is, for each training instance $x_i$ and each class $C_k$ the respective label is

$$y_{ik} := \begin{cases} +1 & \text{if } x_i \in C_k \\ -1 & \text{otherwise.} \end{cases}$$

As the problem is assumed to be *multilabel* but not necessarily *multiclass*, $y_{ik} = 1$ not necessarily implies that $y_{i\ell} = 0$ for $\ell \neq k$. Thus, the classes $C_k$ in general will not partition the instance space $X$.

We first train for each class $C_k$ a corresponding (binary) base learner $h_k$ that shall be able to predict the labels $y_{ik}$ well. More generally, when using $M$ distinct classification algorithms for each single class $C_k$, we have a total of $N \cdot M$ base classifiers. Each base classifier returns a confidence score $s$ for each training example $x_i$. In our case this will be a real value (the distance to the separating hyperplane) that is positive if the classifier predicts that $x_i$ is in the target class and negative otherwise. However, more generally this score could also be a real number with a different interpretation (e.g. a probability distribution as in recent stacking approaches). Let $s_{jk}(x)$ be the confidence score returned by base classifier $h_{jk}$ for class $C_k$ on training instance $x$. The collected confidence scores are then combined by $N$ metalearners, one for each class $C_k$, where the training set consists of the vectors

$$\mathbf{v}_i = \big(s_{1,1}(x_i), s_{1,2}(x_i), ..., s_{1,N}(x_i), s_{2,1}(x_i), ..., s_{2,N}(x_i), ..., s_{M,N}(x_i)\big) \quad (4.1)$$

for all training instances $x_i$. The label of each $\mathbf{v}_i$ is simply the label $y_{ik}$ of $x_i$ with respect to class $C_k$.

## 4.4 Algorithm Specification for Image Classification

A state-of-the-art approach for image classification is the following: First, features are extracted from the images. Then these features are clustered to generate a visual codebook. Based on that codebook a histogram of "visual words" for every image is built. This approach was inspired by the text classification community where it is called "bag-of-words". In text classification the input features are words, while in image classification descriptors take over this part. A powerful descriptor is the scale-invariant feature transform (SIFT) [68], although there are other very good descriptors for certain image classes. A common approach is to build for every descriptor type a histogram and concatenate these histograms.

Our base classifiers work with different features that are learned with a fixed learning algorithm to give the confidence scores. In our experiments the learning algorithm are either LPBoost [26] or SVMs with linear, polynomial, radial basis function (RBF) [22] and Fisher kernels [52]. As metalearner SVMs [115] are deployed.

## 4.5 Some Related Work

There are several papers dealing with multilabel problems in far more complex ways than our straightforward approach. In [60] SVMs are used for combining different types of features for mapping of proteins to Gene Ontology. Their method trains $\frac{N\cdot(N-1)}{2}$ binary classifiers for a problem with $N$ labels. A function for combining and normalizing the output of the $\frac{N\cdot(N-1)}{2}$ binary classifiers is used and the normalization parameters have to be estimated. Additional knowledge of the problem domain is integrated by a directed acyclic graph to speed up the final classification.

Actually, it is quite common to model and use some additional context information in order to process the output of the base learners. This work is subsumed under the term of *Context Based Concept Fusion* (CBCF). For some references and an integrated approach see e.g. [92].

A simpler approach that has more in common with our method has been suggested in [40]. There it is proposed to use the base learners' output labels as additional coordinates in the training vectors. However, unlike our algorithm this does not consider the confidence of the base learners.

Additionally, other researchers are working on methods using classical stacking. Stacking is an approaches of classifier combination that resembles our method. See e.g. the recent [1] which suggests stacking with SVMs in a multiclass image classification problem.

Compared to these exemplary alternative approaches, we find that our method is appealingly simple, and — as will be seen — works surprisingly well.

## 4.6 Other Ways of Feature/Classifier Combination

In the experiments, we compared our approach with other ways to combine the base classifiers.

### 4.6.1 Using All Features

As a baseline on the first dataset (VOC 2006) we compare our approach to the more direct combination of the features by jointly using them for training the base classifiers. More precisely, the base learner — the boosting algorithm LPBoost — may choose in each boosting iteration the best single feature type for a decision stump. We used the boosting approach with decision stumps, because it is very suitable for combining different kinds of feature types.

### 4.6.2 Binary Stacking

In order to show that our algorithm profits from the confidence information of the other classes, we also did a comparison to the following alternative method where the metalearner uses for learning class $C_k$ not the whole vectors $\mathbf{v}_i$ as given in (4.1) but only the confidence information for the class $C_k$ at question. That is, the training vectors in this case are

$$\mathbf{v}_i = \big(s_{1,k}(x_i), ..., s_{M,k}(x_i)\big)$$

for each training instance $x_i$ with label $y_{ik}$. This corresponds to classical stacking on a binary classification problem, and we call this method *binary stacking* in what follows. Indeed, binary stacking with some minor modifications has been considered and empirically evaluated (among other multilabel algorithms) in [28].

### 4.6.3 The Best Binary Base Classifier

Finally, we do a challenging comparison of our approach to the best binary base classifier. That is, we choose for the prediction of each class $C_k$ the base classifier $h_i$ that gives the best prediction accuracy on the test examples. Note that this classifier is usually unknown beforehand, so that choosing this best binary base classifier has an advantage over our method. However, even in this setting we show that our method gives better results.

## 4.7 Data Sets and Setup

We conducted experiments on the well-known image classification databases taken from the Pascal Visual Object Classes Challenges 2006 (VOC 2006) [33], 2007 (VOC 2007) [30], and 2009 (VOC 2009) [32]. The VOC 2006 dataset contains 10 classes in 5,304 images, on which a total of 9,507 annotated objects can be found. The VOC 2007 and VOC 2009 dataset contain 20 classes, where VOC 2007 has 9,963 images with 24,640 annotated objects and VOC

| dataset | training images | validation images | test images | total |
|---|---|---|---|---|
| VOC 2006 | 1,277 | 1,341 | 2,686 | 5,304 |
| VOC 2007 | 2,501 | 2,510 | 4,952 | 9,963 |
| VOC 2009 | 3,473 | 3,581 | 6,650 | 13,704 |

Table 4.1: Overview of training, validation and test set images of the Pascal Visual Object Classes Challenges datasets.

2009 has 13,704 images with 34,047 annotated objects. All datasets are multilabel. They are split into a fixed training, validation, and test set. Tab. 4.1 shows an overview of the datasets.

The training of the base learners was done on the training dataset. The selection of the kernel and the parameters was done using 5-fold cross-validation on the training data. The independent validation set is used for training the SVM metalearners. We learned the combined classifiers using the validation data. All reported results are from the evaluation on the test data. As evaluation criterion we use the average precision as for the original VOC challenges.

## 4.8 Experiments on the VOC 2006 Dataset

In the experiments on the VOC 2006 dataset, we trained for each of the ten classes a classifier using LPBoost [13, 26]. In every boosting round the best feature for a decision stump is selected from a pool of nine different feature types. The following pages describe the feature types as used by the decision stumps.

### 4.8.1 Texture Statistics of Segmented Regions

The first feature type uses texture statistics of segmented regions. We use the segmentation algorithm from Fussenegger et al. [39]. After applying the segmentation algorithm on an image, seven basic moments are calculated for each segment. In the basic moments Eq. (4.2)–(4.7) all $N$ pixels from a segment are used and $I(\mathbf{x}_n)$ returns the gray value of the $n^{th}$ pixel from a segment. We used the following basic moments: the arithmetic mean $\mu$

$$\mu = \frac{1}{N}\sum_{n=1}^{N} I\left(\mathbf{x}_n\right), \tag{4.2}$$

the variance $\sigma^2$

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}\left(I\left(\mathbf{x}_n\right) - \mu\right)^2, \tag{4.3}$$

the coefficient of variation

$$cov = \frac{\sigma}{\mu}, \tag{4.4}$$

the smoothness

$$R = 1 - \frac{1}{1+\sigma^2}, \tag{4.5}$$

the skewness

$$\gamma_1 = \frac{\frac{1}{N}\sum_{n=1}^{N}\left(I\left(\mathbf{x}_n\right) - \mu\right)^3}{\sigma^3}, \tag{4.6}$$

the kurtosis

$$\gamma_2 = \frac{\frac{1}{N}\sum_{n=1}^{N}\left(I\left(\mathbf{x}_n\right) - \mu\right)^4}{\sigma^4}, \tag{4.7}$$

and the gray value energy[1]

$$E = \sum_{c=1}^{C}\left(hist\left(c\right)\right)^2. \tag{4.8}$$

These seven basic moments are our first feature type.

### 4.8.2 Features from Regions of Interest

In this section we introduce six feature types with similar preprocessing which were also used in [4, 82]. In the first step, the regions of interest are obtained by a scale invariant Harris-Laplace detector [81]. The detector gives for each detected region of interest the center coordinate $(x_1, x_2)$ and their corresponding scale $s$. Square regions of interest with a size of $r \times r$ are extracted, where $r$ depends on the reported scale $s$ of the detector

[1]In Eq. (4.8) the function $hist\,(c)$ delivers the $c^{th}$ value of the normalized histogram from a segment. We note that the gray value energy $E$ returns 1 if the segment is plain-colored. If the gray values are uniformly distributed in a segment then the gray value energy is $\frac{1}{C}$.

multiplied by constant factor[2]. The regions of interest have to be normalized depending on the reported scale. We subsample the regions of interest to a $l \times l$ scaled normalized region if $l < r$ and we use linear interpolation otherwise[3]. The above preprocessing is used for the three feature types subsample gray values, intensity moments, and moment invariants.

Additionally, we do an illumination normalization on the scaled normalized regions. This gives us three additional feature types. We use homomorphic filtering [41, page 191] to make the illumination of the region more even. This is done by applying a Fast Fourier Transformation to the logarithm image $ln(I)$ of the region, under the assumption that image intensity $I(x, y)$ can be modeled as a product of the intensity $i(x, y)$ and reflectance $r(x, y)$ components [41, page 50]. In general, the illumination component of an image changes slowly. The reflectance component changes abruptly and can be associated to the high frequency components after the Fast Fourier Transformation. Therefore, a high-pass filter is used in the frequency domain to suppress the effect of the illumination term $i(x, y)$. After the inverse transformation we have an illumination normalized region. These feature types are marked as 'illumination normalized' in Tab. 4.2 on page 93. The illumination normalized feature types with the same preprocessing were also used by Opelt et al. in [82].

### Subsampled Gray Values

The second feature type is a simple vector of gray values. The gray values are extracted from the preprocessed regions of interest. We subsample the squared regions in every direction by two to reduce the storage requirements. The feature vector has a length of $l^2/4$ from a normalized region of the size $l \times l$.

[2]In our experiments we used 6. We used this small value to capture only information near center coordinates.

[3]Our scale normalized regions have a side length of $l = 16$. It controls the trade-off between runtime behavior and storage requirements. A smaller value results in faster training time but stores less information.

### Basic Intensity Moments

The third feature type is based on basic intensity moments. An intensity moment of degree $d$ and order $p$ and $q$ is defined as

$$M_{I^d_{pq}} = \sum_{x \in X_\Omega} \sum_{y \in Y_\Omega} I^d(x,y)\, x^p y^q \,, \tag{4.9}$$

where the sets $X_\Omega = \{x_1, \ldots, x_N\}$ and $Y_\Omega = \{y_1, \ldots y_N\}$ contain all $x$ and $y$ coordinates of the entire considered region $\Omega$. The final feature vector consist of 10 values from the intensity moments

$$\left( M_{I^1_{10}}, M_{I^1_{01}}, M_{I^1_{11}}, M_{I^1_{20}}, M_{I^1_{02}}, M_{I^2_{10}}, M_{I^2_{01}}, M_{I^2_{11}}, M_{I^2_{20}}, M_{I^2_{02}} \right). \tag{4.10}$$

### Moment Invariants

The fourth feature type are based on a subset of affine and photometric moment invariants from the work of Gool et al. in [42]. On the preprocessed regions the moment invariants are calculated by moments up to the order of two, which have been found to be more robust against noise [111].

In the following equations the pair $(x, y)$ represents the coordinates of a region pixel and $I(x, y)$ returns the gray value at $(x, y)$. The equation

$$M_{S_{C_{pq}}} = \sum_{x \in X_\Omega} \sum_{y \in Y_\Omega} x^p y^q \tag{4.11}$$

defines the shape moment of the order $p$ and $q$ using the closed contour $C$ of the region $\Omega$ [42]. The intensity moment of the order $p$ and $q$ is defined by

$$M_{I_{C_{pq}}} = M_{I^1_{C_{pq}}} = \sum_{x \in X_\Omega} \sum_{y \in Y_\Omega} I(x,y)\, x^p y^q \,. \tag{4.12}$$

The notations of the above equations are simplified to $M_{S_{pq}}$ and $M_{I_{pq}}$ if only one closed contour $C$ is used in the calculation.

Two assumptions are made by Gool et al. for the moment invariant features: First, the camera is positioned relatively far away from the object and the second one is that the regions are planar. The two assumptions are used to classify moment invariants that are affine and photometric invariant based on a combination of intensity and shape moments. Gool et al. used two photometric models to subclassify the moment invariants. The first model is pure scaling of the intensity values

$$I'(x,y) = s \cdot I(x,y)\,; \tag{4.13}$$

the second one is scaling of the intensity values plus an offset

$$I'(x,y) = s \cdot I(x,y) + o. \tag{4.14}$$

Moments invariants invariant to the second model have been found more stable in practice [94, 117]. On the next pages the selected affine and photometric invariants are listed with a brief description of their properties.

**Selection of First-order Moment Invariants** Our first selected moment is affine and photometric invariant using the model from Eq. (4.13). It uses two intensity moments and one shape moment.

$$\frac{1}{M_{S_{C_{00}}}} \begin{vmatrix} \frac{M_{I_{C_{10}}}}{M_{I_{C_{00}}}} - \frac{M_{S_{C_{10}}}}{M_{S_{C_{00}}}} & \frac{M_{I_{C_{10}}}}{M_{I_{C_{00}}}} - \frac{M_{I_{D_{10}}}}{M_{I_{D_{00}}}} \\ \frac{M_{I_{C_{01}}}}{M_{I_{C_{00}}}} - \frac{M_{S_{C_{01}}}}{M_{S_{C_{00}}}} & \frac{M_{I_{C_{01}}}}{M_{I_{C_{00}}}} - \frac{M_{I_{D_{01}}}}{M_{I_{D_{00}}}} \end{vmatrix} \tag{4.15}$$

Our second selected moment invariant uses four moments: two shape moments and two intensity moments. The resulting combination is invariant in respect of affine transformations and again only photometric scaling.

$$\begin{vmatrix} \frac{M_{I_{C_{10}}}}{M_{I_{C_{00}}}} - \frac{M_{S_{C_{10}}}}{M_{S_{C_{00}}}} & \frac{M_{I_{D_{10}}}}{M_{I_{D_{00}}}} - \frac{M_{C_{C_{10}}}}{M_{S_{C_{00}}}} \\ \frac{M_{I_{C_{01}}}}{M_{I_{C_{00}}}} - \frac{M_{S_{C_{01}}}}{M_{S_{C_{00}}}} & \frac{M_{I_{D_{01}}}}{M_{I_{D_{00}}}} - \frac{M_{S_{C_{01}}}}{M_{S_{C_{00}}}} \end{vmatrix} \begin{vmatrix} \frac{M_{I_{C_{10}}}}{M_{I_{C_{00}}}} - \frac{M_{S_{C_{10}}}}{M_{S_{C_{00}}}} & \frac{M_{S_{D_{10}}}}{M_{S_{D_{00}}}} - \frac{M_{S_{C_{10}}}}{M_{S_{C_{00}}}} \\ \frac{M_{I_{C_{01}}}}{M_{I_{C_{00}}}} - \frac{M_{S_{C_{01}}}}{M_{S_{C_{00}}}} & \frac{M_{S_{D_{01}}}}{M_{S_{D_{00}}}} - \frac{M_{S_{C_{01}}}}{M_{S_{C_{00}}}} \end{vmatrix} \tag{4.16}$$

The next two moment invariants are affine and photometric invariant, where the photometric model is Eq. (4.14).

$$\frac{M_{I_{C_{00}}}}{M_{I_{C_{00}}} M_{S_{D_{00}}} - M_{I_{D_{00}}} M_{S_{C_{00}}}} \begin{vmatrix} \frac{M_{I_{C_{10}}}}{M_{I_{C_{00}}}} - \frac{M_{S_{C_{10}}}}{M_{S_{C_{00}}}} & \frac{M_{I_{C_{10}}}}{M_{I_{C_{00}}}} - \frac{M_{S_{D_{10}}}}{M_{S_{D_{00}}}} \\ \frac{M_{I_{C_{01}}}}{M_{I_{C_{00}}}} - \frac{M_{S_{C_{01}}}}{M_{S_{C_{00}}}} & \frac{M_{I_{C_{01}}}}{M_{I_{C_{00}}}} - \frac{M_{S_{D_{01}}}}{M_{S_{D_{00}}}} \end{vmatrix} \tag{4.17}$$

$$\frac{M_{I_{C_{00}}} M_{I_{D_{00}}} M_{S_{D_{00}}}}{(M_{I_{C_{00}}} M_{S_{D_{00}}} - M_{I_{D_{00}}} M_{S_{C_{00}}})^2} \begin{vmatrix} \frac{M_{I_{C_{10}}}}{M_{I_{C_{00}}}} - \frac{M_{S_{C_{10}}}}{M_{S_{C_{00}}}} & \frac{M_{I_{C_{10}}}}{M_{I_{C_{00}}}} - \frac{M_{S_{D_{10}}}}{M_{S_{D_{00}}}} \\ \frac{M_{I_{C_{01}}}}{M_{I_{C_{00}}}} - \frac{M_{S_{C_{01}}}}{M_{S_{C_{00}}}} & \frac{M_{I_{C_{01}}}}{M_{I_{C_{00}}}} - \frac{M_{S_{D_{01}}}}{M_{S_{D_{00}}}} \end{vmatrix} \tag{4.18}$$

**Selection of Second-order Moment Invariants** The combination of the moments to second-order affine and photometric moment invariants is more complex. Therefore, we define the following auxiliary variables:

$$\begin{array}{rclrcl}
\alpha & = & M_{I_{11}}M_{I_{00}} - M_{I_{10}}M_{I_{01}} & & & \\
\beta & = & M_{I_{20}}M_{I_{00}} - M_{I_{10}}^2 & A & = & M_{S_{11}}M_{S_{00}} - M_{S_{10}}M_{S_{01}} \\
\gamma & = & M_{I_{02}}M_{I_{00}} - M_{I_{01}}^2 & B & = & M_{S_{20}}M_{S_{00}} - M_{S_{10}}^2 \\
\delta & = & M_{S_{00}}M_{I_{10}} - M_{S_{10}}M_{I_{00}} & C & = & M_{S_{02}}M_{S_{00}} - M_{S_{01}}^2 \\
\epsilon & = & M_{S_{00}}M_{I_{01}} - M_{S_{01}}M_{I_{00}} & & &
\end{array}$$

The next three moment invariants are affine invariant, but not photometric invariant. Eq. (4.19) uses one intensity moment and it is defined on page 648, Eq. 5 in [42]. The Eq. (4.20)–(4.21) consist of an intensity moment and a shape-moment. Eq. (4.21) combines intensity and shape moments up to the second order.

$$\frac{\beta\gamma - \alpha^2}{M_{I_{00}}^6} \tag{4.19}$$

$$\frac{\beta\epsilon^2 - 2\alpha\delta\epsilon + \gamma\delta^2}{M_{S_{00}}^8} \tag{4.20}$$

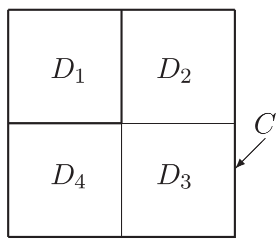
$$\frac{(\alpha A - \gamma B)^2 + (\alpha A - \beta C)^2 + 2(\beta A - \alpha B)(\gamma A - \alpha C)}{M_{I_{00}}^6 M_{S_{00}}^6} \tag{4.21}$$

The Eq. (4.22)–(4.23) are affine and photometric invariant. These moment invariants are based on model Eq. (4.13) which considers only pure scaling of the intensity values.

$$\frac{\beta\gamma - \alpha^2}{M_{I_{00}}^4 M_{S_{00}}^2} \tag{4.22}$$

$$\frac{\beta\epsilon^2 - 2\alpha\delta\epsilon + \gamma\delta^2}{M_{I_{00}}^4 M_{S_{00}}^2} \tag{4.23}$$

Some moment invariants need two shape moments with different contours $C$ and $D$. The first contour $C$ is the boundary of our region of interest. A fraction of the region of interest is used as the second contour $D$. The squared region of interest is divided into four equal region as seen in Figure 4.1. The shape moments of the entire region and one out of four of the region are used to calculated the moment invariants with two contours. This approach is not invariant and therefore we repeat the calculation for the remaining three contours $D_2, \ldots, D_4$ and all four result values are summed as



$$f'(C, D) = \sum_{m=1}^{4} f(M_{S_C}, M_{S_{D_m}}) \tag{4.24}$$

Figure 4.1: The partitioning of a region of interest into two contours. For example, Eq. (4.16) needs two contours $C$ and $D$ therefore the function $f$ is replaced by Eq. (4.16).

it can be seen in Eq. (4.24). The result of the summation is used as one feature value. The feature vector consists of nine feature values based on Eq. (4.15)–(4.23).

### 4.8.3 SIFT Based Features from Regions of Interest

We use two scale-invariant feature transform (SIFT) features from regions of interest. The first feature type is the original SIFT feature from Lowe [69, 68]. The sift feature is invariant to translation, rotation, and scaling transformations [67]. It is to some degree robust against perspective transformations and illumination changes [80]. The feature vector is based on magnitudes and orientations of image gradients in a histogram representation. The dimension is 128.

The second feature type is based on the original SIFT feature and is called PCA-SIFT [57]. The detection procedure is the same as for the SIFT features. The feature calculation is different and is based on magnitudes of gradients in the region of interest. These gradient patches are normalized to $39 \times 39$ patches and all normalized patches are projected to a lower subspace using principal component analysis (PCA) [53]. The authors Ke and Sukthankar claim that the PCA-SIFT can be more distinctive. A detailed evaluation of image features and their properties can be found in [80]. One PCA-SIFT feature property is obvious: a PCA-SIFT feature can be faster matched with a reference database of PCA-SIFT features, because the feature dimension is only 20.

| label | feature type |
|---|---|
| $h_1$ | texture statistics of segments |
| $h_2$ | sub-sampled gray values |
| $h_3$ | sub-sampled gray values (illumination normalized) |
| $h_4$ | intensity moments |
| $h_5$ | intensity moments (illumination normalized) |
| $h_6$ | moment invariants |
| $h_7$ | moment invariants (illumination normalized) |
| $h_8$ | SIFTs |
| $h_9$ | PCA-SIFTs |

Table 4.2: Feature types for the experiments on the VOC 2006 dataset.

### 4.8.4 Results

For our algorithm we used each feature type together with LPBoost. An overview over the used feature types is given in Tab. 4.2. We obtain for each class a total of nine binary classifiers as base learners. The output (the distance to the separating hyperplane) of these base learners (for all classes) was then fed into an SVM metalearner. For completeness, we tested our approach with different kernels. However, results show that the choice of the SVM kernel function is not critical.

We compared our approach to the best of the base classifiers. An overview of the performance of each descriptor on the ten classes can be found in Tab. 4.3. It can be seen from Tab. 4.4 and Figure 4.3 that our combined classifier outperforms the individual classifiers, even if we choose for each class that base classifier that gives the best performance on the test set.

As already indicated before, another comparison was made to the algorithm where the weak learner of LPBoost may choose in each boosting iteration one reference feature among the nine different feature types. Thus, in this setting the boosting algorithm is not restricted to a single feature type and may choose the best feature with the optimal threshold. This approach (denoted 'original classifier $h_{1..9}$' in Tab. 4.4 and 4.2) has been used in the VOC 2006 Challenge and is used as a base line for our experiments. For more details see [6]. While the results for this alternative algorithm sometimes improve even over the best single classifier, it is still outperformed by our algorithm. The collected results can be found in Tab. 4.4.

| class | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ | $h_6$ | $h_7$ | $h_8$ | $h_9$ |
|---|---|---|---|---|---|---|---|---|---|
| bicycle | 17.5 | 40.9 | 40.6 | 41.8 | 39.4 | 37.9 | 31.2 | **56.8** | 54.7 |
| bus | 9.6 | 25.9 | **30.2** | 25.4 | 19.4 | 13.6 | 16.3 | 16.0 | 25.6 |
| car | 40.8 | **70.7** | 70.0 | 48.7 | 51.3 | 38.2 | 41.4 | 56.9 | 60.5 |
| cat | 16.2 | **29.6** | 28.7 | 15.5 | 15.3 | 14.5 | 14.9 | 15.7 | 15.1 |
| cow | 9.0 | 18.2 | 15.4 | 17.6 | 18.7 | 10.2 | 17.5 | 17.6 | **26.1** |
| dog | 14.1 | 20.1 | 20.2 | 18.1 | 14.4 | 15.9 | 15.4 | **22.9** | 15.7 |
| horse | 9.9 | 12.3 | 16.9 | **19.6** | 13.1 | 14.7 | 11.5 | 10.1 | 12.2 |
| motorbike | 13.2 | 27.1 | 29.1 | 33.0 | **39.9** | 28.4 | 26.2 | 10.5 | 12.3 |
| person | 29.7 | 39.9 | 36.9 | 38.3 | **43.5** | 36.0 | 42.6 | 31.5 | 32.0 |
| sheep | 9.0 | 25.0 | 25.3 | 21.9 | 19.5 | 11.1 | 15.9 | 29.4 | **36.4** |
| **avg** | 16.9 | 31.0 | 31.3 | 28.0 | 27.4 | 22.0 | 23.3 | 26.8 | 29.1 |

Table 4.3: Average precision of the individual classifiers on the VOC 2006 dataset in percent. Bold values indicate the best classifier on a given class for the test set.

| class | $\max(h_1, \ldots, h_9)$ | original $h_{1..9}$ | our | | |
|---|---|---|---|---|---|
| | | | linear | polynomial | RBF |
| bicycle | 56.84 | 61.12 | **61.36** | 56.16 | 60.77 |
| bus | 30.17 | 27.32 | **51.66** | 50.00 | 51.54 |
| car | 70.72 | 70.92 | 74.03 | **76.30** | 74.83 |
| cat | 29.60 | 24.41 | 37.13 | **41.15** | 37.98 |
| cow | **26.14** | 18.92 | 23.41 | 24.20 | 25.56 |
| dog | 22.86 | 25.88 | 32.16 | 34.95 | **36.41** |
| horse | 19.58 | 12.12 | 22.44 | **27.44** | 20.86 |
| motorbike | 39.90 | 33.19 | 47.09 | 46.92 | **48.35** |
| person | 43.47 | 35.16 | 42.55 | **46.56** | 43.04 |
| sheep | 36.37 | 29.39 | **41.87** | 37.07 | 40.55 |
| **avg** | 37.57 | 33.84 | 43.37 | **44.07** | 43.99 |

Table 4.4: Comparison of the best individual classifier, the classifier using all descriptor types in the beginning, and our approach on the VOC 2006 dataset. Results are in percentage using the average precision measure. Bold values indicate the optimal method.

| class | $h_{1..9}$ | stacking | our (RBF) |
|---|---|---|---|
| bicycle | **61.12** | 56.77 | 60.77 |
| bus | 27.32 | 27.52 | **51.54** |
| car | 70.92 | 65.86 | **74.83** |
| cat | 24.41 | 14.75 | **37.98** |
| cow | 18.92 | 11.83 | **25.56** |
| dog | 25.88 | 17.55 | **36.41** |
| horse | 12.12 | 12.42 | **20.86** |
| motorbike | 33.19 | 30.21 | **48.35** |
| person | 35.16 | 34.25 | **43.04** |
| sheep | 29.39 | 34.00 | **40.55** |
| **avg** | 33.84 | 30.52 | **43.99** |

Table 4.5: Comparison of the classifier using all descriptor types in the beginning, binary stacking, and our approach on the VOC 2006 dataset. Results are in percentage using the average precision measure. For binary stacking and our method we report the values obtained for the RBF kernel.

Figure 4.2 and Tab. 4.5 show a comparison of our method with the binary stacking approach. For binary stacking we used the same nine base classifiers that are combined by an SVM. We report only the results of binary stacking with an RBF-kernel SVM as metalearner, which performed best. The resulting performance of binary stacking is on some classes slightly better than our base line where all features are used from the beginning. However, on some other classes binary stacking suffers a performance decrease of up to 9.6%. The mean average precision of binary stacking is 30.52%, which compared to our base line means a performance loss of 3.32%. These experiments may also confirm doubts concerning the utility of stacking. However, as already mentioned before, usually class probabilities instead of confidence scores are used for stacking. The choice of the latter may affect the results to the negative. Indeed, a similar stacking method [1] in an image classification problem where class probabilities are the input for the metalearner has been more successful. That confidence scores work fine in our proposed method can be interpreted the way that our metalearner SVMs do the normalization that also has to be done when trying to obtain probability distributions from confidence scores.

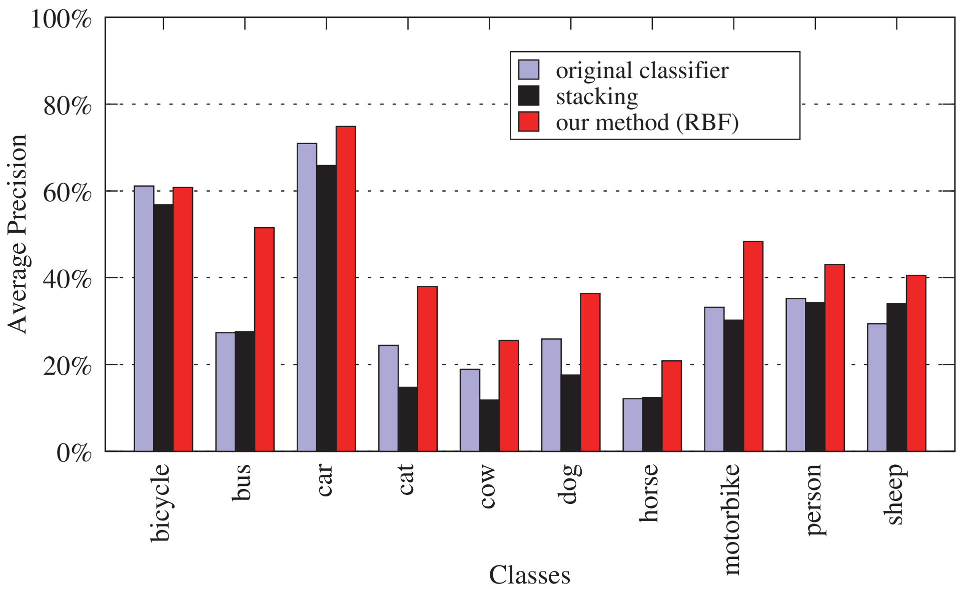


Figure 4.2: Comparison of the classifier using all descriptor types in the beginning, binary stacking, and our approach on the VOC 2006 dataset. Results are in percentage using the average precision measure. For binary stacking and our method we report the values obtained for the RBF kernel.

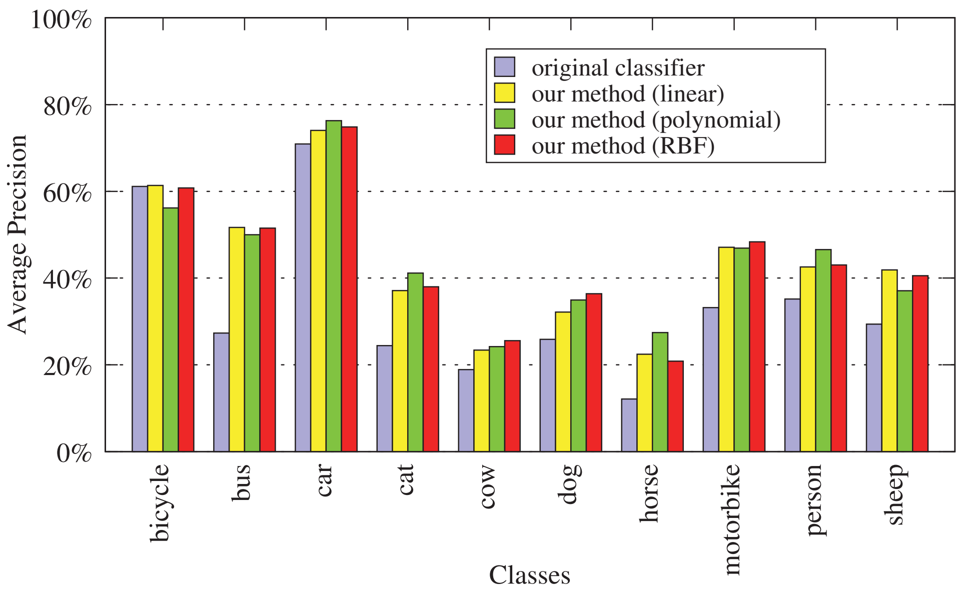


Figure 4.3: Comparison of the classifier using all descriptor types in the beginning and our approach using different kernels on the VOC 2006 dataset. Results are in percentage using the average precision measure. The choice of the SVM kernel function can be seen to be not critical.

## 4.9 Experiments on the VOC 2007 Dataset

In the experiments on the VOC 2007 database we used base classifiers which gives state-of-the-art performance on the VOC 2007 database. The authors Perronnin and Dance gave us the two base classifiers from their work [85]. We show in the experiments that our algorithm improves the classification results using their base classifiers.

The first classifier $h_1$ is based on texture information using the SIFT descriptor [68]. Dimension reduction of the final SIFT description is done using principal component analysis (PCA) [53], which gives us a reduced 50 dimensional vector.

The second classifier $h_2$ is based on Gaussian weighted local color information ($h_2$) [86, page 8]. The color information is extracted from an RGB image. The region of interest is divided into $4 \times 4$ sub-regions. On each sub-region the mean and standard deviation of each RGB channel is calculated. This gives us a $4 \times 4 \times 3 \times 2 = 96$ feature vector. The feature vector is projected to a 50 dimensional subspace using PCA.

Both descriptors are extracted from a dense grid at five different scales. Each classifier is learned with the Fisher kernels framework [85]. Tab. 4.6 shows that the performance of the classifier using the SIFT descriptor yields consistently better results than the descriptor based on the local color information (with the only exception being the class 'pottedplant'). In spite of this and the fact that the information of two classifiers is quite limited, our method was able to improve the mean of the average precision across all the 20 categories by up to 3.46% (for the RBF kernel). When using cross-validation to choose the kernel, the RBF kernel is selected for all classes except one (cf. Tab. 4.6) giving the same average precision as for the RBF kernel. The collected results can be found in Tab. 4.6. Figure 4.4 shows the data as a bar diagram.

In Tab. 4.7 and Figure 4.5 we compare our method with binary stacking. In this experiment a linear kernel gave the best results for binary stacking, but it can be seen that our method gives better results for 14 classes. Our average improvement to the base classifiers is 3.46%, whereas binary stacking only improves by 1.66%.

| class | $h_1$ | $h_2$ | our (linear) | impr. | our (poly.) | impr. | our (RBF) | impr. |
|---|---|---|---|---|---|---|---|---|
| aeroplane | 66.41 | 59.51 | 65.88 | -0.53 | 65.47 | -0.94 | **66.71**$^*$ | 0.30 |
| bicycle | 47.31 | 35.45 | 51.09 | 3.78 | 53.23 | 5.92 | **53.42**$^*$ | 6.11 |
| bird | 44.45 | 42.67 | 49.99 | 5.54 | 53.06 | 8.61 | **53.47**$^*$ | 9.02 |
| boat | 58.87 | 41.12 | 63.21 | 4.34 | **63.26** | 4.39 | 62.38$^*$ | 3.51 |
| bottle | 24.18 | 15.16 | 25.97 | 1.79 | **27.53** | 3.35 | 23.93$^*$ | -0.25 |
| bus | **52.42** | 34.24 | 51.65 | -0.77 | 43.64 | -8.78 | 45.75$^*$ | -6.67 |
| car | 70.70 | 56.47 | 70.67 | -0.03 | **73.32**$^*$ | 2.62 | **73.32** | 2.62 |
| cat | 45.30 | 39.49 | 44.87 | -0.43 | 44.78 | -0.52 | **46.30**$^*$ | 1.00 |
| chair | 47.11 | 37.78 | 50.68 | 3.57 | 49.82 | 2.71 | **50.72**$^*$ | 3.61 |
| cow | 31.25 | 15.03 | 29.00 | -2.25 | 31.28 | 0.03 | **32.99**$^*$ | 1.74 |
| diningtable | 38.21 | 35.75 | 42.29 | 4.08 | 43.12 | 4.91 | **44.71**$^*$ | 6.50 |
| dog | 40.98 | 33.44 | 38.77 | -2.21 | 40.42 | -0.56 | **41.95**$^*$ | 0.97 |
| horse | 67.77 | 64.48 | 71.44 | 3.67 | 73.46 | 5.69 | **73.48**$^*$ | 5.71 |
| motorbike | 52.37 | 46.02 | 55.07 | 2.70 | 56.05 | 3.68 | **57.85**$^*$ | 5.48 |
| person | 80.17 | 78.33 | 82.43 | 2.26 | 83.01 | 2.84 | **83.22**$^*$ | 3.05 |
| pottedplant | 24.30 | 27.14 | 28.71 | 1.57 | 29.11 | 1.97 | **32.92**$^*$ | 5.78 |
| sheep | 27.32 | 25.48 | 36.47 | 9.15 | 28.76 | 1.44 | **38.79**$^*$ | 11.47 |
| sofa | **44.36** | 31.57 | 41.81 | -2.55 | 40.67 | -3.69 | 40.73$^*$ | -3.63 |
| train | 65.21 | 54.42 | 66.88 | 1.67 | 69.52 | 4.31 | **69.87**$^*$ | 4.66 |
| tvmonitor | 41.34 | 34.26 | 44.54 | 3.20 | **48.27** | 6.93 | 46.66$^*$ | 5.32 |
| **avg** | 48.50 | 40.39 | 50.57 | 2.07 | 50.89 | 2.39 | **51.96** | 3.46 |

Table 4.6: Average precision on the VOC 2007 dataset in percent for the two base classifiers as well as for our method with linear, polynomial and RBF kernel. The best method for each class is indicated by a bold entry. Starred values indicate which kernel is chosen by cross-validation.

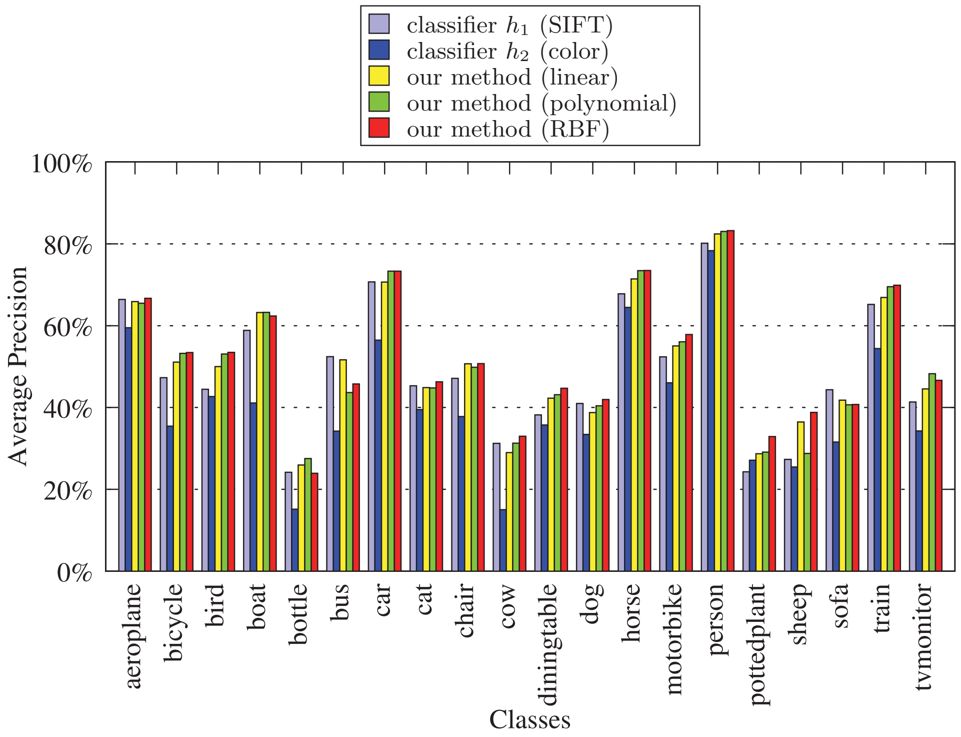


Figure 4.4: Average precision on the VOC 2007 dataset in percent for the two base classifiers as well as for our method with linear, polynomial and RBF kernel.

## 4.10 Experiments on the VOC 2009 Dataset

We used traditional graylevel SIFT [69] and SIFT with color information [114] for the VOC 2009 dataset. The authors van de Sande et al. showed that color information can boost the recognition performance up to 8 percent. They won the challenge in 2008 [31]. The implementation from [114] extracted the features in our experiments. Two different methods for choosing the region of interest are applied. First, the Harris-Laplace detector [79] detects region of interest. The second method applies a dense grid on the image. At grid points SIFT descriptors are extracted. Cross-validation selected the cluster size for the codebook. Additionally, cross-validation decided which color spaces are processed in the color SIFT descriptors. Tab. 4.8 shows an overview of the features used for the base classifiers. The base classifiers are trained with an SVM on the training dataset.

Tab. 4.9 shows the performance of the base classifiers. It shows that

| class | $h_1$ | $h_2$ | stacking | impr. | our (RBF) | impr. |
|---|---|---|---|---|---|---|
| aeroplane | 66.41 | 59.51 | **68.16** | 1.75 | 66.71 | 0.30 |
| bicycle | 47.31 | 35.45 | 48.61 | 1.30 | **53.42** | 6.11 |
| bird | 44.45 | 42.67 | 49.37 | 4.92 | **53.47** | 9.02 |
| boat | 58.87 | 41.12 | 60.65 | 1.78 | **62.38** | 3.51 |
| bottle | 24.18 | 15.16 | **25.76** | 1.58 | 23.93 | -0.25 |
| bus | **52.42** | 34.24 | 51.15 | -1.27 | 45.75 | -6.67 |
| car | 70.70 | 56.47 | 69.72 | -0.98 | **73.32** | 2.62 |
| cat | 45.30 | 39.49 | **46.86** | 1.56 | 46.30 | 1.00 |
| chair | 47.11 | 37.78 | 45.18 | -1.93 | **50.72** | 3.61 |
| cow | 31.25 | 15.03 | 30.96 | -0.29 | **32.99** | 1.74 |
| diningtable | 38.21 | 35.75 | 38.51 | 0.30 | **44.71** | 6.50 |
| dog | 40.98 | 33.44 | **43.20** | 2.22 | 41.95 | 0.97 |
| horse | 67.77 | 64.48 | 70.61 | 2.84 | **73.48** | 5.71 |
| motorbike | 52.37 | 46.02 | 56.45 | 4.08 | **57.85** | 5.48 |
| person | 80.17 | 78.33 | 81.83 | 1.66 | **83.22** | 3.05 |
| pottedplant | 24.30 | 27.14 | 29.50 | 2.36 | **32.92** | 5.78 |
| sheep | 27.32 | 25.48 | 30.97 | 3.65 | **38.79** | 11.47 |
| sofa | 44.36 | 31.57 | **45.15** | 0.79 | 40.73 | -3.63 |
| train | 65.21 | 54.42 | 67.76 | 2.55 | **69.87** | 4.66 |
| tvmonitor | 41.34 | 34.26 | 42.75 | 1.41 | **46.66** | 5.32 |
| **avg** | 48.50 | 40.39 | 50.16 | 1.66 | **51.96** | 3.46 |

Table 4.7: Average precision on the VOC 2007 dataset in percent for the two base classifiers as well as for binary stacking and our method with RBF kernel.

| label | pre-processing | cluster size | feature type |
|---|---|---|---|
| $h_1$ | Harris-Laplace | 5400 | SIFT |
| $h_2$ | Harris-Laplace | 10240 | Color-SIFT |
| $h_3$ | Harris-Laplace | 5120 | Color-SIFT |
| $h_4$ | Harris-Laplace | 40960 | Color-SIFT |
| $h_5$ | Harris-Laplace | 81920 | Color-SIFT |
| $h_6$ | dense grid | 5120 | HSV-SIFT |
| $h_7$ | dense grid | 20480 | Opponent-SIFT |
| $h_8$ | dense grid | 40960 | Opponent-SIFT |
| $h_9$ | dense grid | 40960 | RG-SIFT |
| $h_10$ | dense grid | 20480 | transformed Color-SIFT |

Table 4.8: Feature types for the experiments on the VOC 2009 dataset.

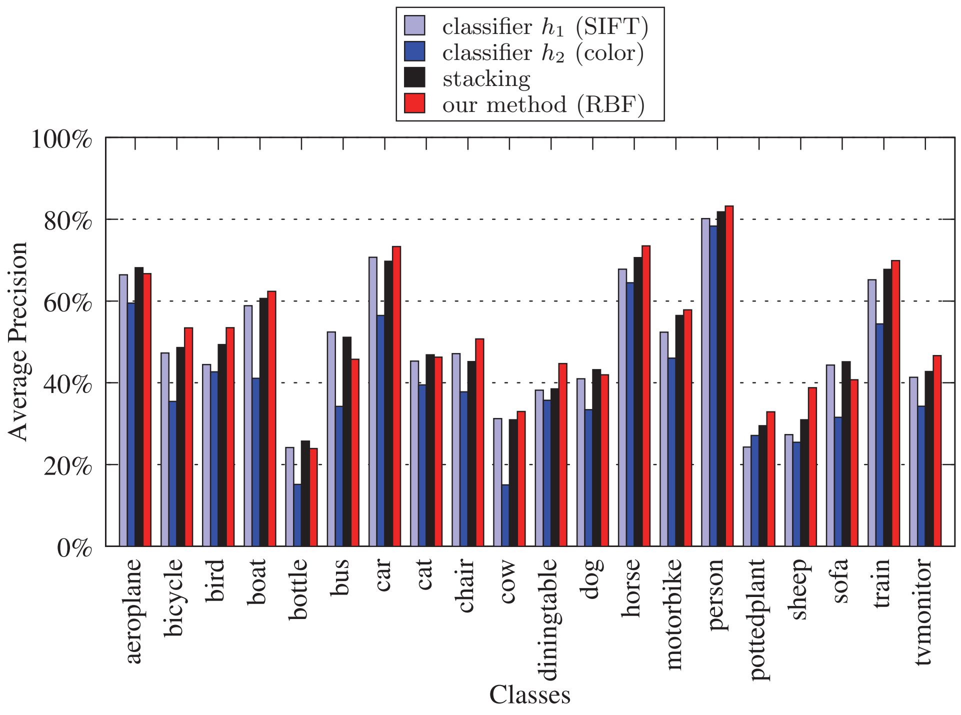


Figure 4.5: Average precision on the VOC 2007 dataset in percent for the two base classifiers as well as for binary stacking and our method with RBF kernel.

traditional SIFT descriptors have a good performance. Color information is useful for some categories, e.g. cow and pottedplant.

Tab. 4.10 compares the best individual classifier with the results of our method. Our method improves the result for every category. The RBF kernel has the best performance on nine categories, followed by the polynomial kernel.

Tab. 4.11 shows the comparison between the best individual classifiers, stacking, and our method. Stacking has a performance loss of −3.06% compared to the best individual classifiers worse. Our method improves by 4.58%.

## 4.11 Conclusion

Figure 4.6 shows an summary of the results. For all three image databases the proposed method improves the results. The RBF kernel of the support vector machine is a good choice over all test data. While our presented

| class | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ | $h_6$ | $h_7$ | $h_8$ | $h_9$ | $h_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| aeroplane | 73.50 | 72.15 | 71.63 | 72.63 | 73.01 | 71.23 | 72.55 | 72.20 | 72.69 | 72.03 |
| bicycle | 45.17 | 39.02 | 38.40 | 43.46 | 44.23 | 37.68 | 39.24 | 40.45 | 40.48 | 39.21 |
| bird | 46.78 | 47.87 | 46.28 | 48.23 | 48.99 | 36.49 | 38.31 | 39.27 | 40.08 | 40.51 |
| boat | 54.84 | 50.19 | 51.14 | 52.26 | 52.17 | 44.36 | 46.85 | 45.73 | 43.46 | 43.58 |
| bottle | 20.48 | 21.88 | 21.38 | 22.76 | 22.89 | 23.00 | 20.45 | 21.22 | 20.08 | 19.21 |
| bus | 58.34 | 53.68 | 52.26 | 54.64 | 55.31 | 52.23 | 55.65 | 55.24 | 53.40 | 54.92 |
| car | 50.24 | 44.15 | 43.84 | 45.88 | 47.21 | 40.41 | 41.14 | 40.72 | 38.34 | 40.13 |
| cat | 47.02 | 42.64 | 41.88 | 43.50 | 43.64 | 43.29 | 45.11 | 43.57 | 44.46 | 42.54 |
| chair | 44.87 | 45.69 | 45.54 | 47.38 | 46.84 | 43.75 | 44.85 | 46.02 | 47.38 | 45.35 |
| cow | 25.34 | 29.05 | 26.92 | 29.24 | 28.55 | 23.48 | 20.39 | 17.74 | 25.66 | 21.11 |
| diningtable | 32.87 | 33.45 | 32.85 | 35.92 | 35.86 | 34.27 | 33.90 | 33.59 | 35.13 | 31.16 |
| dog | 41.14 | 33.03 | 33.64 | 34.10 | 34.51 | 32.51 | 31.21 | 31.52 | 31.93 | 33.97 |
| horse | 50.40 | 48.77 | 47.24 | 50.61 | 50.17 | 39.72 | 43.64 | 41.76 | 44.31 | 43.54 |
| motorbike | 49.84 | 42.67 | 40.31 | 44.82 | 46.09 | 35.68 | 38.05 | 36.93 | 39.27 | 36.24 |
| person | 75.43 | 75.06 | 74.39 | 76.10 | 76.49 | 72.05 | 72.26 | 72.30 | 74.40 | 70.43 |
| pottedplant | 13.85 | 24.77 | 17.78 | 25.16 | 25.67 | 19.81 | 18.47 | 14.80 | 24.16 | 18.43 |
| sheep | 30.44 | 33.88 | 32.48 | 35.82 | 35.74 | 26.85 | 31.21 | 32.63 | 29.38 | 33.22 |
| sofa | 41.70 | 30.04 | 30.27 | 31.45 | 31.05 | 27.45 | 28.88 | 28.71 | 24.65 | 28.96 |
| train | 66.77 | 65.70 | 63.83 | 67.69 | 68.15 | 61.72 | 64.09 | 62.71 | 61.77 | 63.49 |
| tvmonitor | 51.76 | 50.67 | 50.89 | 54.21 | 54.70 | 43.38 | 47.34 | 46.50 | 48.14 | 45.25 |
| **avg** | 46.04 | 44.22 | 43.15 | 45.79 | 46.06 | 40.47 | 41.68 | 41.18 | 41.96 | 41.16 |

Table 4.9: Average precision on the VOC 2009 dataset in percent for the ten base classifiers.

| class | $\max(h_1, \ldots, h_{10})$ | our (linear) | impr. | poly2 | impr. | poly3 | impr. | our (RBF) | impr. |
|---|---|---|---|---|---|---|---|---|---|
| aeroplane | 73.50 | 78.85 | 5.35 | 78.60 | 5.10 | 77.91 | 4.41 | **79.07*** | 5.57 |
| bicycle | 45.17 | 49.30 | 4.13 | **49.91** | 4.74 | 49.47 | 4.30 | 49.62* | 4.45 |
| bird | 48.99 | 53.37 | 4.38 | 54.86* | 5.87 | 54.77 | 5.78 | **56.97** | 7.98 |
| boat | 54.84 | 58.76 | 3.92 | 60.21 | 5.37 | **60.67** | 5.83 | 60.18* | 5.34 |
| bottle | 23.00 | 26.79 | 3.79 | 26.83 | 3.83 | 23.56 | 0.56 | **27.39*** | 4.39 |
| bus | 58.34 | 62.73 | 4.39 | **64.93** | 6.59 | 63.61 | 5.27 | 64.62* | 6.28 |
| car | 50.24 | 51.97 | 1.73 | 52.23 | 1.99 | 51.45 | 1.21 | **53.69*** | 3.45 |
| cat | 47.02 | 50.74 | 3.72 | 52.53 | 5.51 | 51.59 | 4.57 | **52.71*** | 5.69 |
| chair | 47.38 | 49.45 | 2.07 | **50.81** | 3.43 | 50.47 | 3.09 | 50.00* | 2.62 |
| cow | 29.24 | 26.90 | -2.34 | 31.08 | 1.84 | 30.79 | 1.55 | **36.92*** | 7.68 |
| diningtable | 35.92 | 42.48 | 6.56 | 43.19 | 7.27 | **43.44*** | 7.52 | 44.38 | 8.46 |
| dog | 41.14 | 41.08 | -0.06 | **43.12** | 1.98 | 38.55 | -2.59 | 39.76* | -1.38 |
| horse | 50.61 | 52.83 | 2.22 | **55.84*** | 5.23 | 55.35 | 4.74 | 55.67 | 5.06 |
| motorbike | 49.84 | 54.33 | 4.49 | 54.18 | 4.34 | 53.82 | 3.98 | **55.42*** | 5.58 |
| person | 76.49 | 79.20 | 2.71 | 80.31 | 3.82 | 80.30 | 3.81 | **80.41*** | 3.92 |
| pottedplant | 25.67 | 28.93 | 3.26 | **29.72** | 4.05 | 28.71 | 3.04 | 29.16* | 3.49 |
| sheep | 35.82 | 41.75 | 5.93 | **43.63** | 7.81 | 41.74 | 5.92 | 41.32* | 5.50 |
| sofa | 41.70 | 42.04 | 0.34 | 40.48 | -1.22 | 37.99 | -3.71 | **42.44*** | 0.74 |
| train | 68.15 | 74.06 | 5.91 | 73.77* | 5.62 | **74.76** | 6.61 | 73.83 | 5.68 |
| tvmonitor | 54.70 | 59.51 | 4.81 | **58.76*** | 4.06 | 56.99 | 2.29 | 55.96 | 1.26 |
| **avg** | 47.89 | 51.25 | 3.37 | 52.25 | 4.36 | 51.30 | 3.41 | 52.48 | 4.59 |

Table 4.10: Average precision on the VOC 2009 dataset in percent for the best individual classifier and our method with linear, polynomial and RBF kernel. The best method for each class is indicated by a bold entry. Starred values indicate which kernel is chosen by cross-validation.

| class | $\max(h_1, \ldots, h_{10})$ | stacking | impr. | our (Kernel) | impr. |
|---|---|---|---|---|---|
| aeroplane | 73.50 | 75.43 | 1.93 | 79.07 ($r$) | 5.57 |
| bicycle | 45.17 | 46.65 | 1.48 | 49.62 ($r$) | 4.45 |
| bird | 48.99 | 50.72 | 1.73 | 54.86 ($p^2$) | 5.87 |
| boat | 54.84 | 54.84 | 0.00 | 60.18 ($r$) | 5.34 |
| bottle | 23.00 | 17.89 | -5.11 | 27.39 ($r$) | 4.39 |
| bus | 58.34 | 58.21 | -0.13 | 64.62 ($r$) | 6.28 |
| car | 50.24 | 49.22 | -1.02 | 53.69 ($r$) | 3.45 |
| cat | 47.02 | 46.44 | -0.58 | 52.71 ($r$) | 5.69 |
| chair | 47.38 | 45.30 | -2.08 | 50.00 ($r$) | 2.62 |
| cow | 29.24 | 29.17 | -0.07 | 36.92 ($r$) | 7.68 |
| diningtable | 35.92 | 20.97 | -14.95 | 43.44 ($p^3$) | 7.52 |
| dog | 41.14 | 39.81 | -1.33 | 39.76 ($r$) | -1.38 |
| horse | 50.61 | 45.15 | -5.46 | 55.84 ($p^2$) | 5.23 |
| motorbike | 49.84 | 50.54 | 0.70 | 55.42 ($r$) | 5.58 |
| person | 76.49 | 77.39 | 0.90 | 80.41 ($r$) | 3.92 |
| pottedplant | 25.67 | 16.64 | -9.03 | 29.16 ($r$) | 3.49 |
| sheep | 35.82 | 22.10 | -13.72 | 41.32 ($r$) | 5.50 |
| sofa | 41.70 | 36.87 | -4.83 | 42.44 ($r$) | 0.74 |
| train | 68.15 | 72.55 | 4.40 | 73.77 ($p^2$) | 5.62 |
| tvmonitor | 54.70 | 40.67 | -14.03 | 58.76 ($p^2$) | 4.06 |
| **avg** | 47.89 | 44.83 | -3.06 | 52.47 | 4.58 |

Table 4.11: Average precision on the VOC 2009 dataset in percent for the best individual classifier, stacking as well as for our method with kernel selected by cross-validation.

method of combining classifiers for multilabel classification is appealingly simple, it works very well. Our main goal was to investigate the gain of our approach with respect to the base learners and simpler methods for combining features or classifiers.

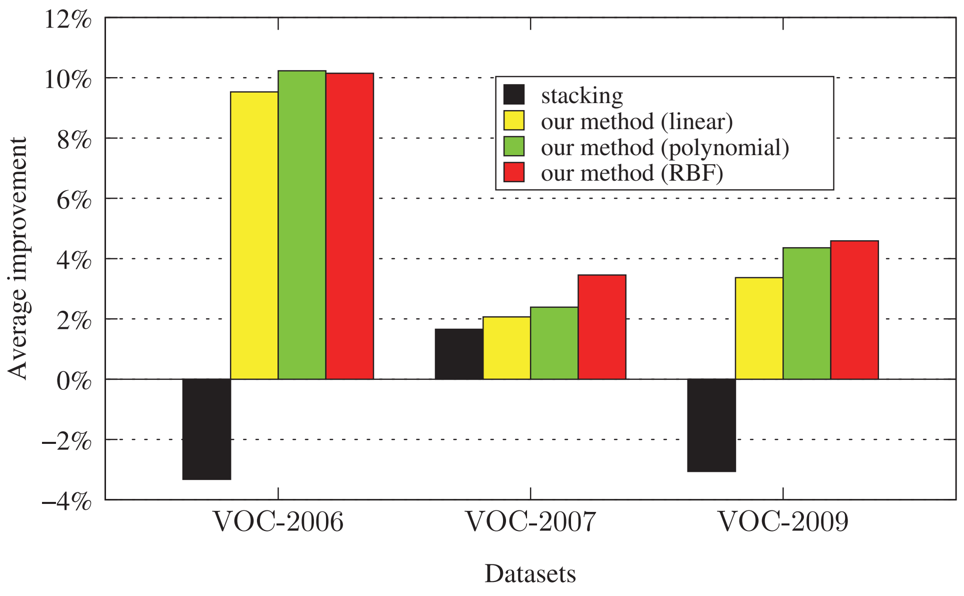


Figure 4.6: Average improvement of the methods on the VOC 2006, VOC 2007, and VOC 2009 datasets in percent.

This page intentionally contains only this sentence.

Chapter 5

# Conclusion

Feedback structures discovered in biological vision systems have long been known to affect the processing of visual input data. However, most computer vision systems lack feedback structures completely. They rely on simple bottom-up processing, from low-level features to high-level descriptions. This thesis proposes a cognitive system which incorporates feedback structures. It does bottom-up and top-down processing depending on the visual input.

It is known that biological vision systems can track objects reliably even if they are occluded. In addition, they can predict possible locations if the objects are totally occluded. There exist various methods for tracking objects. However, most of these tracking methods fail if objects are totally occluded. This thesis offers an architecture of a cognitive vision system which can process videos with highly occluded and totally occluded objects. It builds interpretations of the observed visual data. An evaluation function selects the most likely interpretation.

Can the cognitive vision system incorporate results from various trackers with different low-level features? Such a property would increase the stability of a cognitive vision system. If one tracker fails but the other remaining trackers can track the object, then there will be no influence on the result. This thesis shows in the second part that information from additional detectors can improve results. This was shown for detection of object categories in images. Further research is needed for the case where many low-level vision components track the objects of interest.

What does the cognitive vision system do if additional trackers report different locations for the same object? The system can calculate a mean location. However, this simple approach only works if the locations are close. If there is a large disagreement between the trackers then strategies on how to handle such situations have to be found end evaluated.

In the second part, supervised learning was used to learn the weights for the combination of the detectors. One drawback is that supervised learning needs feedback. What can be used to provide feedback in a cognitive vision system? One possible solution may be to use reinforcement learning with its delayed feedback. A cognitive agent with a vision system uses actions to reach a defined goal. The delayed feedback can be used to update the internal structure of the vision system.

The cognitive vision system in this thesis keeps track of the objects of interest even if they are totally occluded and can give answers where the objects are. The latter information may serves as an input for a higher-level cognitive system which understands actions in videos and infers context, e.g. "Is this man stealing a car from the parking lot?" or "Was my cup in the dishwasher and is it clean now?". Therefore, future work will be needed to find a reliably cognitive vision system for scene understanding.

# Nomenclature

| | |
|---|---|
| $area_{occ}\left(o_j \vert h_i\right)$ | relative occluded area of an object $o_j$ given a hypothesis $h_i$ |
| $conf\left(o_j \vert h_i, I_t\right)$ | confidence value of an object $o_j$ given a hypothesis $h_i$ and image $I_t$ |
| $\mathbf{H}_{aff}$ | $3 \times 3$ affine homography matrix |
| $hist\left(n\right)$ | returns the $n^{th}$ entry of a normalized histogram |
| $H_t$ | all hypotheses of an image at time $t$ |
| $I$ | image matrix |
| $\overline{I}_t^{x,y}$ | average of an image region at center $(x, y)$ and time $t$ |
| $I\left(\mathbf{x}_n\right)$ | returns the $n^{th}$ pixel from a segment of image $I$ at position $\mathbf{x}$. |
| $I_t$ | image matrix at time $t$ |
| $o$ | orientation of an interest-point |
| $p\left(\mathbf{x}\right)$ | probability density function in the feature space at location $\mathbf{x}$ (candidate region) |
| $\hat{\mathbf{p}}\left(\mathbf{x}\right)$ | estimated probability density function of $p\left(\mathbf{x}\right)$ |
| $\hat{p}_u\left(\mathbf{x}\right)$ | histogram bin $u$ of $\hat{\mathbf{p}}\left(\mathbf{x}\right)$ |
| $q$ | probability density function in the feature space (model) |
| $\hat{\mathbf{q}}$ | estimated probability density function of $q$ |
| $\hat{q}_u$ | histogram bin $u$ of $\hat{\mathbf{q}}$ |
| $\mathbf{R}$ | $3 \times 3$ rotation matrix |
| $s$ | scale of an interest-point |
| $T$ | template matrix |
| $T_0$ | original template matrix |
| $\overline{T}_t^{\beta}$ | average of template matrix at time $t$ |
| $T_t$ | template matrix at time $t$ |
| $\mathbf{x}_t$ | vector $\mathbf{x}$ at time $t$ |

This page intentionally contains only this sentence.

# List of Figures

# List of Tables

This page intentionally contains only this sentence.

# Bibliography

[1] Azizi Abdullah, Remco Veltkamp, and Marco Wiering. Spatial pyramids and two-layer stacking SVM classifiers for image categorization: A comparative study. In *International Joint Conference on Neural Networks (IJCNN 2009)*, pages 5–12, 2009.

[2] Shivani Agarwal, Aatif Awan, and Dan Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, 2004.

[3] Thomas S. Anantharaman, Murray Campbell, and Feng hsiung Hsu. Singular extensions: Adding selectivity to brute-force searching. *Artificial Intelligence*, 43(1):99–109, 1990.

[4] Martin Antenreiter. Object recognition using geometric properties. Master's thesis (Diplomarbeit), Graz University of Technology, Austria, 2005.

[5] Martin Antenreiter and Peter Auer. A reasoning system to track movements of totally occluded objects. In *Second International Cognitive Vision Workshop (ICVW06), ECCV'06*, Graz, Austria, May 2006. Published electronically on CD-ROM.

[6] Martin Antenreiter, Christian Savu-Krohn, and Peter Auer. Visual classification of images by learning geometric appearances through boosting. In *Artificial Neural Networks in Pattern Recognition (ANNPR 2006)*, pages 233–243, 2006.

[7] Martin Antenreiter, Johann Prankl, Markus Vincze, and Peter Auer. Using a spatio-temporal reasoning system to improve object models on the fly. In *Visual Learning*, pages 25–36. Oesterreichische Computer Gesellschaft, May 2009. ISBN 978-3-85403-254-0.

[8] John W. Backus, Friedrich L. Bauer, Julien Green, C. Katz, John McCarthy, Alan J. Perlis, Heinz Rutishauser, Klaus Samelson, Bernard Vauquois, Joseph Henry Wegstein, Adriaan van Wijngaarden, Michael Woodger, and Peter Naur. Revised report on the algorithm language algol 60. *Communications of the ACM*, 6(1):1–17, 1963.

[9] Dana H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[10] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc J. Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[11] Brandon Bennett, Derek R. Magee, Anthony G. Cohn, and David C. Hogg. Using spatio-temporal continuity constraints to enhance visual tracking of moving objects. In Ramon López de Mántaras and Lorenza Saitta, editors, *ECAI*, pages 922–926. IOS Press, 2004. ISBN 1-58603-452-9.

[12] Brandon Bennett, Derek R. Magee, Anthony G. Cohn, and David C. Hogg. Enhanced tracking and recognition of moving objects by reasoning about spatio-temporal continuity. *Image and Vision Computing*, 26(1):67–81, 2008. ISSN 0262-8856. Cognitive Vision-Special Issue.

[13] Kristin P. Bennett, Ayhan Demiriz, and John Shawe-Taylor. A column generation algorithm for boosting. In *Proceedings of the 17th International Conference on Machine Learning (COLT 2000)*, pages 65–72. Morgan Kaufmann, 2000.

[14] Irving Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

[15] Bruce G. Buchanan and Edward H. Shortliffe. *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project (The Addison-Wesley series in artificial intelligence)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984. ISBN 0201101726.

[16] Bruce G. Buchanan, Georgia L. Sutherland, and Edward A. Feigenbaum. Heuristic DENDRAL: a program for generating explanatory

hypotheses in organic chemistry. In Bernard Meltzer, Donald Michie, and Michael Swann, editors, *Machine Intelligence 4*, pages 209–254. Edinburgh University Press, Edinburgh, Scotland, 1969.

[17] Murray Campbell, A. Joseph Hoane Jr., and Feng hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1-2):57–83, 2002.

[18] Sheng Chen, Alan Fern, and Sinisa Todorovic. Multi-object tracking via constrained sequential labeling. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1130–1137, 2014.

[19] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[20] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition (CVPR 2000)*, pages 142–149, 2000.

[21] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–575, 2003.

[22] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000. ISBN 0521780195.

[23] James L. Crowley and Alice C. Parker. A representation for shape based on peaks and ridges in the difference of low-pass transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6 (2):156–170, 1984.

[24] Adrianus D. De Groot. *Het denken van den schaker*. PhD thesis, University of Amsterdam, Amsterdam, 1946.

[25] Adrianus D. De Groot. *Thought and choice in chess*. Mouton : The Hague, 1965.

[26] Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46 (1-3):225–254, 2002.

[27] Thomas G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.

[28] Anastasios Dimou, Grigorios Tsoumakas, Vasileios Mezaris, Ioannis Kompatsiaris, and Ioannis Vlahavas. An empirical study of multi-label learning methods for video annotation. In *7th International Workshop on Content-Based Multimedia Indexing (CBMI 2009)*, pages 19–24, 2009.

[29] Sašo Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54(3): 255–273, 2004.

[30] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, .

[31] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html, .

[32] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html, .

[33] Mark Everingham, Andrew Zisserman, Christopher K. I. Williams, and Luc J. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results, .

[34] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[35] Luca Fiaschi, Ferran Diego, Konstantin Gregor, Ullrich Köthe, Marta Zlatic, and Fred A. Hamprecht. Tracking indistinguishable translucent objects over time using weakly supervised structured learning. In

*Computer Vision and Pattern Recognition (CVPR)*, pages 2736–2743, 2014.

[36] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice (2nd Ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990. ISBN 0-201-12110-7.

[37] Yoav Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory (COLT 1990)*, pages 202–216, 1990.

[38] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, Second edition, October 1990. ISBN 0122698517.

[39] Michael Fussenegger, Andreas Opelt, Axel Pinz, and Peter Auer. Object recognition using segmentation for feature detection. In *International Conference on Pattern Recognition (ICPR) (3)*, pages 41–44, 2004.

[40] Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In *Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference (PAKDD 2004)*, pages 22–30. Springer, 2004.

[41] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, Second edition, January 2002. ISBN 0201180758.

[42] Luc J. Van Gool, Theo Moons, and Dorin Ungureanu. Affine/photometric invariants for planar intensity patterns. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision – Volume I*, pages 642–651. Springer, 1996. ISBN 3-540-61122-3.

[43] Helmut Grabner, Jiri Matas, Luc J. Van Gool, and Philippe C. Cattin. Tracking the invisible: Learning where the object might be. In *Computer Vision and Pattern Recognition (CVPR 2010)*, pages 1285–1292. IEEE, 2010.

[44] Gustaf Gredebäck. *Infants' Knowledge of Occluded Objects: Evidence of Early Spatiotemporal Representation*. PhD thesis, Acta Universi-

tatis Upsaliensis; Faculty of Social Sciences, 2004. ISBN 91-554-5898-X.

[45] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346, June 1990. ISSN 01672789.

[46] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, Second edition, 2003. ISBN 0521540518.

[47] Richard I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:580–593, June 1997. ISSN 0162-8828.

[48] John Haugland. Semantic engines: an introduction to mind design. In John Haugland, editor, *Mind Design: Philosophy, Psychology, Artificial Intelligence*, Cambridge, 1982. Bradford Books, MIT Press.

[49] David E. Heckermann. *Probabilistic similarity networks*. MIT Press, Cambridge, MA, USA, 1991. ISBN 0-262-08206-3.

[50] Paul V. C. Hough. Machine analysis of bubble chamber pictures. In *International Conference on High Energy Accelerators and Instrumentation*, pages 554–556, 1959.

[51] ISO/IEC 14977:1996. *Information technology – Syntactic metalanguage – Extended BNF*. International Organization for Standardization, Geneva, Switzerland.

[52] Tommi Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*, pages 487–493. MIT Press, 1998.

[53] Ian T. Jolliffe. *Principal component analysis*. Springer, New York, 1986.

[54] Timor Kadir and Michael Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

[55] Thomas Kailath. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Transactions on Communications Technology*, 15(1):52–60, 1967.

[56] Rudolph E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[57] Yan Ke and Rahul Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition (CVPR 2004)*, pages 506–513. IEEE Computer Society, 2004.

[58] John F. Kihlstrom. The cognitive unconscious. *Science*, 237:1445–1452, 1987.

[59] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[60] Hans-Peter Kriegel, Peer Kröger, Alexey Pryakhin, and Matthias Schubert. Using support vector machines for classifying large sets of multi-represented objects. In *Proceedings of the Fourth SIAM International Conference on Data Mining (SDM 2004)*, pages 102–113. SIAM, 2004.

[61] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3):259–289, 2008.

[62] Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, 2006.

[63] Alex Po Leung and Shaogang Gong. Optimizing distribution-based matching by random subsampling. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE Computer Society, 2007.

[64] John P. Lewis. Fast template matching. *Vision Interface*, pages 120–123, 1995.

[65] Michael J. Lighthill. Artificial intelligence: A general survey. In M. J. Lighthill, N. S. Sutherland, R. M. Needham, H. C. Longuet-Higgins, and D. Michie, editors, *Artificial Intelligence: A Paper Symposium.* Science Research Council of Great Britain, London, 1973.

[66] Jianhua Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory*, 37:145–151, 1991.

[67] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, 1998.

[68] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[69] David G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision (ICCV)*, pages 1150–1157, 1999.

[70] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[71] Ziyang Ma and Enhua Wu. Real-time and robust hand tracking with a single depth camera. *The Visual Computer*, 30(10):1133–1144, October 2014. ISSN 0178-2789.

[72] Emilio Maggio and Andrea Cavallaro. *Video Tracking - Theory and Practice*. Wiley, 2011. ISBN 978-0-470-74964-7.

[73] David Marr. Artificial intelligence - a personal view. *Artificial Intelligence*, 9(1):37–48, 1977.

[74] John Mccarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, pages 463–502. Edinburgh University Press, 1969.

[75] John P. McDermott. R1: A rule-based configurer of computer systems. *Artificial Intelligence*, 19(1):39–88, 1982.

[76] Stephen J. McKenna, Sumer Jabri, Zoran. Duric, Azriel Rosenfeld, and Harry Wechsler. Tracking groups of people. *Computer Vision and Image Understanding*, 80(1):42–56, 2000.

[77] Christopher J. Merz. Using correspondence analysis to combine classifiers. *Machine Learning*, 36(1-2):33–58, 1999.

[78] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision-Part I*, ECCV '02, pages 128–142, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-43745-2.

[79] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[80] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

[81] Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *International Conference on Computer Vision (ICCV)*, pages 525–531, 2001.

[82] Andreas Opelt, Michael Fussenegger, Axel Pinz, and Peter Auer. Generic object recognition with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):416–431, March 2006.

[83] David L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, December 1972.

[84] Patrick Pérez, Carine Hue, Jaco Vermaak, and Michel Gangnet. Color-based probabilistic tracking. In *7th European Conference on Computer Vision*, pages 661–675. Springer, 2002.

[85] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition (CVPR 2007)*, pages 1–8. IEEE Computer Society, 2007. ISBN 1-4244-1179-3.

[86] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 143–156, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15560-X, 978-3-642-15560-4.

[87] Steven Pinker. Visual Cognition: An introduction. In Steven Pinker, editor, *Visual Cognition*, pages 1–64, Cambridge, 1984. MIT Press.

[88] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Peter J. Bartlett, Bernhard Schölkopf, Dale Schuurmans, and Alex J. Smola, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, Boston, 1999.

[89] Johann Prankl, Martin Antenreiter, Peter Auer, and Markus Vincze. Consistent interpretation of image sequences to improve object models on the fly. In Mario Fritz, Bernt Schiele, and Justus H. Piater, editors, *ICVS*, volume 5815 of *Lecture Notes in Computer Science*, pages 384–393. Springer, 2009. ISBN 978-3-642-04666-7.

[90] Prior Design NA. PD Mercedes E Klasse C207, May 2011. URL `http://www.flickr.com/photos/priordesignna/5762730657/`. Shared under the Attribution-NonCommercial-ShareAlike 2.0 Generic (CC BY-NC-SA 2.0) license, Accessed on 29 Jun. 2011.

[91] Andrzej W. Przybyszewski. Vision: does top-down processing help us to see? *Current Biology*, 8(4):R135–R139, 1998.

[92] Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. Correlative multi-label video annotation. In *Proceedings of the 15th International Conference on Multimedia 2007 (MM 2007)*, pages 17–26. ACM, 2007.

[93] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. Real-time and robust hand tracking from depth. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1130–1137, 2014.

[94] Thomas H. Reiss. *Recognizing planar objects using invariant image features*, volume 679 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1993. ISBN 3-540-56713-5.

[95] Ray Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical*

*Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, San Diego, CA, 1991.

[96] Ryan M. Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[97] Kathleen S. Rockland and Genny W. Drash. Collateralized divergent feedback connections that target multiple cortical areas. *The Journal of comparative neurology*, 373(4):529–548, 1996.

[98] Kathleen. S. Rockland and Agnes Virga. Terminal arbors of individual "feedback" axons projecting from area V2 to V1 in the macaque monkey: a study using immunohistochemistry of anterogradely transported Phaseolus vulgaris-leucoagglutinin. *The Journal of comparative neurology*, 285(1):54–72, Jul 1989. ISSN 0021-9967.

[99] Peter J. Rousseeuw. Least median of squares regression. *Journal of The ACM*, 79(388):871–880, December 1984.

[100] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[101] John R. Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3:417–457, 1980.

[102] Jean Serra. *Image analysis and mathematical morphology.* Number 1 in Image Analysis and Mathematical Morphology. Academic Press, 1982. ISBN 978-0-126-37240-3.

[103] Jean Serra. *Image analysis and mathematical morphology.* Number 2 in Image Analysis and Mathematical Morphology. Academic Press, 1988. ISBN 978-0-126-37241-0.

[104] Murray Shanahan. *Solving the frame problem: a mathematical investigation of the common sense law of inertia.* MIT Press, Cambridge, MA, USA, 1997. ISBN 0-262-19384-1.

[105] Claude E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41(314):256–275, 1950.

[106] Roger N. Shepard. *Mind sights: original visual illusions, ambiguities, and other anomalies, with a commentary on the play of mind in perception and art*. W.H. Freeman and Co., New York, NY, USA, 1990. ISBN 9780716721338.

[107] Adam M. Sillito, Helen E. Jones, George L. Gerstein, and David C. West. Feature-linked synchronization of thalamic relay cell firing induced by feedback from the visual cortex. *Nature*, 369(6480):479–482, Jun 1994.

[108] Adam M. Sillito, Kenneth L. Grieve, Helen E. Jones, Javier Cudeiro, and Justin Davls. Visual cortical mechanisms detecting focal orientation discontinuities. *Nature*, 378(6556):492–496, Nov 1995.

[109] Elizabeth S. Spelke and Claes von Hofsten. Predictive reaching for occluded objects by 6-month-old infants. *Journal of Cognition and Development*, 2:261–281, 2001.

[110] Markus Stumptner. An overview of knowledge-based configuration. *AI Communications*, 10(2):111–125, 1997. ISSN 0921-7126.

[111] Cho-Huak Teh and Roland T. Chin. On image analysis by the methods of moments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(4):496–513, 1988. ISSN 0162-8828.

[112] Kai Ming Ting and Ian H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.

[113] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59: 433–460, 1950.

[114] Koen E. A. van de Sande, Theo Gevers, and Cees G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010.

[115] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8.

[116] Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, December 2004. ISBN 0387402721.

[117] Lawrence B. Wolff. On the relative brightness of specular and diffuse reflection. In *Computer Vision and Pattern Recognition (CVPR 1994)*, pages 369–376. IEEE Computer Society, 1994.

[118] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2): 241–259, 1992.

[119] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4), December 2006. ISSN 0360-0300.

*“Evolution does not produce something that is perfect,*
*it just produces something that works.”*
*– Tim Skern*