



Universität Bremen

Fachbereich 3: Mathematik und Informatik

Diploma Thesis

Structure-Oriented Perception Control in the Kitchen Domain driven by Methods of Computational Attention

Daniel Beßler

Matriculation No.212 945 8

14/07/2014

First Examiner: Michael Beetz

Second Examiner: Yohannes Kassahun

Advisor: Thomas Wagner

Daniel Beßler

Structure-Oriented Perception Control in the Kitchen Domain driven by Methods of Computational Attention

Diploma Thesis, Fachbereich 3: Mathematik und Informatik

Universität Bremen, July 2014

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Bremen, den 14/07/2014

Daniel Beßler

Acknowledgements

An dieser Stelle möchte ich einen besonderen Dank an Dr. Thomas Wagner richten für sein hohes Engagement im Rahmen dieser Diplomarbeit. Die regelmäßigen Diskussionen haben diese Arbeit wesentlich mitgeprägt.

Desweiteren möchte ich Prof. Michael Beetz sowie der Arbeitsgruppe des IAI meinen Dank aussprechen für die Bereitstellung der Labor-Küche, echten Daten vom PR2 Roboter sowie der Software für die Wahrnehmung vom PR2. Insbesondere möchte ich Ferenc Balint-Benczedi für seine Ratschläge und dem Expertenwissen für die Wahrnehmungsmethoden danken.

Zudem möchte ich Christoph Landgraf meinen Dank aussprechen für die Bereitstellung der Wissensbasis sowie für seine Bemühungen die Software an meine Bedürfnisse anzupassen.

Contents

1	Introduction	1
2	Motivation and Problem Statement	5
2.1	Problem Statement	8
2.1.1	Solution Space in Kitchens	9
2.1.2	Limited Computational Resources	10
2.1.3	Uncertainty in the Belief State	11
2.1.4	Control Knowledge Acquisition and Maintenance	11
2.1.5	Adaptability to different Household Activities	12
2.2	Application Domain	12
2.2.1	Kitchen Environment	13
2.2.2	Robot Platform	14
2.2.3	RoboSherlock	14
2.2.4	Household Activity	16
2.3	Reference Scenarios	17
2.4	Objective and Requirements	20
3	Top-Down Attention Approaches	23
3.1	Cognitive Models	25
3.1.1	Basic Theories and Architectures of Attention	27
3.1.2	\mathcal{SNR} Maximization	30
3.1.3	Euclidean Distance Minimization	33
3.1.4	Relational Task Graph	35
3.2	Probabilistic Models	37
3.2.1	Gist Projection Matrix	39
3.2.2	Feature Prior for Target Classification	41
3.2.3	Mutual Information of Evidences and Target Class	43
3.2.4	Contextual Priors for High-Level Object Features	45
3.3	Conclusion and Discussion	48
4	Perception Control in the Kitchen Domain	51

4.1	Kitchen Domain Ontology for Perception Procedures	53
4.1.1	Ontology Formalism	53
4.1.2	Ontology for Perception in Kitchens	55
4.2	The Perception Control Loop	58
4.3	Ontology-Based Task Representation	60
4.4	Control Knowledge for Perception Procedures	62
4.4.1	Perception Control Agenda	63
4.4.1.1	Agenda Focus	65
4.4.2	Selection of Perception Methods and Recognized Objects	65
4.4.2.1	Relational Task Distance	67
4.4.2.2	Edit Distance between Model Instances	71
4.4.2.3	Similarity to Task Entities	72
4.4.3	Invocation of Perception Methods	75
4.5	Summary and Discussion	80
5	Implementation Details	81
5.1	Modular Architecture	82
5.1.1	Open Service Gateway initiative (OSGi)	83
5.1.2	Unstructured Information Management Architecture (UIMA)	84
5.1.3	Control Framework Services	87
5.2	Control Rules	88
5.2.1	Drools Control Rules	90
5.3	Acquisition of Control Knowledge	92
5.4	Application of Control Knowledge	94
6	Empirical Investigations	97
6.1	Experiment Modalities	98
6.1.1	Validation Methods	98
6.1.2	Experiment Procedure	101
6.1.3	System Parameters	103
6.1.4	Acquisition of Training Data	107
6.2	Classification Experiments	108
6.2.1	Experiment A1 - Goggles Annotation	108
6.2.2	Experiment A2 - LINE-MOD Annotation	111
6.2.3	Experiment A3 - Color Ratio Annotation	113
6.2.4	Experiment A4 - Bayes' Classifier	115
6.2.5	Classification Parameters	116
6.3	Configuration Experiments	117
6.3.1	Experiment B1 - Edit Distance Weight	117
6.3.2	Experiment B2 - Minimum Task Relation Similarity	119

6.3.3	Experiment B3 - Prioritization of Annotations	121
6.3.4	System Parameters	123
6.4	Control Experiments	124
6.4.1	Experiment C1 - The Default Scenario	124
6.5	Summary and Discussion	126
7	Conclusion and Perspective	127
A	Appendix	131
A.1	List of Figures	131
A.2	Bibliography	134
A.3	List of Abbreviations	137
B	Empirical Results	139
B.1	Experiment A1	139
B.2	Experiment A2	142
B.3	Experiment A3	145
B.4	Experiment A4	148
B.5	Experiment B1	151
B.6	Experiment B2	152
B.7	Experiment B3	154
B.8	Experiment C1	157
C	CD Appendix	159

Chapter 1

Introduction

Attention is a cognitive process that allows humans to concentrate selectively on regions, objects and features of interest out of a multitude of possible targets. All magicians know how to manipulate the attention of their audience in order to hide their tricks on stage. A bright spotlight that illuminates only a part of the scene, a pretty assistant wearing a conspicuous dress, a misleading comment or a suspicious movement distracts the audience from the trick by tempting them to concentrate on irrelevant regions of the stage.

In psychophysics, human visual attention is often investigated by so called cueing experiments. In these experiments, cues are given to human subjects in order to investigate their effect on human visual attention. For example, the “Gorilla-Experiment” [SC99] is a well-known selective attention experiment. In this experiment, a video of six people playing basketball in a hallway is shown to human subjects. Three people wear white shirts and three people wear black shirts. Human subjects have to count the number of passes made by people in white shirts. About 50% of all subjects did not notice a gorilla strolling through the video. This experiment shows that humans concentrate on objects of interest based on goals and expectations while they may miss other aspects of the scene. Even significant changes, like the appearance of a gorilla in a sports scene, may remain unnoticed. The focus of attention is one of the mechanisms that makes it possible for humans to accomplish complex activities without noticing and understanding everything around them.

In this thesis, knowledge-based (top-down) attention is investigated in the context of mobile robotics in the kitchen domain. Household robots have to accomplish everyday activities such as the preparation of breakfasts. The preparation of breakfasts requires that household robots find a set of items in the kitchen, carry items to the kitchen table as well it requires that household robots arrange items on the table. For example, there could be a particular household where a specific red cup is always used during breakfast. It is (usually) required to open drawers and doors of cabinets in order to search for kitchen items. In unknown kitchens, it is hard to find the location of the wanted item with only one try because the layout of kitchens differs significantly between different households and individual habits. Thus, knowledge about the household can

reduce the problem space in this scenario by focusing on likely locations of the wanted item. For each investigated location, the robot has to estimate if a red cup is visible or not. Knowledge about the appearance of cups as well as the knowledge that the wanted cup is red can be used to focus the attention on visible regions, objects and features that might belong to the red cup.

The major contributions of this thesis are listed and explained to some extent in the following:

- *Structure-Oriented Perception Control*

Structure-oriented perception control is based on knowledge about the current task, objects and perception methods that can be used to guide (control) the perception process. The major purpose of the perception control mechanism is to reduce the search space of the perception procedure based on attention mechanisms. Additionally, the perception control framework is flexible: It allows the adaptation to other domains than kitchens and to different household tasks without modification of the source code. The adaptability is supported by a user interface that is dedicated to the acquisition of control knowledge (i.e., knowledge that is used to guide the perception of the robot). The problem domain (i.e., kitchens, kitchen items, household activities and perception methods) is modeled hierarchically using an ontology (the knowledge base). This knowledge base is used to infer different operations that can be performed in particular situations (structure-oriented). For example, a set of features for recognized objects is declared in the ontology. Each feature corresponds to a perception method that must be executed in order to obtain a value for the feature.

- *Modular Framework for Attention Methods*

In the perception control framework, attention is modeled as a prioritization of the operations that can be performed in a particular situation. The set of possible operations is inferred from the knowledge base and it depends on the representation of the domain knowledge (e.g., the set of features of recognized objects). Different attention methods can be combined in the proposed framework using a prioritized sequence of attention methods. Each attention method represents preferences for particular operations, objects and features.

- *Attention-based on Relations between Objects*

In this framework, it is possible to specify relations between items that are relevant for a particular activity. For example, it is possible to specify the location of wanted items for visual search tasks. This task knowledge can be used to focus the attention to a particular location in the kitchen. The location of items is modeled using spatial relations in the knowledge base that is used by the perception control framework. The proposed attention method is able to utilize knowledge about relations of objects in order to prefer objects and features with relations to the current activity.

This thesis is organized as follows: The kitchen domain, the robot platform, perception methods and household activities are described in chapter 2 along with the problem class of this application domain as well as objectives and requirements. In chapter 3, different methods of top-down attention are discussed with respect to the suitability for the application domain of this thesis. The proposed structure-oriented perception control framework is formally defined and discussed in more detail in chapter 4 and implementation details are described in chapter 5. Finally, the framework is validated in chapter 6 based on varying system parameters and application scenarios.

Motivation and Problem Statement

Visual search is an everyday problem with interesting characteristics for visual attention. For visual search tasks, a target description is known in advance and the subject has to decide whether there is an instance of the target visible or not. For example, finding known persons in the audience of a concert is such a search task. There could be hundreds of people in the crowd, far too many to look at each one in detail instantly. Nevertheless, humans can quickly find known persons in crowds based on their goals and expectations (bounded visual search). For instance, if it is known in advance that the searched person wears a red shirt, then any region in the scene with a dominant red color is likely to attract attention while details of regions with a different dominant color are more likely to remain unnoticed.

In psychophysical experiments, the efficiency of visual search is often measured by the reaction time of human subjects with varying number of objects that differ from the target (distractors). Tsotsos [Tso89] proved that bounded visual search has a time complexity linear in the number of distractors while unbounded visual search is a problem that can't be solved in acceptable time in practice (the problem is NP-complete). Thus, the efficiency of the search procedure highly depends on the knowledge about the target and the task. Efficient search is often accompanied by so called pop-outs [Wol94]. For instance, visually searching for a unique red letter under a set of black letters can be done in constant time by humans regardless of the number of black letters. The unique red color of the letter reflexively engages the attention (pop-out effect). A visual search process is called parallel if the time complexity is constant in the number of distractors, otherwise the process is called sequential. The difference between parallel and sequential search is illustrated in figure 2.1.

Attention can be influenced by data-driven (bottom-up) as well as model-driven (top-down) factors [DD95]. The most salient regions in a scene attract attention regardless of prior knowledge. For example, the unexpected loud noise of a glass shattering on a floor or the sudden motion of a knife falling from a table attracts attention in a bottom-up way. Top-down attention is based on knowledge, expectations and current goals [CS02]. For instance, an expert cook who regularly prepares vegetables is more likely to notice bad quality of vegetables in a shop than an

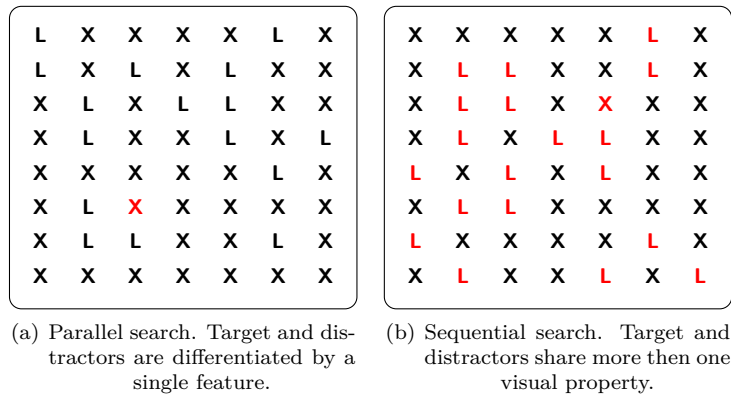


Figure 2.1 Visual search efficiency: It is more efficient to search for a letter that visually pops out by the color (i.e., the red “X” on the left figure).

inexperienced cook. The expert cook has experience with vegetables and knows the appearance of bad vegetables and where to look for bad spots. Both factors are important for everyday life. The degree of importance depends on the current activity and environment. For some activities, like driving a car, bottom-up attention is essential for survival. Car drivers have to react quickly on unexpected events like a suddenly appearing obstacle. Suspicious velocity and shape of objects on the road attract the attention of car drivers in a bottom-up way. Top-down factors are important too. For example, car drivers should know that it might be unnecessary to observe the sky and that it is essential for survival to observe the road, sidewalk and the street signs.

The high amount of environmental stimuli humans can perceive at every moment is in general too high to be completely processed in detail. This problem also occurs in many technical domains with real-time requirements like computer vision, cognitive systems or mobile robotics. Autonomous robots perceive environmental stimuli through artificial sensors like cameras or depth sensors. Computational resources for processing, analyzing and understanding the large amount of sensory data are limited and therefore mechanisms for the selection of the most interesting regions, objects and features in a scene are desirable. Computational attention systems provide such a mechanism that is inspired by psychological and neurobiological models of the human cognition.

Attention applications in the robotic domain can roughly be distinguished in three categories [FRC10]. The first category aims at finding the most salient Region of Interest (ROI) based on low-level features for tasks like matching images with models in a database, segmentation of images based on a similarity criterion or compression of images with stronger compression in none focused regions. The mid-level category acts as a front-end for high-level tasks like object recognition. Object recognition is the task of finding and identifying objects in a scene based on unstructured data that was gathered from sensors. This task is often subdivided in three steps. In the first step, the scene is segmented into several clusters based on low-level features of the



(a) 3D reconstruction of the scene based on a RGB-D sensor. (b) Labeled scene. Each color corresponds to a different object category.

Figure 2.2 Labeling results for a household scene.

Source: <http://ai.cs.washington.edu/projects/rgbd-object-recognition-and-detection>

sensory signal (e.g., the perceived depth). In the second step, individual features of clusters (e.g., shape, color) are computed by perception methods in order to match the annotated clusters with models (labels) known to the system in the last step. This labeling process is illustrated in figure 2.2. The highest-level category of computational attention systems aims at guiding the actions of autonomous robots in a human-like way for tasks like object manipulation, robot navigation or human-robot interaction.

In this thesis, knowledge-based (top-down) attention is investigated as a front-end for object recognition. The attention and object recognition methods are integrated into an expert system. Expert systems are used to solve complex problems where it is required to make decisions based on knowledge. They consist of a knowledge base (information about things, their properties and relations between them) and an inference engine (reasoning about knowledge). Additionally, many expert systems include a control component. The control component is responsible for the focusing on parts of the problem (e.g., ignore the sky while driving), the ordering of actions (e.g., focus on color when searching for a red object), the selection and configuration of methods to perform the actions (i.e., the selection and configuration of perception methods) as well as it is responsible for the handling of conflict situations (e.g., contradictory perception method outcomes).

The objective of this thesis is to implement an expert system control framework that uses top-down attention methods in order to control the perception procedure of an autonomous robot that performs everyday household activities. The control component of expert systems has to deal with problems in the areas of acquisition, maintenance, reasoning, adaptability, consistency and modularity [Gün92]. Furthermore, households are difficult environments because they are complex, dynamic and knowledge about households and household activities may be uncertain. These problems are discussed in section 2.1. The application domain of this thesis includes the software that is used for the perception of the robot (*RoboSherlock*) as well as the Personal

Robot 2 (PR2). *RoboSherlock* provides a set of perception methods that can be used in the kitchen domain for the PR2. The application domain is introduced in section 2.2. In section 2.3, the reference scenarios for the control framework are specified. Reference scenarios represent particular situations in the application domain. They are used for the validation of the control framework (see section 6). Finally, goals and requirements for the proposed framework are described in section 2.4.

2.1 Problem Statement

It is a challenging problem to use autonomous robots in real world environments like households. Households are relatively good observable but robots might not be able to perceive everything around them. For example, most robots have limited field of view and cannot see what is going on behind them. In addition, household items may occlude a part of the scene as seen from the perspective of the robot. Furthermore, households are stochastic because residents, pets or ongoing processes can change the state of the environment and interfere with the current activity of the robot. But the number of actors is relatively low in households compared to other domains like the driving of cars on highways.

Everyday life involves actions in the kitchen including cooking of meals, setting of tables and cleaning of dishes. We all have knowledge about kitchens, items that might occur in kitchens and tasks that are performed in kitchens based on our everyday life. In the context of this thesis, expert knowledge about the kitchen domain is important in order to be able to acquire the knowledge that is used guide the perception of the PR2 (control knowledge).

Top-down attention systems are based on (uncertain) knowledge that is acquired by experts or learning algorithms. Depending on the method of knowledge acquisition and the domain coverage of knowledge different problems might occur. For this reason, a rough subdivision of control knowledge is given below:

- *Task Descriptions*: Task descriptions may contain direct cues of regions, objects or features that are particularly relevant for the task. For instance, the description could contain the cue that the robot looks for a red, cup-shaped object on a table. Furthermore, task descriptions restrict the domain to a particular activity.
- *Common Sense*: Universal knowledge about the application domain that is acquired by experts or learning algorithms. For example, knowledge about the average size of refrigerators as well as the knowledge that crisper for vegetables can be part of refrigerators belongs to this category.
- *Experiences*: Expert knowledge about the application domain that is acquired by learning from a set of training samples. For example, knowledge about the common appearance of

particular kitchen items such as milk packages belongs to this category.

- *Instructions*: Domain specific knowledge acquired by experts. For instance, knowledge about the robot platform belongs to this category.

Acquisition of control knowledge is part of the control framework. The acquisition and maintenance of control knowledge is usually supported by an user interface (the expert system shell) that is dedicated to the acquisition (and application) of control knowledge. The representation of control knowledge in the user interface is an important factor for the acquisition and maintainability of control knowledge. Problems that may occur for expert systems include the modularity of the expert system, the adaptability to other scenarios and domains as well as the explicability and adequacy of control decisions [Gün92]. For example, it is desirable that (generic) parts of the control knowledge can be used for more than one tasks. Different parts of the control knowledge should be independent from each other (modular) in order to support re-usability and maintainability of the control knowledge.

In the following sections, the most relevant problems are discussed in more detail. These problems are subdivided with respect to complexity and versatility of households (discussed in section 2.1.1), limited computational resources (discussed in section 2.1.2) and uncertainty of knowledge (discussed in section 2.1.3).

2.1.1 Solution Space in Kitchens

The household domain requires methods that perform robust with regard to the complexity and versatility of households and household items. The equipment of households depends on various factors like individual preferences, cultural habits, regional habits, availability and budget. For instance, kitchens in Japan are often equipped with an electric rice cooker. In Europe, rice is usually cooked in a pot instead. In addition, the arrangement of furniture, appliances and household items may vary between different kitchens and points in time.

For household activities, one of the most challenging variability in environmental conditions is the illumination of the scene. Illumination influences the appearance of the scene in terms of color and brightness of materials and it is usually not constant within a single room. The brightness of light has influence on perceived material colors too. For instance, it is inadvisable for humans to perform household tasks in complete darkness because it is hard to identify objects without the ability to distinguish them by color. Rooms in households can be equipped with multiple light sources with different colors and intensities. Furthermore, light emitted by local sources usually extends within a few meters. Therefore, local light sources can result in complex color and brightness patterns on appliances, furniture and items in household scenes.

Manifestations of appliances, furniture and household items may vary in material, size, shape and functionality (intra-class variability). For instance, there is a high variability in manifestations

of tables in kitchens. Most frequently wood tables with characteristic wood patterns or plastic tables in arbitrary colors (often plain white) are used. The shape of the shelf space is usually a simple geometric primitive like a rectangle or a circle. The size of tables primary depends on the number of intended seats because each seat requires private space on the table dedicated to the seat. Furthermore, tables may have special functionality like extendible elements for increasing the shelf area.

For the control framework, the complexity of households leads to a large solution space: Each possible specialization (e.g., finding out the color of an object) that is possible in a particular situation belongs to the solution space of the problem (e.g., the set of all colors is the solution space for the parametrization of object colors).

2.1.2 Limited Computational Resources

The household environment is a real world environment that is stochastic and dynamic. It is desirable that robots can react in real-time on environmental changes such as suddenly appearing obstacles. Real-time responses are only possible using sensors with real-time frame rates. The amount of sensory data at real-time frame rates is often too high to be processed in detail. The dimensionality of a sensory signal depends on the measured physical quantity and on the resolution of the sensor. In the simplest case a sensor could represent the state of a switch with a single bit. More complex sensors, like cameras, require much more memory for representing the perceived state. For example, a camera with a resolution of 512x512 pixels where each pixel is represented by 24 bits (1 byte for each color channel) requires 768 kB memory for representing an uncompressed frame. Seen as a camera, without varying resolution in the field of vision, the human eye would be able to vision about 576 million pixels [Cla05]. A camera representing as many pixels would need roughly 1.6 GB of data to encode a single uncompressed frame. Sensor resolution has influence on the computation time. Lower resolution leads to less computation time because less data has to be processed. On the other hand, important details, small and faraway objects could be missed if the sensor resolution is too low. Therefore, the resolution of the sensors should be high enough to perceive details of the environment relevant to the current activity while being low enough to be computational tractable in real-time.

The perception module is only one of many software components running on a mobile robot and the main memory, the hard drive memory, the clock speed of processors and the number of processors and cores is limited. For instance, the PR2 has two processors with four cores per processor. This is a lot compared to modern personal computers but operations performed by these processors can't be arbitrary complex.

2.1.3 Uncertainty in the Belief State

Mobile robots that act in kitchens have to maintain a virtual model of the environment (called belief state) based on the history of perceptions and the sequence of previously performed actions. Kitchens are stochastic and effects of actions could be predicted wrong by the robot. Thus, it is required to continuously update the belief state in order to handle these unpredictable state changes. The belief state is only an estimation of the real world state. Robots can usually only perceive a fraction of a kitchen scene because they have limited field of view and because items could be hidden in containers like drawers. Everything that was perceived by the robot in the past may have changed since the last time it was in the field of view of the robot. Furthermore, the perception module has to estimate the state of the environment based on unstructured data gathered from sensors. This estimation is aggravated by hardware limitations of the robot (e.g., sensor noise, sensor resolution), high versatility of households (e.g., intra-class variability, ignorance), the requirement to continuously update the belief state and uncertainty in general knowledge. The control framework uses this uncertain knowledge about the current state of the environment in order to make control decisions.

Control knowledge acquired by learning is used to make predictions in unknown environments. Insufficient representation of the complexity and versatility of the domain in training samples can result in wrong forecasts. The household domain is rather complex and the gathering of ground truth training samples is sophisticated in real world environments. Additionally, the state of the environment is continuous and discretization of continuous features might be required in order to do reasoning with them. For instance, the discretization of object positions is relevant for finding statistical correlations between different objects and their positions. The absolute position of objects is not very expressive; it restricts correlations to a specific point in space. A more expressive approach is to discretize positions by finding spatial relations to other objects in the scene. For instance, silverware might be located next to dishes on many kitchen tables. In the control framework, such knowledge about relationships between objects could be used to infer likely locations of target objects.

2.1.4 Control Knowledge Acquisition and Maintenance

The acquisition and maintenance of control knowledge is an essential aspect of expert systems that is done by so called knowledge engineers. Knowledge engineers must have expert knowledge in the problem domain. In the kitchen domain, we all are experts. It is desirable that control knowledge can be acquired without deeper knowledge about the control framework. Dependencies between aspects of the control knowledge may lead to inconsistencies in the control knowledge (i.e., contradictions which may lead to the insolvability of the current task) and they



Figure 2.3 PR2 looking at household items.

Source: <http://pr2-looking-at-things.com>

may lead to problems for the maintenance of control knowledge (i.e., high complexity due to dependencies).

2.1.5 Adaptability to different Household Activities

The proposed control framework is intended to be a re-usable framework for various perception tasks in households. Thus, it is required that the framework is adaptable to different scenarios without modification of the source code. Furthermore, particular scenarios should be adaptable to similar scenarios without the need to rewrite the entire control knowledge. For the reusability of control knowledge, it is important that different aspects of the knowledge are independent from each other (modularity). For example, a specific perception method could be needed in order to recognize a spatula object. This knowledge can be applied to various household tasks, which involve a spatula (independent from other control knowledge that is involved).

2.2 Application Domain

The application domain of this thesis can be divided as follows: The kitchen environment, the robot platform, the perception software (*RoboSherlock*) and the household activity. In this



(a) PR2 looking in the fridge.



(b) PR2 looking on the kitchen table.

Figure 2.4 The field of view of PR2 in the reference kitchen.

thesis, the proposed control framework is evaluated using a PR2 that performs everyday tasks in a reference kitchen. A discussion of the kitchen environment, the PR2, the *RoboSherlock* software and kitchen activities is given in following sections.

2.2.1 Kitchen Environment

Households are complex and versatile environments (as discussed in section 2.1.1). Equipment of households and arrangement of household items are influenced by factors such as individual preferences, cultural factors and regional factors. Kitchens are primary used for cooking, preparing of food, dish washing and dining but are also commonly used for other household activities including doing laundry and watching television. These tasks primary differ in actions needed to accomplish the task and in the set of household items that might be involved. For example, preparing breakfasts involves various appliances, silverware, dinnerware, tools and ingredients that may be used during breakfast.

In this thesis, the household domain is restricted to a particular reference kitchen. The reference kitchen is equipped with a table, a counter top, a refrigerator, a stove, a microwave oven, a dishwasher, a sink with hot and cold water and cabinets for silverware, dinnerware, tools and ingredients.

Silverware, dinnerware, tools and ingredients involved widely differ between cultures and regions. For example, a traditional French breakfast primary includes sweet ingredients like strawberry jam or cereals. Contrary, a German breakfast usually includes savory ingredients like sliced cold meats. Typical silverware, dinnerware and tools on breakfast tables in Europe include

various types of forks, spoons, knives, dishes, bowls, glasses, cups and cases for ingredients. The equipment of the reference kitchen reflects the capabilities of the PR2 (see section 2.2.2). The set of kitchen items, which are used in this thesis, includes a pancake maker, a pancake tube, ice tea packages, milk packages, corn flakes packages, ketchup bottles, cups and bowls. Figure 2.3 shows the selection of kitchen items and the point of view of the PR2 in the reference kitchen can be seen on figure 2.4.

2.2.2 Robot Platform

A mobile robot has to be suitable equipped in order to be able to perform everyday household tasks. The actuators of the robot must be accurate enough to grasp and carry household items and the sensor resolution must be high enough to perceive household items, which are required for mastering everyday household activities. In the scope of this thesis, the perception control framework is evaluated with a PR2 (shown in figure 2.3). PR2 is a robotics research and development platform developed by Willow Garage ¹. The robot has a size and a radius of action similar to humans and the platform includes two 7-DOF arms and 1-DOF parallel-jaw grippers, an omnidirectional mobile base, two quad-core i7 Xeon processors, 24 GB main memory, 1.5 TB hard drive memory and a sensor suite including a depth camera, a 5 megapixel camera, an inertial measurement unit (IMU), forearm cameras and gripper tip sensors.

The PR2 can carry items up to a weight of 1.8 kg. Big ingredient packages, large stacks of dishes or chairs may be too heavy for the PR2 but most household items are light enough to be carried. Moreover, objects can be too flat, too small or too large for the grippers due to mechanical restrictions. These objects must be equipped with an additional handle that fits with the size of the grippers in order for the robot to be able to grasp them.

2.2.3 RoboSherlock

The perception control framework uses an existing perception component that was used before for the PR2 in the reference kitchen. The perception component is called *RoboSherlock*. *RoboSherlock* consists of a set of methods dedicated to Unstructured Information Management (UIM) in the context of robotic perception. In UIM systems, pieces of structured information are isolated from the unstructured input data (i.e., sensory input) and a set of perception methods is used in order to obtain information about the structured pieces. Furthermore, UIM systems use a type system for the structured information pieces. In the scope of this thesis, the UIM types can be mapped to the knowledge base that is used in the control procedure (i.e., the knowledge that

¹<http://www.willowgarage.com>

is represented by UIM types is also represented in the control knowledge base). This allows a seamless integration of *RoboSherlock* with the control framework.

The set of perception methods in *RoboSherlock* is as follows:

- *Color*
Computes the dominant colors of an object (a color sequence sorted by dominance) based on the color distribution in HSV (Hue-Saturation-Value) color space. Each computed color can be one of *blue*, *red*, *black*, *green*, *yellow* or *white*.
- *Size*
The *Size* annotation roughly divides objects into *small* and *big* objects based on the maximum distance between points, which belong to the object. The distance is normalized with respect to the distance to the camera (i.e., invariant for the visible scale of objects).
- *Goggles*
*Google Goggles*² is a web service that can be used to classify pictures as well it can be used to find out text that is displayed on pictures. The *Goggles* method is able to send a particular ROI to this web service in order to annotate recognized objects with the result of the service.
- *FlatObject*
Clustering of objects is done based on point clouds. Point clouds are 3D reconstructions of the perceived depth. Flat objects with low variance in depth may remain unrecognized by the 3D clustering algorithm. The *FlatObject* method is used to find additional objects in color space (i.e., based on color contrast).
- *PrimShape*
The *PrimShape* is used to roughly classify the shape of objects into *box* and *round*. This is done based on fitting primitive geometry (lines and circles) to the region that corresponds to a particular cluster.
- *LINE-MOD*
Classification of objects can be done using the *LINE-MOD* [Hin+13] method (i.e., finding out the object type). The *LINE-MOD* method requires a probabilistic model for kitchen items. The model is used to estimate how likely it is that a particular region in the scene corresponds to a particular object type.
- *Location*
The *Location* annotation maps positions of objects to semantic locations in the reference

²<http://www.google.com/mobile/goggles>

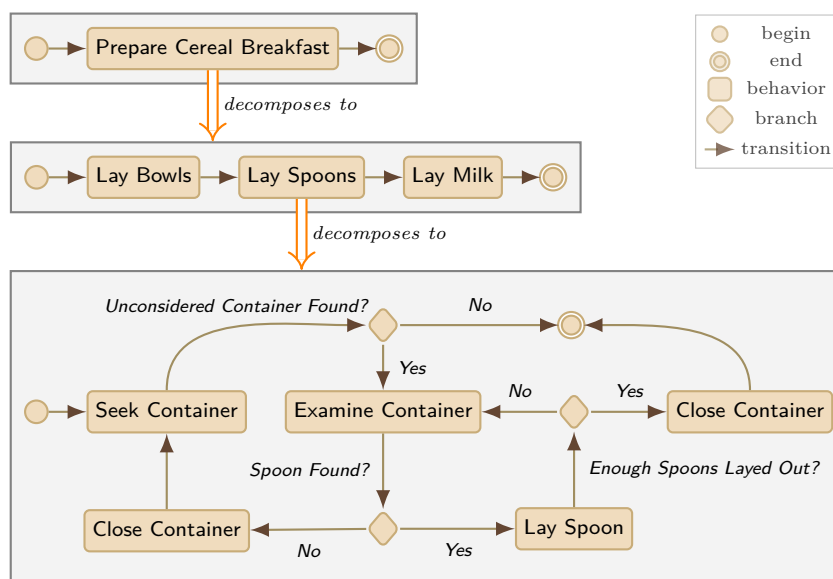
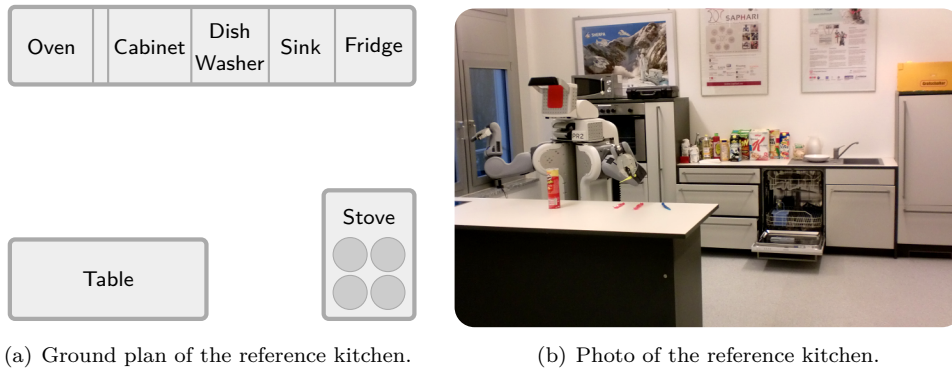


Figure 2.5 Three steps in the decomposition of the task to lay a table for a cereal breakfast.

kitchen. The set of semantic locations in the reference kitchen includes the counter top, the kitchen table, the fridge and drawers of the cabinet. A bounding box is associated to semantic locations. It is used to filter all visible points that do not belong to the semantic location (e.g., the filter keeps items on the table only if the kitchen table is the semantic location of interest).

2.2.4 Household Activity

Many everyday activities can be performed in households. Setting breakfast tables is one of them. In order to set a breakfast table, the robot has to be able to navigate in the kitchen, to grasp, carry and place household items and to visually search for appliances, furniture, free spots and household items. It's essential to decompose the activity into several sub-activities in order to break down high-level tasks like the preparation of breakfast tables to a sequence of actions that can be performed by the robot. Such an incremental decomposition is called hierarchical plan. The decomposition process is illustrated in figure 2.5. One approach for breakfasts is to decompose by the category of household items on the first level of the plan. For instance, it may make sense to put dinnerware on the table before silverware because it's easier to find meaningful positions for the silverware when dishes are already placed on the table. A lower level of the plan includes tasks such as navigation to a cupboard, grasping of one item from the cupboard based on a selection criterion, navigation to the breakfast table and placing of carried items on the



(a) Ground plan of the reference kitchen.

(b) Photo of the reference kitchen.

Figure 2.6 The reference kitchen that is used in all reference scenarios.

table. The primary use of the perception module for this activity is to visually search for objects based on a selection criterion and to observe regions of the scene, which are most relevant to the hierarchical plan. For instance, the attention should be focused on the trajectory of the robot and on objects that may collide with the robot during navigation.

2.3 Reference Scenarios

In order to show properties of the perception control framework, a set of reference scenarios is defined in this section. Each scenario defines one explicit situation in the application domain (see section 2.2) including occurring furniture, appliances, silverware, dinnerware, tools and ingredients as well as the activity of the PR2 and the state of the environment.

In the kitchen domain, there are essentially three different activities of the robot with different difficulties for the perception procedure: Object manipulation, navigation to a particular target location and visual search. Object manipulation enables the robot to open and close cabinet doors and drawers, to grasp household items and to place household items on the kitchen table or the shelf space. Control knowledge can be used to guide the attentional focus during object manipulation. The manipulated object is known in advance. The region between robot and object is particular relevant during manipulation because the actuators of the robot could collide with other objects. Furthermore, the manipulated object itself is particular relevant. It is recognized in advance but the manipulation of the object can fail. This can yield in unpredictable changes in object pose, which should not remain unnoticed by the perception module. Knowledge can also be used to guide the attentional focus during navigation tasks. The target location is known during navigation. The region between robot and target location is particular relevant because the robot could collide with obstacles on the floor or moving objects. Furthermore, knowledge

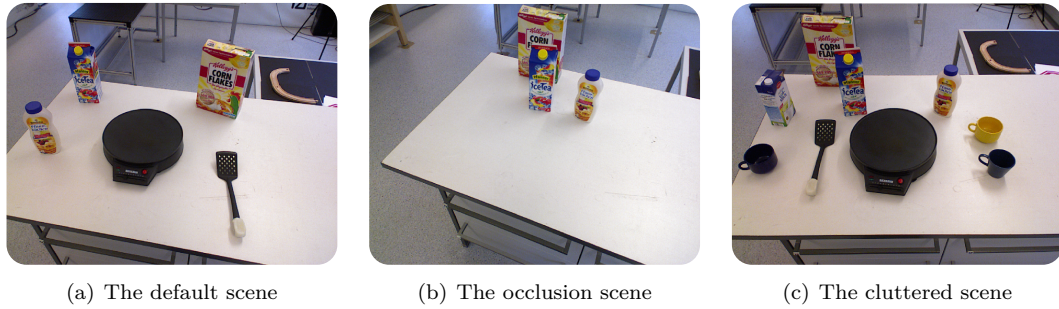


Figure 2.7 The reference scenes that are used for the validation of the perception control framework.

about prospective tasks and target objects could be used during navigation in order to analyze scene regions and objects, which might be task relevant in the near future. Nevertheless, visual search is the most interesting activity for attention mechanisms and navigation and manipulation tasks are not investigated any further in the scope of this thesis. For visual search, there are two different scenarios that can occur when the robot visually searches for a kitchen item: The presence of an object that satisfies the search query or the absence of such an object.

Other difficulties for the perception control framework occur due to difficult spatial configurations of visible objects. Objects can occlude each other as seen from the point of view of the PR2. The classification of occluded objects is difficult and perception methods may not work correctly. Another difficulty for perception methods occurs when visible objects are too close to each other. The point cloud clustering procedure does not work correctly in this case but image based clustering can recognize both objects if they differentiate in color. Furthermore, small and flat objects are difficult for the perception methods. The recognition of such objects requires special perception methods and a control framework that is able to select the specialized methods if required. Finally, a huge amount of visible objects leads to a huge solution space for the perception control framework. This problem is subject of the control framework: The control framework uses control knowledge to guide the perception of the PR2 using top-down attention methods and other domain independent methods.

For each reference scenario, the same kitchen is used. The layout of this reference kitchen is shown in figure 2.6. It is assumed that the robot knows this layout in advance and that the perception control framework has access to this knowledge. Therefore, no identification of appliances is needed for the reference scenarios. But the location of household items is not known in advance. Ingredients, silverware and dinnerware can be located on the kitchen table, in the cabinet, under the sink or on the shelf space on top of the cabinet. Each of the reference scenarios defines a particular visual search task and the scene where the search task is performed. The set of different scenes, which are used for the reference scenarios is shown in figure 2.7.

A *Searching for a Cornflakes Package in the Default Scene*

This scenario represents the most basic situation that can occur for visual search tasks. The PR2 stands in front of the kitchen table and looks onto the shelf space of the table. Multiple objects are located on the shelf space. All objects are previously unknown and clearly visible (i.e., with sufficient gap between them) from the point of view of the PR2. The search target is an arbitrary cornflakes package (i.e., no special brand or variant) that is present on the table. The set of distractor objects is as follows: An ice tea package, a pancake tube, a hotplate and a spatula.

B *Searching for a Labeled Cornflakes Package in the Default Scene*

This scenario is identical to the first scenario except that additional knowledge about the search target is available to the perception control framework. The *Goggles* annotation (see section 2.2.3) is able to extract the visible text of the ROI that corresponds to a recognized object. The text that is visible best on the corn flakes package in the default scene is the label “CORN FLAKES”. This knowledge is represented in the task description in order to investigate the influence of known annotations on the perception control framework.

C *Searching for a Black Spatula in the Default Scene*

In this scenario, the visual search target is the spatula that is clearly visible in the default scene. Contrary to previous search targets, the spatula has a plain color. The black color channel clearly dominates the appearance of the spatula. Furthermore, there is another object visible with a dominant black color (the pancake maker). This ambiguity might distract the attention mechanism so that more operations are needed in order to fulfill the search task. The *Color Ratio* annotation (see section 2.2.3) is able to extract information about different color channels of the ROI that corresponds to a recognized object. This knowledge of the black color of the spatula is roughly represented in the task description in order to investigate the influence of known annotations on the perception control framework.

D *Searching for a Partially Occluded Cornflakes Package in the Occlusion Scene*

In previous scenarios, all considered objects are clearly visible. Partial occlusion of objects and objects, which are close to each other may lead to difficulties because the perception methods may not work reliably in such situations. In this scenario, the PR2 is looking for the cornflakes package on the shelf space of the kitchen table. A cornflakes package is present but it is partially occluded (from the point of view of the PR2) by an ice tea package.

E *Searching for a Partially Occluded Cornflakes Package in the Cluttered Scene*

In this scenario, the PR2 looks for a corn flakes package on the shelf space of the table. There are 9 visible objects on the table including a corn flakes package which is partially occluded by an ice tea package. This scenario has the objective to investigate the influence

of the number of recognized objects on the perception control framework.

2.4 Objective and Requirements

The objective of this thesis is to adapt a particular expert system architecture for continuous perception and methods of computational attention in the kitchen domain. Only knowledge-based (top-down) attention is investigated (see section 3). The perception control framework is responsible for the selection of objects, regions and features based on the outcomes of the attention methods as well as it is responsible for the selection and configuration of perception methods.

Top-down attention is usually based on knowledge about the current task. Thus, tasks play a special role for the perception control framework and methods of computational attention must be able to access the task information. Furthermore, it is required that tasks can be specified dynamically in order to allow continuous perception with changing tasks.

Hayes-Roth [Hay85] defines the *control problem* as the problem to select actions in a problem solving process. The definition can be transferred to perception control as follows: The perception control problem is the problem to select and configure perception methods at each point in the perception process.

Requirements of the perception control framework were identified based on problems that occur in the application domain (discussed in section 2.1). Claimed requirements and objectives are discussed in following paragraphs.

Expandability of the Control Framework The control framework that is implemented in the scope of this thesis is designed for perception tasks in the household domain. It is desirable that perception related components could be replaced to allow the expandability of the framework to other domains (e.g., the manipulation of objects). Furthermore, it should be possible to add new functionality to the framework without deeper knowledge about all control components. Ideally, functionality of the framework should be represented by different interfaces, which allow different implementations of the functionality (e.g., different rule formalism through a well defined interface for rules).

Adaptability to Different Scenarios In the scope of this thesis, the perception control framework is validated using a PR2 that performs everyday household tasks in a particular reference kitchen. The perception control framework should be adaptable for other robots and kitchens. This requires that the robot and the kitchen are only represented in the knowledge base and in the control knowledge (i.e., no source code modification is required for adaption).

Furthermore, it should be possible to use parts of the control knowledge in different contexts (i.e., for different tasks).

Maintainability of Control Knowledge The modification and extension of the knowledge base that is used by the control component may lead to inconsistencies in the control knowledge. This is caused by an unclear structure of control knowledge where different types of knowledge are mixed with each other. A clear user interface for the acquisition and maintenance of control knowledge is required as well as the separation of different types of control knowledge.

Explicability of Control Decisions For the acquisition and maintainability of control knowledge, it is important that knowledge engineers can understand the reason for particular (wrong) control decisions. This allows identifying the part of the control knowledge that is responsible for the particular decision. Furthermore, the explicability of control knowledge is important for intelligent backtracking methods (i.e., backtracking where only decisions are taken back that were based on the wrong decision).

Robustness with respect to the Versatility of Household Items One difficulty in the household domain is that household items may appear in various sizes, shapes and colors and with different textures. For example, the texture of ice tea packages is different for each manufacturer. The control framework must be able to handle this versatility in manifestations of household items.

Scalability with respect the Number of Visible Objects For each recognized object, there is a set of control decisions that can be selected in order to specialize particular aspects of the object (e.g., find out the color of the object). The main purpose of the perception control framework is the selection of such decisions based on knowledge about the current task. Thus, the number of possible control decisions is proportional to the number of recognized objects. The control framework must be able to act in scenes with a realistic number of visible object with a large number of possible control decisions.

Selection of Control Decisions The perception control framework uses computational attention methods in order to select objects, regions and features of visible objects that should be analyzed by perception methods. The usefulness of the attention methods can be estimated by comparing the attention-based selection with a selection by random chance. The attention methods are useful if the selection is better than by random chance.

Top-Down Attention Approaches

In this chapter, different approaches for knowledge-based attention models are discussed with respect to the application domain (see section 2.2) and problems (see section 2.1) that might occur in this domain. Thus, the focus in this chapter is on top-down attention models which can perform in natural and dynamic environments. In most attention systems, only bottom-up factors are investigated and saliency is computed based on the conspicuity of low-level image features such as color, intensity and orientation. The input of such systems is usually a RGB image and the saliency is computed separately for distinct rectangular image regions while the relation of these regions with physical objects is ignored. In literature, top-down factors are usually divided in knowledge about particular search targets (e.g., discriminative features), scene context (e.g., the semantic category of the scene) and knowledge about the current task. Some temporal models also considered knowledge about previous evidences (such as previous gaze positions) in the attention mechanism.

A selection of such knowledge-based approaches is discussed in this chapter. The objective of this discussion is to get an overview of the state-of-the-art in knowledge-based attention modeling in order to be able to make a well-founded decision about the integration of attention methods into the perception control framework.

The most intuitive way to integrate top-down factors in attention models is to enhance features that distinguish a target object best from other objects in the scene as proposed by Wolfe, Cave, and Franzel [WCF89]. This is illustrated in figure 3.1. For instance, every location with a dominant red color could be enhanced when the search target is a red object. Some weighting approaches inhibit the target-irrelevant regions while other approaches prefer to excite target-relevant regions. This procedure is also called *top-down weighting* or *top-down biasing*.

In many other approaches, the semantic category of the scene is used to infer likely gaze positions based on eye tracking training data. Such approaches often correlate the scene category to the gaze positions with respect to the current task. Where the task is only coarsely associated with a label (e.g., “grasping”). More elaborated approaches associate atoms from the task description with objects known to the system based on relationships between objects and tasks

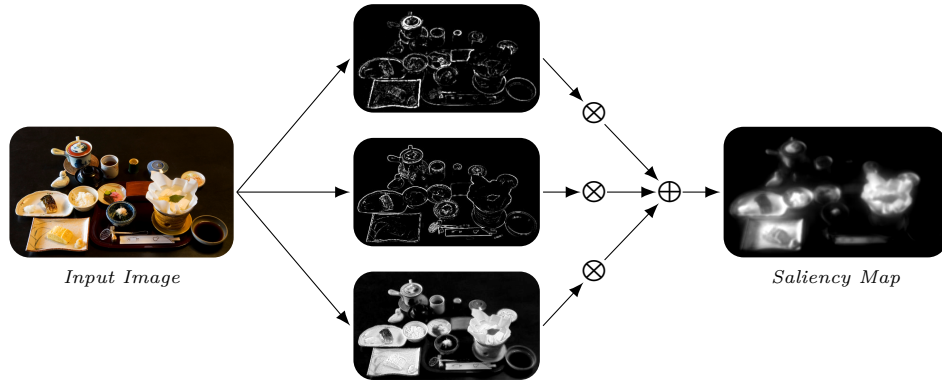


Figure 3.1 Basic top-down biasing architecture. First, the input image is decomposed into multiple feature channels. From top to bottom, the feature maps correspond to RG, BY color channels and intensity channel. The topographic conspicuity is then estimated by a weighted linear combination of all feature responses.

(e.g., “grasping” correlates with small objects).

Many models used standard metrics for the evaluation. The most frequently used metrics are introduced in following paragraph.

Evaluation Metrics One of the most used evaluation metrics for visual saliency is the Area under the ROC curve (AUC) [TBG05] metric. It can be computed based on a comparison of human eye fixation data (ground-truth) with the conspicuity that was estimated by the attention model. Using a threshold, the ground-truth fixations can be interpreted as a classification into attended locations and unattended locations. This ground-truth about the eye fixations of humans is used to find the true positive rate (*recall*) and the false positive rate (*fall-out*) of the estimated conspicuity in the spatial domain. The Receiver Operating Characteristics (ROC) curve is obtained by plotting the true positive rate against the false positive rate for different classification thresholds. The area under this curve is defined as AUC. $AUC = 1$ indicates perfect prediction while $AUC = 0.5$ indicates that the model does not perform better than by chance. Another frequently used metric is the Normalized Scanpath Saliency (NSS) score [Pet+05]. NSS is the average of normalized saliency over all locations which were fixated on the scanpath of a human observer. For the normalization, each response is transformed to have zero mean and a unit standard deviation. Let μ_S be the mean saliency, σ_S be the saliency variance and S_i the saliency at location i . Then, the NSS score can be written as: $NSS = \frac{1}{N} \sum_{i=1}^N \frac{S_i - \mu_S}{\sigma_S}$. Where N is the number of fixated locations on the scanpath. $NSS = 0$ indicates that the model does not perform better than by random chance because the average saliency over the scanpath is not higher than the mean saliency over all locations. On the other hand, values greater than zero indicate that the saliency for locations on the scanpath was higher than the average saliency over all locations.

Furthermore, attention modules, which act as a part of a bigger system can be evaluated, based

on the performance and quality of this system. For instance, attention modules that act as a front-end for object-recognition can be evaluated in terms of recall, fall-out and speed of the object-recognition process. This allows investigating the benefit of using an attention mechanism for a particular application while the NSS and AUC scores can be used to compare different attention approaches.

It is hard to compare the perception control procedure with other attention methods. Methods of the perception component rely on the perceived depth and on knowledge about the environment. Thus, a test set for the comparison of the perception control with other attention methods can only include samples from the reference kitchen (see figure 2.6). It is not intended to create such a test set that includes a reasonable amount of samples.

3.1 Cognitive Models

Most computational attention models have in common that they are inspired by psychological and neurobiological concepts and theories of the human attention mechanism. Cognitive models usually can't be implemented directly on computers because they don't provide enough algorithmic details of the attention process. Nevertheless, they serve as a basis for many knowledge-based computational attention systems. For this reason, a brief overview of the most influential basic concepts and architectures of attention systems is given in section 3.1.1.

For the computational branch, the most influential psychological theory is the *Feature Integration Theory* [TG80]. It introduces the concept of *saliency maps*. A saliency map is a topographic map that highlights scene regions which attract special attention (conspicuity). Most attention models are designed for RGB images. In those models, the saliency is usually computed separately for distinct regions in the image without knowledge about the physical object that corresponds to the investigated region. Such models are called *region-based* or *space-based*. Contrary, models are called *object-based* if the sensory input is segmented into multiple object representations before the attention mechanism is involved. An object-based approach – Rensink's *triadic architecture of attention* – is introduced in section 3.1.1.

Saliency maps can be envisioned as gray-scale maps where brightness corresponds to conspicuity. Saliency is often determined by the degree of difference to neighbor locations in a bottom-up manner (e.g., by *center-surround-differences* [IKN98]). The center-surround operation can be interpreted as contrast computation of feature responses in the spatial domain where locations with high contrast correspond to high saliency. It can be computed as the difference of a (blurred) feature map at different scales (across-scale difference). Center-surround features are usually computed for low-level image features like color, orientation and intensity. Furthermore, several other features had been used for the computation of bottom-up saliency including flicker, motion and depth ([Dha03]; [MNE00]).

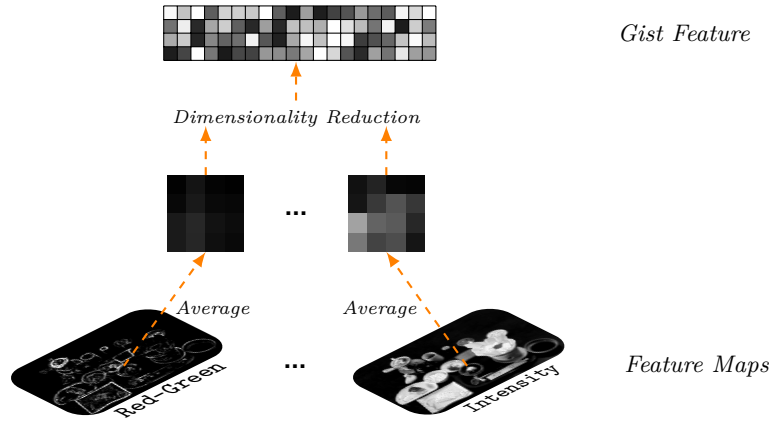


Figure 3.2 Holistic scene characteristics using low-level features as proposed by Siagian and Itti [SI07]. Feature maps are averaged in a fixed four-by-four grid and the dimensionality of the final scene description is reduced using Principal Component Analysis or Independent Component Analysis to obtain the final gist feature.

Knowledge about likely target appearance and feature responses only helps if the target object actually differentiates from the background. For instance, a plain white cup on a white kitchen table might not result in high saliency for the cup because the color feature responses are similar. In an object-based approach, both objects could be differentiated based on high-level features such as the size of the objects. Another approach is to use scene context knowledge or task knowledge to infer interesting regions, objects and features. Using such knowledge, the saliency does not necessary depend on local object features.

The semantic category of a scene can be defined at different levels of granularity. At a coarse level, it can be used to distinguish between drastically different environments (e.g., indoor or outdoor environments). It can also be used to distinguish scenes on a finer scale (e.g., in front of table or in front of cupboard). Many attempts were made to dynamically compute a holistic scene feature (also called *gist*) from low-level features which are also used for bottom-up saliency computation and object recognition. For instance, Siagian and Itti [SI07] proposed a method to extract holistic scene characteristics using low-level features obtained by center-surround differences. They averaged the responses of different feature channels in a fixed four-by-four grid of sub-regions over the maps. The gist vector over all feature channels (concatenation of all four-by-four grids) has 544 values in their discussion. The authors used *Principal Component Analysis* and *Independent Component Analysis* to reduce the dimensionality of the gist vector to 80 values while preserving 97% of the variance in a set of 30,000 campus images. This approach of gist vector computation is illustrated in figure 3.2.

Another approach for the computation of gist was presented by Renninger and Malik [RM04]. The authors proposed to extract so called *universal textons* from input images using an edge detection filter (*Gabor filter*) and a clustering mechanism (*K-means clustering*). The gist vector

is then defined as a histogram of those universal textons. The authors used the gist vector for the classification of scenes based on a set of 10 basic categories including a category for kitchens, bedrooms and bathrooms. Overall, they achieved 76% correct scene classifications with their model using a test set of 750 images and a training set of 250 images (25 examples for each basic category).

The computation of task relevancy of features, objects and regions is an important factor for visual attention [Yar67] and it requires prior knowledge about entities that are referred in the task description. Most authors interpreted the task as a label without semantic relationships to objects in the scene. In such approaches, the saliency is computed with respect to the task label. For instance, probabilistic models might use different density functions for different tasks in such a case. More sophisticated object-based approaches could use relations between objects in the scene and the current task to infer a saliency value with respect to the current task. For example, only small objects are graspable and grasping involves the grippers of the robot. Such relations can be used to infer a task relevancy value for regions, objects and features. Unfortunately, approaches based on relational knowledge are rare. One interesting approach where the task relevancy of objects is estimated based on pathfinding in a task graph is discussed in section 3.1.4.

3.1.1 Basic Theories and Architectures of Attention

The origin of all saliency map models is the *Feature Integration Theory* (FIT) that has been introduced by Treisman and Gelade [TG80]. It was introduced in 1980 but it was adapted constantly to reflect latest research findings. The original theory didn't incorporate attentional top-down factors explicitly. But many attentional top-down approaches are influenced by the FIT (e.g., [Fri06]; [NI06]) and therefore it's relevant in the scope of this thesis.

The theory suggests that attention is serially directed to different stimuli of interest whenever multiple visual features are needed to differentiate between a search target and distractors. Furthermore, the authors claimed that attention is a two stage process where functionally separable primitive features (basic features) – such as color, orientation or intensity – are registered in parallel in a first preattentive stage of the perceptual process and serially combined in a second stage. The preattentive stage is inspired by physiological evidence which shows that receptors respond selectively to basic features by mapping them to different regions in the human brain [Pal99]. The information gathered about different features is represented in so called *feature maps*. Feature maps are topographical maps that highlight the conspicuity of a particular feature in the scene.

Different feature maps are then combined into a single map, called the *master map of location*. This map highlights the overall conspicuity in the spatial domain. In the second stage, the focus of attention shifts between different scene regions by serially scanning through the master map.

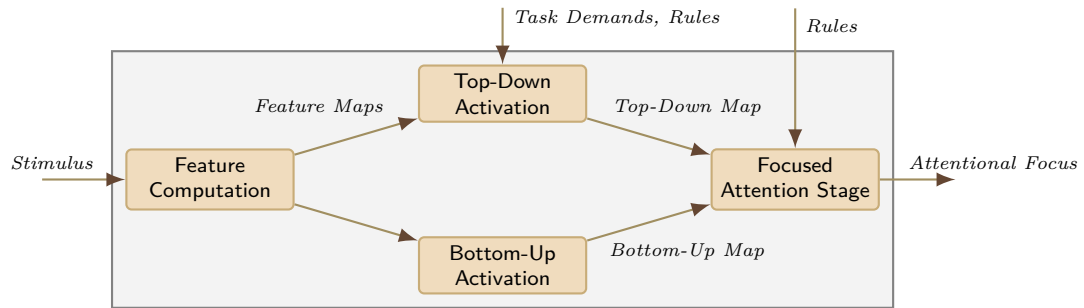


Figure 3.3 The stages of attention in the Guided Search model. The arrows indicate the inputs and outputs of the stages.

The authors claimed that focus of attention is required to integrate the data from different feature maps into high-level representations of unitary objects or regions that are used by higher perception tasks.

Guided Search Model The *Guided Search Model* (GS) is influenced by FIT but many aspects of the attentional mechanism are specified in more detailed which makes the GS model more suitable for the computational domain. The model also uses a preattentive stage where features are registered in parallel in order to obtain separate feature maps. The authors proposed to use one map for each considered feature type (e.g., one map for color and one map for orientation). In addition to bottom-up factors, the model also considers top-down factors (called *top-down commands*) in the preattentive stage. The separate feature maps are combined into a single map, called the *activation map*. This map is used to focus the attention by finding peaks in the map. This architecture is illustrated in figure 3.3.

In the GS model, top-down control of the attentional focus is accomplished by allowing to select one categorical channel per feature based on knowledge about the current task demands. For instance, the model allows to select the “red” channel of the color feature in a top-down manner. It’s intended to select the channel that best differentiates the target from surrounding distractors. This channel selection is accomplished using a set of hand-coded rules. For each feature, a top-down feature map is computed that represents the discrete feature channel selection.

The purpose of the activation map is to direct the attention to locations in the scene where high conspicuity indicates relevant regions, objects or features. Top-down and bottom-up feature maps are combined in the activation map. This is accomplished by reducing the bottom-up feature response strength for target features that appear multiple times in the scene. For instance, the response strength of the color feature is reduced proportional to the number of distractors with the same color as the target.

This is a basic approach for the extension of bottom-up attention models with a top-down component. But the channel selection procedure as well as the representation of tasks is not discussed by the authors. Furthermore, bottom-up attention is not investigated in this thesis.

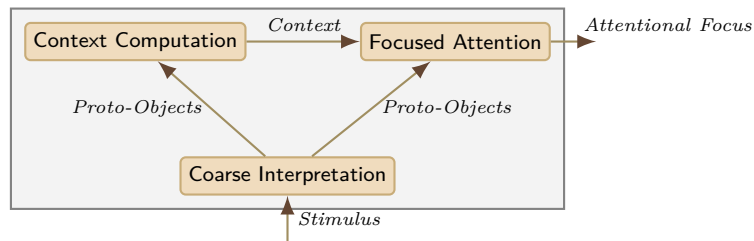


Figure 3.4 Different stages in Rensink’s triadic architecture of attention. First, a low level vision component segments the sensory input into different proto-object representations. Then, a contextual component computes layout and other context features based on the proto-objects. Finally, context and proto-objects are used to find the attentional focus.

But the idea of channel selection based on task demands can be adapted by a knowledge-based attention approach.

Triadic Architecture of Attention The *triadic architecture of attention* was introduced by Rensink [Ren00]. The author proposed to decompose the attention mechanism in three components.

In the first component, coarse representations of objects, groups of objects, or object parts – called *proto-objects* – are obtained from low-level features in parallel without focused attention. This component is similar to the preattentive stage of saliency map models.

The proto-objects are then refined to stable object representations using focused attention in a second stage. The authors claimed that humans only need detailed description of a small number of objects (usually one) and that focused attention is used to sequentially select proto-objects for the computation of a more detailed representation.

The last component is used to guide the attentional focus using context knowledge such as knowledge about the semantic category of the scene (e.g., indoor or outdoor) or the spatial arrangement (layout) of objects. The author proposed that this knowledge can be used to prime likely target objects by comparing the dynamically computed values to so called *scene schemata* which are stored in long-term memory. This triadic architecture of attention is illustrated in figure 3.4.

For object-based approaches, one of the most important differences compared to region-based approaches is that objects must be represented separately before the attention mechanism is involved. Thus, a segmentation algorithm is required in order to find different clusters which correspond to different objects before the object-based attentional focus is computed. A saliency-based approach for the segmentation of proto-objects in images was proposed by Walther and Koch [WK06]. They used the peaks in pixel-level (region-based) saliency maps for the estimation of proto-object extents.

Recently, Yanulevskaya et al. [Yan+13] proposed a bottom-up attention model based on ideas from the triadic architecture of attention. The authors proposed to estimate the saliency of objects using rarity-based and contrast-based saliency. Contrast-based saliency is computed based on the difference of feature responses of a proto-object to the responses of surrounding regions and other proto-objects. The authors computed internal and external contrast-based saliency. Internal contrast-based saliency reflects the variance of feature responses within proto-objects (the visual complexity) and external contrast-based saliency reflects the variance between different proto-objects. The authors proposed to compute the external contrast-based saliency between proto-objects using the *chi-square distance* between the color histograms. Internal and external contrast-based saliency are then averaged to obtain a result with respect to both modes. The rarity-based saliency primes proto-objects with rare and outstanding details. This follows the assumption that frequently occurring patterns are part of the scene background. The authors proposed to represent images using a *bag-of-visual words* approach based on the responses of Scale-Invariant Feature Transform (SIFT) [Low99] features. For rarity-based saliency, pixels with rare visual words are considered as salient. Finally, a rarity-based saliency value per proto-object is computed by averaging the rarity values over all pixels which belong to the proto-object. The resulting rarity-based saliency is then summed with the contrast-based saliency to obtain a saliency estimation per proto-object.

This theory suits well for the scope of this thesis. The model is object based (i.e., objects attract attention) and *RoboSherlock* (see section 2.2.3) provides perception methods for the clustering of proto-objects as well as it provides methods for the refinement of the recognized proto-objects. Unfortunately, the author did not proposed how to incorporate top-down factors into the model. Nevertheless, the idea of proto-objects can be adapted for knowledge-based attention approaches where the refinement of proto-objects is driven by knowledge about the environment and the problem-solving process (i.e., control knowledge).

3.1.2 \mathcal{SNR} Maximization

A seminal approach for top-down biasing is to compute the optimal weights based on the *signal-to-noise ratio* (\mathcal{SNR}). \mathcal{SNR} is the ratio of the strength of a target signal (e.g., the intensity of color) over the noise from distractors. In this model, feature weights are usually used to sum the saliency within feature dimensions as well as they are used to combine the saliency across feature dimensions. Maximization of \mathcal{SNR} leads to maximization of the object detection speed and to optimal feature weights.

Several authors discussed this approach. For instance, Frintrop [Fri06] learned the weights based on a set of manually annotated training scenes and based on a geometric mean computation. Another approach was presented by Navalpakkam and Itti [NI06] where the maximization was done based on differentiation of the ratio with respect to both feature weight levels (within and

across dimensions).

Both works proposed a model of combined bottom-up and top-down attention. Input images are decomposed into multiple features (intensity (I), color (C), orientation (O)), different feature channels per feature and each channel is represented at different scales using an image pyramid. In [NI06], the feature channel maps are linearly combined using weights that maximize the \mathcal{SNR} . The resulting feature maps are linearly combined again to obtain the saliency map of combined top-down and bottom-up influences. In [Fri06], the top-down computation is separated from the bottom-up processing pass. In a first step, learned top-down weights are used to compute an *excitation map* and an *inhibition map*. The excitation map represents regions with features similar to the target and the inhibition map represents regions with features that are more present in the background. The *top-down map* is obtained by point-wise subtraction of the inhibition saliency from the excitation saliency. Finally, the top-down map is linearly combined with the bottom-up map to obtain the saliency map.

Geometric Mean Approach Frintrop [Fri06] computes the weight ω_i for feature i of target object t by dividing the mean saliency in the target region by the mean saliency in the background region for a set of N training images. The resulting weights are averaged over all training images using the geometric mean of weights. Thus, the weight ω_i for feature i can be written as:

$$\omega_i = \sqrt[N]{\prod_{j=1}^N \mathcal{SNR}_{ij}} = \sqrt[N]{\prod_{j=1}^N \frac{\frac{1}{T_j} \sum_{t=1}^{T_j} \mathcal{S}_{ijt}}{\frac{1}{B_j} \sum_{b=1}^{B_j} \mathcal{S}_{ijb}}} \quad (3.1)$$

Where T_j, B_j are the numbers of target and background locations or objects in training image j and $\mathcal{S}_{ijt}, \mathcal{S}_{ijb}$ are the bottom-up feature responses for feature i in target and background regions of training image j . The weight is high for features which yield in high saliency for the target rather than the background. For instance, training samples with many colorful cups and plain colored background yield in high weights for color features of cups.

In the testing phase, these weights are used to compute an excitation map for likely target feature responses and an inhibition map for likely background feature responses. The top-down map \mathcal{S}_{TD} is then obtained by subtraction of the inhibition from the excitation responses:

$$\mathcal{S}_{TD} = \sum_i^K \omega_i \mathcal{S}_i - \sum_j^K \omega_j^{-1} \mathcal{S}_j \quad | \quad \forall \omega_i > 1, \omega_j < 1 \quad (3.2)$$

Where \mathcal{S}_x is the bottom-up response of feature x and K is the number of bottom-up feature maps (within and across feature dimensions). The top-down map is then linearly combined with the bottom-up saliency map \mathcal{S}_{BU} to yield the final saliency map: $\mathcal{S} = (1.0 - \gamma)\mathcal{S}_{BU} + \gamma\mathcal{S}_{TD}$. Where γ is a user defined factor between 0.0 and 1.0.

The author tested the model for visual search in more than 1000 real-world images where target

objects occur in different context, with different distractors and with different colored and structured background. The results were compared for different values of γ (0.0, 0.5, 1.0) and it was shown that the top-down influences improved the detection rate (i.e., the percentage of samples where the target was found within a fixed number of steps) while the average hit number (i.e., the number of target fixations) was lower then without top-down factor ($\gamma = 0$).

One drawback of this approach is that strong responses of target features maximize the \mathcal{SNR} more then weak responses. As a result, gains of target features with weak feature responses can't be enhanced (e.g., the black color of a cup can not be enhanced if the color is discretized in RGB color space). Also, the model doesn't allow to incorporate other sources of knowledge, such as knowledge about the task or relationships between objects in the scene. On the other hand, this approach is easy to implement, can be integrated into an expert system architecture (ordering knowledge) and the model showed good results in a real-time scenario similar to the application domain investigated in this thesis (as a front-end for object recognition). Furthermore, the top-down component (i.e., the weight computation) can easily be used without the bottom-up part which does not match the scope of this thesis. Finally, the attention model is region based (i.e., spatial locations attract attention) but it can easily be adapted to an object based approach where the features of different objects are compared with each other instead of features at image regions.

Differential Approach In [NI06], the weights are applied between stages of the standard bottom-up saliency computation process rather then separated from it. Let \mathcal{S}_{ij} be the saliency that corresponds to feature $i \in \mathcal{F} = \{I, C, O\}$ and feature dimension j . Then, the saliency \mathcal{S}_i of feature i is computed as linear combination across K_i feature dimensions using learned feature weights ω_{ij} : $\mathcal{S}_i = \sum_{j=1}^{K_i} \omega_{ij} \mathcal{S}_{ij}$. Similarly, conspicuity maps are linearly combined across all features using learned feature weights ω_i . The resulting saliency \mathcal{S} is computed as follows:

$$\mathcal{S} = \sum_i^{\mathcal{F}} \omega_i \sum_{j=1}^{K_i} \omega_{ij} \mathcal{S}_{ij} \quad (3.3)$$

In the learning phase, the authors distinguished between expected target saliency \mathcal{S}_t and expected background saliency \mathcal{S}_b across and within feature dimensions. \mathcal{SNR} is then defined as the ratio of \mathcal{S}_t over \mathcal{S}_b :

$$\mathcal{SNR} = \frac{\mathcal{S}_t}{\mathcal{S}_b} = \frac{\sum_i^{\mathcal{F}} \omega_i \sum_{j=1}^{K_i} \omega_{ij} \mathcal{S}_{ijt}}{\sum_i^{\mathcal{F}} \omega_i \sum_{j=1}^{K_i} \omega_{ij} \mathcal{S}_{ijb}} \quad (3.4)$$

$\mathcal{S}_{ijt}, \mathcal{S}_{ijb}$ are the bottom-up saliency responses of feature i at feature dimension j in target and background region. In order to compute weights which maximize the \mathcal{SNR} , the authors proposed to differentiate with respect to ω_i and ω_{ij} . This results in:

$$\omega_{ij} = \frac{\mathcal{SNR}_{ij}}{\frac{1}{K_i} \sum_{k=1}^{K_i} \mathcal{SNR}_{ik}} ; \quad \omega_i = \frac{\mathcal{SNR}_i}{\frac{1}{\#\mathcal{F}} \sum_n \mathcal{SNR}_n} \quad (3.5)$$

Where $\mathcal{SNR}_{ij} = \frac{\mathcal{S}_{ijt}}{\mathcal{S}_{ijb}}$ and $\mathcal{SNR}_i = \frac{\mathcal{S}_{it}}{\mathcal{S}_{ib}}$.

The authors tested the model on natural scenes by training it on 10 images containing different views of a cellphone and applying it to 50 other test scenes with slightly different backgrounds, locations, views and sizes. They compared the results with a naive bottom-up approach and showed that their model improved the object detection speed.

This model has the same drawbacks as the geometric mean approach: Low target feature responses can't enhance saliency and other sources of knowledge can't be integrated into this model. But this model is more flexible than the geometric mean approach because weights are applied at two levels of the \mathcal{SNR} computation. The computation of top-down weights can be separated from the bottom-up map and it can be integrated into the perception control framework using ordering knowledge. The region based definition of the \mathcal{SNR} can easily be adapted to an object based attention approach as required in the context of this thesis.

3.1.3 Euclidean Distance Minimization

Borji, Ahmadabadi, and Araabi [BAA11] proposed an attention model that uses a global optimization algorithm to find a weight vector with maximum detection rate and minimum processing costs. The optimization for minimum processing costs makes this model particularly useful in dynamic environments when limited resources are available for processing sensory stimuli. In their model, weights are learned within and across feature dimensions. Additionally, weights for six scales of feature pyramids are learned.

In the first step, input images are decomposed into intensity, color and orientation features with multiple channels and at six scales (using Gaussian pyramids). A center-surround operation is applied to the pyramids at all scales. This operation enhances features that discriminate a location from surrounding locations. Afterwards, conspicuity maps within feature dimensions are computed by normalizing and summing the weighted pyramid maps at different scales. Next, these conspicuity maps are normalized, weighted and summed again to obtain one conspicuity map per feature. Finally, the overall saliency map is computed as weighted sum of normalized responses per feature.

To formalize this, the saliency \mathcal{S}_i of feature $i \in \{I, C, O\}$ is computed using a weighted sum of saliency responses \mathcal{S}_{ij} of different feature channels $j \in \{1, \dots, K_i\}$, where K_i is the number of channels of feature i . Each channel j is weighted by a learned factor ω_{ij} . The sum is normalized using a nonlinear and iterative normalization operator \mathcal{N} [IK01] which is used to enhance strong responses while impairing weak responses (i.e., peaks are more evident after normalization). \mathcal{S}_{ij} is computed similarly based on the bottom-up responses \mathcal{S}_{ijk} and the weighting factors ω_{ijk} over the six different scales of feature i in channel j . Finally, the overall saliency \mathcal{S} can be computed

as:

$$\mathcal{S} = \sum_i^{\{I,C,O\}} \omega_i \mathcal{N} \left(\sum_{j=1}^{K_i} \omega_{ij} \mathcal{N} \left(\sum_{k=1}^6 \omega_{ijk} \mathcal{S}_{ijk} \right) \right) \quad (3.6)$$

Where ω_i is the weight of the feature with name i .

The authors proposed to learn the weights using a global optimization algorithm that finds a trade-off between detection rate and processing costs. The resulting weight vector ω is a concatenation of scale, feature channel and feature weights. In the optimization process, a fitness value for each weight vector is computed. This is done based on a set of M training images. For each training image \mathcal{I}_i , the most salient position x_i was manually specified by the authors. The average euclidean distance between estimated most salient point $X(\mathcal{I}_i, \omega)$ and x_i over all training images was used to define an optimization problem:

$$\min_{\omega} \frac{1}{M} \sum_{i=1}^M \|X(\mathcal{I}_i, \omega) - x_i\| \quad (3.7)$$

The authors solved this problem using the *Comprehensive Learning Particle Swarm Optimizer* (CLPSO) [Lia+06]. To incorporate computational costs in the optimization process, the authors manually specified a cost vector for different pyramid scales (based on image resolution) and features (based on algorithmic complexity). The sum of costs is multiplied with the average euclidean distance to obtain a fitness value that corresponds to detection rate and computational costs. However, the selection of cost values for features is vague because the feature detection complexity is not discussed in detail and the weighting between costs and average euclidean distance in the heuristic is unclear.

The model was tested for pop-out tasks and object detection tasks in natural scenes for 5 different objects (ten training images per object). The authors compared the experimental results to a basic saliency model and reported that the detection rate is 8.4% – 15.6% higher compared to the basic model using the heuristic without cost factor while the detection rate is 0.0% – 9.1% higher using the heuristic with cost factor. Object were considered as detected if an attended point was generated for a region of 30 pixels around x_i within three fixations. The authors reported the computational costs in terms of time it took to recognize an object. The recognition took roughly 98 – 107 ± 13 milliseconds for their approach. They also reported that the bottom-up approach was slightly slower compared to their approach when the cost factor was used.

The optimization problem is formulated generic and can be used for all region based attention systems where ground-truth eye fixation data is available. For the reference kitchen, no such ground-truth eye fixation data set exists. It is not intended to acquire such training data in the scope of this thesis. In object based models, the distance between objects can be used instead of the distance between regions (i.e., optimization of object fixations against the ground-truth eye fixation training data).

Compared to previously discussed \mathcal{SNR} approaches, the across-scale weighting gives a finer

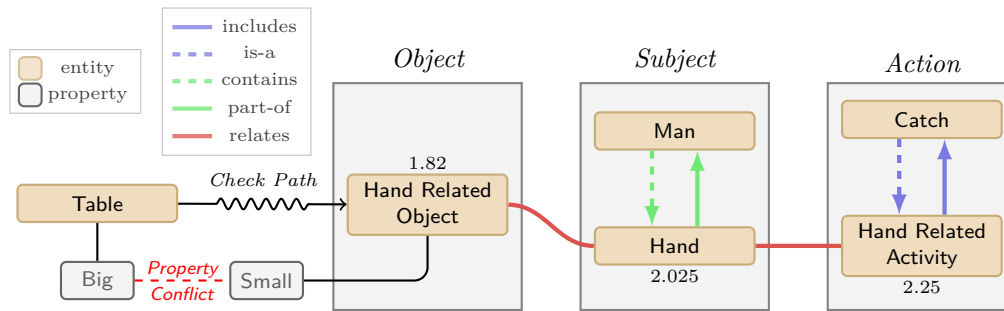


Figure 3.5 Task graph and task relevancy estimation proposed by Navalpakkam and Itti [NI05]. An initial task graph is expanded for the entities *Man* and *Catch*. The task relevancy of the *Table* entity is estimated based on a path that connects the entity with the task graph. In this example, *Table* is not task-relevant because hand related objects must be small.

control over the top-down influences and the optimization heuristic allows to associate high saliency to weak target feature responses (contrary to SNR approaches). However, this top-down approach is rather tied to the bottom-up architecture. This makes it hard to extend this model to other knowledge sources. Furthermore, the authors only tested the model on static images but the performance evaluation indicates that their model is real-time capable.

3.1.4 Relational Task Graph

Navalpakkam and Itti [NI05] proposed to parse task descriptions using an ontology (i.e., a knowledge base that contains prior knowledge about entities and their relationships) to yield task related entities and their relationships. Their ontologies are hand-coded symbolic knowledge bases that encode knowledge about objects and actions. They are represented as graphs with entities as vertices and relations as edges and they include relations such as *is-a*, *includes*, *part-of*, *contains*, *similar* and *related*. Spatial relations are not used in the model. Furthermore, the authors proposed to rank relations according to a priority value.

In this model, task keywords are expanded to *task graphs* that contain task-relevant entities and their relations. Only task specifications in the form object, subject and action are supported. The relevance of each keyword is manually defined by the user. Initially, the task graph only contains task keywords and their relevance value. Then the graph is expanded along the *is-a*, *has-part* and *related* relations. For each entity v , whose task relevance is unknown, a path in the ontology is obtained that connects v with an entity u in the task graph. This architecture is illustrated in figure 3.5.

The authors proposed to estimate the task relevancy of an entity v based on three factors: The task relevancy of other entities u with relationships to v , the priority of relationships between u and v and finally the probability of co-occurrence of both objects. Let $r(u, v)$ be a relation

between u and v . The priority function $g(\cdot)$ of this relation is then defined as $g(r(u, v))$. The authors proposed to generally prefer some relations over others:

$$\begin{aligned} g(\textit{contains}) &> g(\textit{part-of}) \\ g(\textit{is-a}) &> g(\textit{includes}) \\ g(\textit{related}) &> g(\textit{similar}) \end{aligned} \tag{3.8}$$

Where the actual priority values were hand-selected by the authors.

Furthermore, let U be a random variable that denotes the presence or absence of an object u and let V denote the presence or absence of an object v . The inheritance of relevancy along the path is weighted based on the object co-occurrence: $\mathcal{P}(U=1 | V=1) = \mathcal{P}(U=1, V=1)\mathcal{P}(V=1)^{-1}$. This heuristic prefers transitions between objects which appeared together often in training samples. The authors proposed to estimate this probability by counting co-occurrence of objects in training samples. Finally, the relevancy is decayed along the path using a user defined factor λ . Altogether, the task relevancy \mathcal{R}_v of entity v with respect to adjacent entities can be described as:

$$\mathcal{R}_v = \max_{u:\exists r(u,v)} \mathcal{R}_u g(r(u, v)) \mathcal{P}(U=1 | V=1) \lambda \tag{3.9}$$

Thus, the task relevancy of an entity is decayed by three factors: The transition costs $g(\cdot)$, the object co-occurrence and an user defined factor λ .

The authors used a *task-relevancy map* that encodes the relevancy of scene entities based on the task graph. Furthermore, they used a standard bottom-up attention approach that is biased using likely target features (as discussed in section 3.1.2). Both maps are then combined by a point-wise product to yield the so called *attention guidance map*.

Evaluation of multiple target detection was done by comparing the model with a naive bottom-up model. Visual features of targets were learned based on 12 training images and tested on 28 new scenes containing fire hydrants and handicap signs. On average, their model was 6.2 times faster than a naive bottom-up model. Furthermore, the authors compared their model with the *template-matching* model proposed by Rao et al. [Rao+02]. The proposed model is slightly faster in finding pop-outs but less efficient for conjunction search tasks.

The authors proposed a basic approach for the integration of relational knowledge in the attention mechanism. Unfortunately, many components were not implemented and many parameters were manually defined by the authors. Additionally, the object, subject and action format of tasks might not be a good fit for all household task (e.g., there is no object during navigation). Nevertheless, it's one of the only attention models with support for relationships between objects and it's possible to integrate other knowledge sources and other relations (such as spatial relations) into this model. Furthermore, it is possible to integrate this approach into the perception control framework using ordering knowledge in order to prefer control decisions that relate to the current task.

3.2 Probabilistic Models

In probabilistic attention models, the problem to find a meaningful attentional focus in a scene is formalized using a Probability Density Function (PDF). The underlying formalism of such models is often based on *Bayes' rule* (e.g., [Zha+08]; [IB09]). Bayes' rule can be used to combine sensory evidence with prior knowledge. For instance, it can be used to estimate the presence of a particular object class based on local image features (such as SIFT features).

To formalize this, let C be a binary random variable that denotes the presence ($C=1$) or absence ($C=0$) of a particular target class. Furthermore, let \mathcal{F} be a random variable that denotes a visual feature and let f be the last feature measurement at an arbitrary location. According to Bayes' rule, the probability to find the target given the local feature measurement can be written as:

$$\mathcal{P}(C=1 | \mathcal{F}=f) = \mathcal{P}(\mathcal{F}=f)^{-1} \mathcal{P}(\mathcal{F}=f | C=1) \mathcal{P}(C=1) \quad (3.10)$$

The first term, $\mathcal{P}(\mathcal{F}=f)^{-1}$ does not depend on the existence of any object classes. It reflects the conspicuity of the feature response f in a bottom-up manner (it's often estimated using a standard bottom-up attention model). The other terms depend on the target class, they reflect the attentional top-down factors. The term $\mathcal{P}(C=1)$ is the target prior, it gives the probability that the scene contains a target regardless of previous measurements. Finally, the term $\mathcal{P}(\mathcal{F}=f | C=1)$ indicates how likely it is to measure feature response f at locations where the target object class is present. In principal, Bayesian frameworks can be formulated for the prediction of both the next attended object (what) and the next attended spatial location (where).

One of the most influential Bayesian models of visual attention – called SUN (Saliency Using Natural statistics) – was proposed by Zhang et al. [Zha+08]. Bottom-up saliency emerges naturally as *self-information* (novelty) of visual features in this model and top-down factors are incorporated as a *log-likelihood* term in Bayes' rule. In this model, novelty is defined based on previous experiences, rather than on statistics of the current scene. The authors claimed that the saliency is given by the conditional probability $\mathcal{P}(C | \mathcal{F}, X)$, where X is a random variable that denotes the attended location (e.g., the pixel coordinates). Using Bayes' rule and assuming that features and locations are independent, this probability can be written as:

$$\mathcal{P}(C | \mathcal{F}, X) = \mathcal{P}(\mathcal{F})^{-1} \mathcal{P}(\mathcal{F} | C) \mathcal{P}(C | X) \quad (3.11)$$

The most salient location can be computed by maximization of the PDF over all locations. The estimation of probabilities is often based on normal distributions which include an exponential factor. This factor makes it difficult to differentiate the PDF in order to find the most salient location. For this reason, the *log-probability* is often used to eliminate the basis of the exponential

factor. The *log*-saliency $\log(\mathcal{S})$ can be written as:

$$\log(\mathcal{P}(C | \mathcal{F}, X)) = -\log \mathcal{P}(\mathcal{F}) + \log \mathcal{P}(\mathcal{F} | C) + \log \mathcal{P}(C | X) \quad (3.12)$$

The *log*-saliency can be used because the mapped values remain in the interval $[0, 1]$ (actually the log maps to $[0, -1]$ and the additive inverse of the log is often used to obtain positive values). The term $-\log \mathcal{P}(\mathcal{F})$ is called *self-information* of the random variable \mathcal{F} . Self-information increases with the probability of feature occurrence. This model reduces to the self-information term when no search target is given (then, only bottom-up influences are considered). The log-likelihood term $\log \mathcal{P}(\mathcal{F} | C)$ is used to prime feature values that are consistent with the knowledge about features of the target. For instance, the log-likelihood is high for red colored locations when the search target is a red cup. Finally, the third term $\log \mathcal{P}(C | X)$ is used to prime likely target locations independent of visual features (e.g., by spatial pooling). Both conditional probabilities can easily be learned based on training images with annotated object classes and object locations (natural statistics).

In the decision theory, attention is driven by optimality in regard to the end task (i.e., minimum probability of error). For instance, Gao and Vasconcelos [GV04] introduced *discriminant saliency* based on decision theoretic concepts. Saliency is interpreted as discriminant feature selection in this model and computed based on local features of the target and distractors. The authors used a decision-theoretic rule to avoid salient background features in the attention mechanism. This approach is discussed in more detail in section 3.2.3.

Another approach is to maintain the conditional independence structure between random variables using a graph structure (*graphical models*). Graphical models treat the attention process as a time series and use formalism that incorporate hidden variables such as *Hidden Markov Models* (HMM), *Dynamic Bayesian Networks* (DBN) or *Conditional Random Fields* (CRF). In graphical models, the saliency prediction is usually done with respect to previous gaze points or attended objects. For example, *graph-based visual saliency (GBVS)* is a graph-based model. It was introduced by Harel, Koch, and Perona [HKP07]. The authors used a fully connected graph over all locations of feature maps. Edges between nodes are weighted based on similarity and spatial distance and the resulting graph is interpreted as a Markov chain.

Finally, machine learning algorithms have been used to model the attention mechanism (*pattern classification models*). This is usually accomplished by training a “stimuli-to-saliency” function that is used to select, re-weight and integrate the input stimuli. For instance, Peters and Itti [PI07] used a regression classifier in combination with global features in order to learn a task-dependent association between gaze points and holistic scene representations. Their approach is discussed in more detail in section 3.2.1.

Probabilistic attention approaches provide an extensible framework for top-down attention. The basic approach is based on a Bayes’ rule where a Bayes’ classifier is used in order to estimate the probability that a feature measurement belongs to a particular target class. Other evidences –

such as high level object features – can be integrated into this model. Furthermore, it is possible to handle continuous features with Bayesian models using a normal distribution function. Thus, features such as the color of kitchen items must not be discretized in order to find statistical correlations. Many authors use the gaze position (i.e., the eye fixation position) as evidence in the probabilistic model (region based attention). This is not suitable for the context of this thesis because only object based attention is investigated. Nevertheless, it is possible to adapt this feature for object based approaches where the feature represents the fixated object instead of the fixated region.

3.2.1 Gist Projection Matrix

Peters and Itti [PI07] proposed a model of combined bottom-up and top-down attention that uses a statistical association between the gist of a scene and gaze locations in the context of video games. The bottom-up component is a standard model – based on [IKN98] – that computes saliency based on center-surround differences of visual features such as intensity, color, orientation and motion. The authors tested their model with two different gist representations: A pyramid-feature (spatial domain) representation with 448 elements per feature based on the mean and variance of center-surround differences at multiple scales (shown in figure 3.2) and a Fourier-feature (frequency domain) representation with 384 elements per feature based on different orientations and spatial frequencies in the image (as proposed by [Tor03]).

In this model, input images are passed to the bottom-up and top-down component in parallel. The bottom-up component computes a saliency map based on center surround-differences of features while the top-down map is computed based on a mapping of the scene gist to gaze locations. In the top-down component, the scene gist feature is dynamically computed each frame. This vector is then used to estimate a gaze position prediction map (i.e., with high values for likely gaze positions) based on a learned mapping from gist features to gaze locations. Finally, top-down and bottom-up maps are combined by point-wise multiplication to yield the final saliency map.

In this model, a projection between gist features and gaze points is learned in a training phase using a set of training images. For each training image, ground truth about the gaze points was collected using an eye tracking device. A coarse 20×15 grid is used to segment the scene into different regions and the number of eye fixations is counted for each grid cell. This yields in a 300-element gaze vector (density map) for each of the K training images. Furthermore, the gist vector is computed for each of the training images based on the gist approaches of [SI07] or [Tor03]. Let x_{kn} correspond to the element with index n of gaze vector x_k , where $k \in \{1, \dots, K\}$, $n \in \{1, \dots, N\}$ and $N = 300$. Furthermore, let f_{km} correspond to the element with index m of gist vector f_k , where $m \in \{1, \dots, M\}$ and M is the dimension of the gist vectors. Then, the

sequence of gist vectors ($K \times M$ matrix F) and the sequence of gaze vectors ($K \times N$ matrix X) which were obtained from K training samples can be written as:

$$F = \begin{bmatrix} f_{11} & \dots & f_{K1} \\ \vdots & & \vdots \\ f_{1M} & \dots & f_{KM} \end{bmatrix} ; X = \begin{bmatrix} x_{11} & \dots & x_{K1} \\ \vdots & & \vdots \\ x_{1N} & \dots & x_{KN} \end{bmatrix} \quad (3.13)$$

Both matrices are associated with each other using a $M \times N$ matrix W , where $X = FW$. W defines a projection from gist vectors to gaze density vectors. The objective in the learning phase is to estimate W based on the K training samples. If the inverse of F is defined, W can be estimated from: $W = F^{-1}X$. The authors used a more general approach to compute W based on the *pseudo inverse* F^+ of the gist matrix. It's called pseudo inverse because it does not share all properties of the matrix inverse in all cases. The computation of matrix W based on the pseudo inverse is given by: $W = F^+X$.

The pseudo inverse can be estimated by $F^+ = (F^T F)^{-1} F^T$ in case of linear independence between columns (gist vectors) in F (because the inverse is then defined). This equation is usually not solved directly due to numerical instability in computing the inverse. Instead, *singular value decomposition* (SVD) is widely used for the computation of the pseudo inverse. The SVD of the gist matrix yields in following factorization: $F = U \Sigma V^T$. Where U, V are orthogonal matrices (i.e., multiplication with transpose gives identity) and Σ is a diagonal matrix of non-negative real numbers (i.e., multiplication with pseudo inverse Σ^+ gives identity). The pseudo inverse can then be expressed in terms of SVD as follows:

$$X = U \Sigma V^T W \iff W = V \Sigma^+ U^T X \\ = F^+ X \quad (3.14)$$

Finally, the learned matrix W can be used to estimate the gaze density x given a gist feature f where $x = fW$. The estimated gaze vector is then mapped to a 20×15 grid over the visual field in order to obtain the top-down map.

The authors tested their model across 192,000 self-recorded video frames. They compared the predicted eye positions with tracked eye positions using the *NSS* score. The authors showed that the top-down mechanism outperforms standard bottom-up mechanisms and that a combination of both approaches yields in best results. The *NSS* score was roughly doubled compared to a standard bottom-up mechanism (from 0.58 ± 0.08 to roughly 1.2 ± 0.10). Furthermore, the model performed better than a simple heuristic that used the mean position of eye tracking samples. Additionally, they showed that the Fourier-feature gist representation performed slightly better than the pyramid-based representation.

In the context of this thesis, one of the major drawbacks of this approach is that it's limited to the processing of low-level image features. The authors did not proposed how object-based features – such as relations between objects – can be integrated into their model which is required

in the context of this thesis. Furthermore, the model does not support fast online learning – the projection matrix must be recomputed for every new training sample. Furthermore, the model is region based and it is not obvious how to adapt this approach for object based attention.

3.2.2 Feature Prior for Target Classification

Elazary and Itti [EI10] proposed a probabilistic approach for top-down biasing using a Bayesian framework in combination with a basic saliency model. The saliency map is computed based on a set of topographical *probability maps* (one for each feature channel). A high value in a probability map indicates high probability for that location to contain a target object.

The authors claimed that the saliency map that’s generated by their model can be used to find more robust feature sets and key-point locations for object recognition than wide-spread approaches like SIFT- or HMAX-based recognition ([Low99]; [SWP05]). Furthermore, the authors claimed that their model speeds up the recognition process compared to those methods.

In the proposed attention mechanism, input images are decomposed into color, intensity and orientation channels with two, one and four channels respectively. For each channel, an image pyramid is computed and center-surround operations are applied to the resulting pyramids at different scales to obtain multiple conspicuity maps for each channel. These maps are used to sample a PDF that corresponds to the probability of a particular feature channel response to belong to the target object. Contributions from different scales are multiplied to obtain one probability map per feature channel. The saliency map is then computed by multiplication of all probability maps.

Let the random variable $\mathcal{F} = (\mathcal{F}_1, \dots, \mathcal{F}_K)$ denote the continuous feature measurements from K different maps. The random variable \mathcal{F}_k denotes the feature response of a particular feature map with index $k \in \{1, \dots, K\}$. For a particular target class c and feature channel k , parameters μ_{ck} and σ_{ck} for the probability estimation are learned in advance. μ_{ck} represents the mean feature channel response and σ_{ck} the standard deviation around the mean over all training samples. The learned parameters are used to estimate the probability to measure feature response \mathcal{F}_k at target objects: $\mathcal{P}(\mathcal{F}_k | C)$. The authors used the Gaussian Distribution (GD) for the probability estimation:

$$\begin{aligned} \mathcal{P}(\mathcal{F}_k = f_k | C = c) &\propto GD(f_k, \mu_{ck}, \sigma_{ck}) \\ &= \frac{1}{\sigma_{ck} \sqrt{2\pi}} \exp\left(-\frac{(f_k - \mu_{ck})^2}{2\sigma_{ck}^2}\right) \end{aligned} \quad (3.15)$$

GD was chosen for its simplicity and efficiency in obtaining the parameters μ_{ck} and σ_{ck} in an on-line method from training images. The probability is distributed symmetrically and descending about the mean μ_{ck} (bell-shaped) where the width of the “bell” is controlled by the standard deviation σ_{ck} . Thus, it’s assumed that $\mathcal{P}(\mathcal{F}_k = \mu_{ck} + \epsilon | C) = \mathcal{P}(\mathcal{F}_k = \mu_{ck} - \epsilon | C)$ and

$\mathcal{P}(\mathcal{F}_k = \mu_{ck} + \epsilon_1 | C) > \mathcal{P}(\mathcal{F}_k = \mu_{ck} + \epsilon_2 | C)$ if $|\epsilon_1| < |\epsilon_2|$.

The saliency \mathcal{S}_c with respect to a particular target object c is expressed using above PDF over all feature maps: $\mathcal{P}(\mathcal{F} | C=c)$. The authors assumed conditional independence between feature channels given the target class ($\mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_j | C$). Thus, $\mathcal{P}(\mathcal{F} | C=c) = \prod_k^K \mathcal{P}(\mathcal{F}_k | C=c)$. This is certainly not true (e.g., correlations between intensity and color) but the authors claimed that the dependence between features would increase the accuracy only by a small amount. As in many other models, the *log* of the probability is used to allow easier maximization of the PDF. Above discussion yields in:

$$\begin{aligned} \mathcal{S}_c &= \log \mathcal{P}(\mathcal{F} | C=c) = \log \prod_{k=1}^K \mathcal{P}(\mathcal{F}_k | C=c) \\ &= \sum_{k=1}^K \log \mathcal{P}(\mathcal{F}_k | C=c) \end{aligned} \quad (3.16)$$

The authors used this saliency estimation for a Bayes' classifier in order to find the most likely object class \mathcal{C} given the local feature measurements:

$$\mathcal{C} = \arg \max_c \mathcal{P}(C=c | \mathcal{F}) \quad (3.17)$$

Where the feature prior was dropped because it's the same for all object classes. Thus, it can be interpreted as constant without influence on the classification problem.

The authors claimed that their model of combined attention and recognition is robust to changes in transformation (i.e., position and scale) and illumination of objects. Furthermore, they claimed that the recognition performance is on par or better than SIFT- and HMAX-based recognition while the computational costs are significant reduced. The model was tested across three popular data-sets (ALOI [GBS05], COIL [NNM96], SOIL-47 [BAK]) that include 87,810 photographs (scaled to 256×256 pixels) of more than 1000 objects under various transformations and illuminations. According to the authors, the average detection rate over all data-sets – using 25% of the data-sets as training images – is 88.64% for the proposed model, 84.78% for the SIFT approach and 72.77% for the HMAX approach. Furthermore, their approach (3.42h) was roughly 1500 times faster than SIFT (4878.3h) and 279 times faster than HMAX (678.55h) for one half of the ALOI data-set (40,000 images). On average, the classification for a single image took 0.165 seconds for their approach.

The reported performance improvement is impressive and the recognition rate of their Bayes' classifier is on par with the state-of-the-art in object recognition. The performance evaluation indicates that the model is real-time capable but the authors did not tested it for dynamic scenes. One advantage over previously discussed methods is that the learned parameters of the GD (mean and variance) can easily be updated from online samples without interfering with parameters of other features. On the other hand, the model only uses knowledge about char-

acteristics of a single search target while knowledge about the scene, relations between objects and the current activity is ignored. Furthermore, the model misses some statistical correlations due to independence assumption between features. The authors reported that the accuracy is only improved by a small amount when dependence between features is assumed. Since the authors used natural images for the validation of their model this might hold true for the kitchen domain as well. Finally, the GD is symmetric about the mean over all training samples which is unfavorable for features with high variance such as the color of cups.

3.2.3 Mutual Information of Evidences and Target Class

In [GHV09], the authors defined top-down saliency with respect to a one-versus-all classification problem: Target features correspond to a single object class of interest and background features correspond to the rest of the scene. Similar to the approach discussed in section 3.2.2, the model estimates the likelihood of feature responses given that the location belongs to the target or background. But the estimation is done using a more general distribution function: The Generalized Gaussian Distribution (GGD). The learning problem reduces to the learning of the parameters of the distribution function per feature channel. Another difference is that this model additionally learns a selection of discriminative target features. Uninformative target features are ignored in the testing phase. Finally, the authors proposed a saliency measure based on the *Kullback-Leibler* (KL) divergence between the likelihood term and the feature prior. The *KL* divergence is used to estimate if a particular feature response is characteristic for the target class.

Let the random variable $\mathcal{F} = (\mathcal{F}_1, \dots, \mathcal{F}_K)$ denote the continuous feature measurements from K different maps. The likelihood $\mathcal{P}(\mathcal{F}_k | C=i)$ of feature \mathcal{F}_k with $k \in \{1, \dots, K\}$ given that the location belongs to the target ($C = 1$) or background ($C = 0$) is estimated using a GGD. The GGD estimation depends on three parameters: The mean feature response μ_{ik} , the *scale* parameter α_{ik} and the *shape* parameter β_{ik} . α_{ik} is proportional to the standard deviation σ_{ik} but it also depends on the shape parameter. The shape parameter is a measure of the “peakedness” of the distribution: The distribution has sharp and long peaks for high values while low values yield in round and short peaks. This probability estimation can be written as:

$$\begin{aligned} \mathcal{P}(F_k = f_k | C=i) &\propto GGD(f_k, \mu_{ik}, \alpha_{ik}, \beta_{ik}) \\ &= \frac{\beta_{ik}}{2\alpha_{ik}\Gamma(\beta_{ik}^{-1})} \exp\left(-\frac{(f_k - \mu_{ik})^{\beta_{ik}}}{\alpha_{ik}^{\beta_{ik}}}\right) \end{aligned} \quad (3.18)$$

Where $\Gamma(z+1) = \int_0^\infty t^z e^{-t} dt$ is the gamma function. It can be interpreted as factorial extended for floating point numbers. This is evident when the function is rewritten using partial integration: $\Gamma(z+1) = [-t^z e^{-t}]_0^\infty + z \int_0^\infty t^{z-1} e^{-t} dt = z\Gamma(z)$. The shape parameter $\beta_{ik} = 1$ controls the decay of the peak. For instance, Gaussian normal distributions can be achieved with $\beta_{ik} = 2$

(since $\Gamma(\frac{1}{2}) = \sqrt{\pi}$). For $\beta_{ik} = 1$ the gamma function yields in $\Gamma(\frac{1}{1}) = 0! = 1$ while it goes to infinity for growing values of β_{ik} .

The authors proposed to use the *method of moments* for the estimation of scale and shape parameters. The shape parameter β_{ik} is estimated from the *kurtosis* $\kappa_{ik} = \Gamma(\frac{1}{\beta_{ik}})\Gamma(\frac{5}{\beta_{ik}})\Gamma(\frac{3}{\beta_{ik}})^{-2}$. And the scale parameter α_{ik} is estimated from the standard deviation and shape parameter: $\sigma_{ik}^2 = \alpha_{ik}^2 \Gamma(\frac{3}{\beta_{ik}})\Gamma(\frac{1}{\beta_{ik}})^{-1}$. For $\beta_{ik} = 2$ this yields in $\sigma_{ik}^2 = \alpha_{ik}^2 (\frac{1}{2}\sqrt{\pi})(\sqrt{\pi})^{-1} = \frac{1}{2}\alpha_{ik}^2$. Standard deviation, mean and kurtosis can be acquired from training samples and updated online when new samples are available.

In the information theory, the information content of a random variable is expressed in terms of *Shannon entropy*. The entropy can be interpreted as average information gain for the observation of a particular random variable: The entropy is maximal if all the outcomes of the variable are equally likely and it's minimal (zero) if all future outcomes are known in advance. The authors proposed to use the entropy as a selection criterion for discriminative target features. A feature \mathcal{F}_k is discarded if the entropy of \mathcal{F}_k given the background class label is higher then the entropy given the target class label (i.e., if the feature is more present in the background). The selection test can be written as:

$$E(\mathcal{F}_k | C=1) \leq E(\mathcal{F}_k | C=0) \quad (3.19)$$

The authors referenced that the entropy of feature \mathcal{F}_k can be directly estimated from the learned GGD parameters: $E(\mathcal{F}_k | C=i) = \frac{1}{\beta_{ik}} + \log\left(\frac{2\alpha_{ik}\Gamma(\beta_{ik}^{-1})}{\beta_{ik}}\right)$.

Additionally, a decision rule for feature selection based on the concept of *mutual information* was proposed. Mutual information measures the amount of information that one particular random variable contains about another one. It's zero between independent variables, while it's high for variables with strong dependencies. In this model, the mutual information between features and class labels is used to select features with strong dependencies to the target and background class. The mutual information between a feature \mathcal{F}_k and the class labels can be written as: $I(\mathcal{F}_k; C)$. \mathcal{F}_k is considered more informative about the target then another feature \mathcal{F}_m if $I(\mathcal{F}_k; C) > I(\mathcal{F}_m; C)$ holds true. The estimation of the mutual information is done using the *KL divergence*:

$$I(\mathcal{F}_k; C) = \sum_i \mathcal{P}(C=i) KL[\mathcal{P}(\mathcal{F}_k | C=i) || \mathcal{P}(\mathcal{F}_k)] \quad (3.20)$$

Where $KL[\mathcal{P}(\mathcal{F}_k | C=i) || \mathcal{P}(\mathcal{F}_k)] = \int_x \mathcal{P}(\mathcal{F}_k=x | C=i) \log \frac{\mathcal{P}(\mathcal{F}_k=x | C=i)}{\mathcal{P}(\mathcal{F}_k)}$ measures the difference between feature prior and feature likelihood given the class label. A large difference indicates strong dependence between class label and feature while a difference of zero indicates independence between the distributions. The authors also referenced a closed form of the *KL divergence* based on learned GGD parameters.

Finally, the saliency \mathcal{S}_k with respect to feature \mathcal{F}_k is defined as the mutual information between class labels and feature response f_k . All features \mathcal{F}_j that are more likely for the background class are ignored ($\mathcal{S}_j = 0$). The saliency \mathcal{S} is then defined as sum of saliency responses over K

selected features:

$$\mathcal{S} = \sum_{k=1}^K \mathcal{S}_k \ ; \ \mathcal{S}_k = \begin{cases} I(C; \mathcal{F}_k = f_k) & \text{if } k \in A \\ 0 & \text{otherwise} \end{cases} \quad (3.21)$$

Where $A = \{k \mid \mathcal{P}(\mathcal{F}_k = f_k \mid C=1) > \mathcal{P}(\mathcal{F}_k = f_k \mid C=0)\}$ is a selection set of more likely target features given the feature responses.

The authors used precision-recall curves to measure the localization accuracy of salient points against the bounding boxes of target objects on images from the PASCAL2006 data set [Eve+]. They compared the model with 3 other classification approaches (discriminative visual words [CZ07], linear support vector machine [JT05] and probabilistic latent semantic analysis [Siv+05]) and showed that the average precision over all object classes was at least 15% higher for their model.

Furthermore, the authors evaluated the classification performance by comparing their approach to a bottom-up model. They claimed that, for all object classes, their approach pruned away at least 30% of all points of interest according to the bottom-up model and that the savings in computational time are proportional to this value.

Summarizing, the GGD based probability estimation is more flexible than the GD based probability estimation because the additional shape parameter controls the “peakedness” of the *GGD*. The GGD parameters (mean, variance, scale and shape) can be updated in an online learning method but with higher computational cost due to additional parameters for the probability distributions. Another advantage of this model is that the saliency computation is restricted to informative target features while features and particular feature responses which are more likely for the background are ignored in the saliency computation. The knowledge involved is restricted to features which are informative for a particular target, other sources of knowledge are ignored (e.g., relations between objects).

3.2.4 Contextual Priors for High-Level Object Features

Torralba [Tor03] proposed an attention model which incorporates the global scene structure based on a probabilistic model over the statistical correlations between the gist of a scene and other random variables (including local object features, object class, gaze point and object properties). Their model can be used to predict the presence or absence of objects as well as their location, scale and appearance (e.g., scale, pose). In the proposed model, top-down and bottom-up computation is done on separate paths. On the bottom-up path, a conspicuity map is computed based on the likeliness of local feature measurements (i.e., large saliency for unlikely feature responses). Feature responses used for the computation of the bottom-up map are also used for the computation of the gist feature. The gist feature is computed based on different orientations and spatial frequencies in the image (Fourier-feature). This gist vector is then used to estimate

likely target positions using a Bayesian framework. Finally, the top-down and bottom-up map are linearly combined to yield the saliency map.

The basic approach of Bayesian saliency models is to estimate the likelihood of object class C given local feature measurements \mathcal{F} : $\mathcal{P}(C | \mathcal{F})$. This approach can trivially be extended to gist features by concatenation of local features and gist feature. However, this would lead to high dimensionality of feature vectors (which might yield in slow estimation of the PDF). In the proposed model, two sets of image features are used instead: Local image features \mathcal{F}_L obtained by center surround mechanisms and contextual image features \mathcal{F}_C that represent the gist of the scene. The gist is represented with 60 elements per feature based on different orientations and spatial frequencies in the image. Furthermore, the author augmented the PDF by introducing a random variable X that denotes the position of objects and a random variable T that denotes the appearance of objects. Where the appearance is expressed as a vector of object scale and aspect ratio. Let $O = (C, X, T)$ be a random variable that denotes an object. Using Bayes' rule, the probability of objects given local and contextual features can be written as:

$$\mathcal{P}(O | \mathcal{F}_L, \mathcal{F}_C) = \mathcal{P}(\mathcal{F}_L | \mathcal{F}_C)^{-1} \mathcal{P}(\mathcal{F}_L | O, \mathcal{F}_C) \mathcal{P}(O | \mathcal{F}_C) \quad (3.22)$$

The first term can be interpreted as bottom-up saliency. It gives the probability to perceive particular low-level features within scene context \mathcal{F}_C (large PDF values indicate uninteresting features). The PDF $\mathcal{P}(\mathcal{F}_L | O, \mathcal{F}_C)$ is the likelihood of the local measurement \mathcal{F}_L given the presence of object O in scene context \mathcal{F}_C . Thus, it gives high values for features which are likely for a particular object. In their discussion, the authors ignored this term and simplified the PDF to: $\mathcal{P}(O | \mathcal{F}_L, \mathcal{F}_C) = \mathcal{P}(\mathcal{F}_L | \mathcal{F}_C)^{-1} \mathcal{P}(O | \mathcal{F}_C)$. Finally, the last term $\mathcal{P}(O | \mathcal{F}_C)$ gives the probability of object O to occur in scene context \mathcal{F}_C (contextual priors). The author decomposed the contextual priors in three factors based on the proposed object representation:

$$\mathcal{P}(O | \mathcal{F}_C) = \mathcal{P}(T | X, C, \mathcal{F}_C) \mathcal{P}(X | C, \mathcal{F}_C) \mathcal{P}(C | \mathcal{F}_C) \quad (3.23)$$

These density functions represent the contextual selection of target appearances, positions and object classes.

The author proposed to use a mixture of $K = 2$ Multivariate Gaussian Distribution (MGD) weighted by learned factors α for the estimation of the density functions. Multivariate Gaussian distributions are parametrized by the mean vector μ of training samples and the covariance of different elements in the training vectors (covariance matrix Σ). The covariance cov_{ij} of vector elements i, j is computed as average of the statistical error of i multiplied by the statistical error of j over N training samples: $cov_{ij} = \frac{1}{N} \sum_{k=1}^N (x_{ik} - \mu_i)(x_{jk} - \mu_j)$. Where x_{ik}, x_{jk} are the values of vector element i, j for training sample k . In this approach, each dimension of the training vectors is considered to be a random variable. Mean and covariance are learned for all contextual

priors. For instance, the contextual class prior is estimated from:

$$\begin{aligned} \mathcal{P}(\mathcal{F}_C = f_k \mid C = c) &\propto \sum_{i=1}^K \alpha_{ci} MGD(f_k, \mu_{ci}, \Sigma_{ci}) \\ &= \sum_{i=1}^K \frac{\alpha_{ci}}{(2\pi)^{K/2} |\Sigma_{ci}|^{1/2}} \exp\left(-\frac{1}{2} \Delta^2\right) \end{aligned} \quad (3.24)$$

Where $|\Sigma_{ci}|$ is the determinant of Σ_{ci} and α_{ci} is the learned weight of the i 'th Gaussian function in the mixture. The weights are chosen so that they sum up to 1 over all Gaussian functions. The exponent $\Delta^2 = (f_k - \mu_{ci})^T \Sigma_{ci}^{-1} (f_k - \mu_{ci})$ is the squared Mahalanobis distance between f_k and μ_{ci} . It provides a measure of distance between the gist vector f_k and the mean vector μ_{ci} of the distribution with respect to the covariances in the distribution. Low distance yields in high probability. The distance reduces to euclidean distance if the covariance matrix is a diagonal matrix.

The log-likelihood function of mixture densities is hard to optimize (due to the log of the sum). Thus, the authors proposed to learn the distribution parameters based on the *expectation-maximization* (EM) algorithm. Initially, the EM algorithm chooses random values for the parameters of Gaussian functions in the mixture. In the expectation step, the current parameters of the distribution (e.g., initially random parameters) are used to estimate the ‘‘membership weight’’ w_{cik} of each training sample f_k and cluster i :

$$w_{cik} = \alpha_{ci} MGD(f_k, \mu_{ci}, \Sigma_{ci}) \left(\sum_{j=1}^L \alpha_{cj} MGD(f_k, \mu_{cj}, \Sigma_{cj}) \right)^{-1} \quad (3.25)$$

The membership weight of sample k in cluster i is high if the normal distribution of the cluster yields a higher probability for the sample than other clusters. In the maximization step, the membership weights are used to update all parameters of the Gaussian functions. Let $w_{ci} = \sum_{k=1}^N w_{cik}$ the sum of membership weights of cluster i . The weight of each cluster i is computed as average membership weight over all training samples: $\alpha'_{ci} = N^{-1} w_{ci}$. The mean of the distribution is then computed as sum of weighted training samples normalized by the membership weight of the cluster: $\mu'_{ci} = w_{ci}^{-1} \sum_{k=1}^N w_{cik} f_k$. Finally, the covariance matrix is updated using the standard covariance formula except that the contribution of each training sample is weighted by the membership weight: $\Sigma'_{ci} = w_{ci}^{-1} \sum_{k=1}^N w_{cik} (f_k - \mu'_{ci})(f_k - \mu'_{ci})^T$. This procedure is repeated multiple times until the EM algorithm converges.

The performance of this model depends on how well the gist feature differentiates scenes, on the relationship between objects and scene (object priming) and on the spatial distribution of objects (location priming). The model was tested using 2700 greyscale images (color feature was ignored by authors) with annotated object classes, locations and scales. Half of the images was used for testing and the other half was used for training. The authors reported that their model

correctly predicted the presence or absence of search targets in 81% of all test images (object-class priming). Furthermore, the authors compared the actual mean scale of target objects H with the scale prediction and showed that the prediction was in the interval $[0.5H, 2H]$ for 84% of the test images.

This attention model combines an object based and a region based attention approach. It incorporates high-level object features such as size and scale and it allows to extend the object definition in order to incorporate other high-level features which is required in the context of this thesis. Furthermore, no independence between object features and contextual features is assumed which allows object feature priors with respect to the current context (e.g., the current activity). In their discussion, the authors ignored the priming of likely target features but the approach can easily be extended to support this (e.g., using the approach discussed in section 3.2.2 or 3.2.3).

3.3 Conclusion and Discussion

In existing attention models, top-down factors are often integrated into a standard bottom-up attention architecture. Most of the discussed methods focus on knowledge about low-level features of the current task target (e.g., knowledge about the color of task targets). High level object features and relations between objects are rarely investigated.

The attention models are often used to predict the next eye fixation based on ground-truth eye fixation data that was gathered using an eye tracking device. In some attention models, the gaze position is associated to the semantic category of the scene (gist) in order to predict the next fixation in dependence of the semantic category of the scene.

Finally, most discussed approaches are region based (i.e., image locations attract attention) which is not in the scope of this thesis. But the adaption of the top-down components in region based models into an object based architecture is possible for the set of discussed attention models.

The priming of likely target features (see [Fri06], [NI06] and [BAA11]) is often done using top-down biasing where feature responses are biased using learned weights. The discussed approaches build on top of standard bottom-up attention architectures. The subject of this thesis is knowledge-based attention (i.e., top-down attention). Bottom-up aspects are not investigated. Thus, the architectures do not fit well with the scope of this thesis. Nevertheless, it is possible to adapt the top-down components of the discussed architectures for the perception control framework.

The top-down biasing methods do not work for situations such as navigation where no explicit target is known to the system. But the methods work for visual search tasks which are the most important perception tasks (e.g., it is only possible to grasp something that was recognized earlier).

Frintrop [Fri06] proposed to maximize the SNR (signal-to-noise ratio) in order to find search target specific weights for object features (see in section 3.1.2). One drawback of this approach is that low feature responses of target features cannot yield in high saliency responses. This problem does not occur for the euclidean distance minimization approach (see section 3.1.3) that was proposed by Borji, Ahmadabadi, and Araabi [BAA11]. Nevertheless, the euclidean distance minimization approach requires ground-truth eye fixation data which is not available for the reference kitchen and not intended to be acquired in the scope of this thesis. Thus, only the SNR approach can be investigated in the scope of this thesis.

The holistic scene representation (gist; see [Tor03], [RM04] and [SI07]) is primary used for a scene classification purpose. In the application domain of this thesis, it is assumed that the attention mechanism has access to knowledge about the current pose of the robot in the kitchen. For instance, it is known if the robot stands in front of the table or the cupboard. But, for instance, the set of items and their distribution on the table is unknown. It is not clear how well the gist representation performs for such an application because most authors used it for classification on a coarser level.

Only a few attempts exist that incorporate knowledge about relations of task and scene entities. The relational task graph approach that was proposed by Navalpakkam and Itti [NI05] is one of them (see section 3.1.4). Unfortunately, the authors used many user defined factors in the attention process and the definition of tasks was bound to a syntactical structure (object, subject and action) that does not fit with all activities that can be performed in kitchens (e.g., navigation). Nevertheless, the idea that task relevance can be estimated from the relation hierarchy of objects can be adapted for the perception control framework.

Models of attention which are based on Bayes' rule provide a generic framework that can easily be augmented by different types of knowledge. These models are often based on the classification of low-level feature responses (see [Zha+08] and [IB09]). For example, Elazary and Itti [EI10] proposed a basic probabilistic model of attention where only low-level features of the target class are used (see section 3.2.2). One disadvantage of this approach is that the saliency of a feature response only depends on the probability distribution of the target class while the correlations to other objects in the scene are ignored. For instance, Gao, Han, and Vasconcelos [GHV09] used the mutual information between feature prior and feature likelihood in order to enhance the saliency for features which are more characteristic for the target object class than for other object classes (section 3.2.3).

Torralba [Tor03] proposed a probabilistic attention model that uses multiple types of evidences: Low-level image features, the semantic category of the scene and a set of high-level object features such as size, shape and location (section 3.2.4). The authors used the semantic category of the scene for rough scene categorization (e.g., indoor or outdoor scene). This is not required for perception control in the kitchen domain because the semantic category of kitchens is always the same. The authors used the position feature for spatial pooling (i.e., high saliency for likely

target regions). *RoboSherlock* already provides methods for spatial pooling based on the currently investigated semantic location in the kitchen (e.g., the counter top of the table). Thus, a position feature for spatial pooling is not required for the perception control framework. Nevertheless, Torralba [Tor03] showed how to incorporate multiple evidences in a Bayesian attention model including high-level object features which are relevant in the scope of this thesis (i.e., size and shape).

In probabilistic attention models, normal distributions – such as the GD – are usually used for the probability estimation of continuous features. The parameters of these functions are learned in a training phase based on a set of annotated training images. Torralba [Tor03] used a mixture (weighted sum) of MGD for continuous features (see section 3.2.4). In this approach, the probability is not necessary symmetric about the mean of training samples which might be a good representation for the versatility in the household domain (e.g., the variance in color of different cups). Elazary and Itti [EI10] used a simple GD for the representation of probability densities of continuous features. In the GD distribution, the density is symmetric about the mean of all training samples which is unfavorable for features with high variance. Nevertheless, the parameters can be computed at lower computational cost than the parameters of the MGD and many existing machine learning frameworks support this distribution for continuous features. The parameters of both distributions can be updated in an online learning methods.

Perception Control in the Kitchen Domain

The objective of this thesis is to adapt a particular expert system architecture (PLAKON [CG91]) for the context of perception in the kitchen domain. Furthermore, it is intended to integrate a selection of computational attention methods (discussed in section 3) into the expert system architecture. There is a rich set of methods for the computational attention research topic, but most of these methods focus on feature based attention without knowledge about objects or relations between them. Usually, the methods of computational attention describe how the conspicuity of features, regions or objects can be estimated. To the best of my knowledge, there was no attempt made yet to integrate these methods into a well defined expert system architecture. Another objective of thesis is to interface with an existing perception component that is based on the Unstructured Information Management Architecture (UIMA). In UIMA, methods of the perception components are called analysis engines, which are modules that can be loaded dynamically. The analysis engines that are most relevant in the context of this thesis are discussed in section 2.2.3. The sequence of executed analysis engines can be specified using a UIMA component that is called *flow controller*. In order to interface with UIMA, a special flow controller is designed that uses the control mechanism internally. The existing perception component includes analysis engines for clustering of RGB-D input frames and for computation of annotations for clusters. The purpose of the control procedure is to dynamically select and configure the analysis engines that will be executed in the UIMA process. The selection is based on control knowledge and results of computational attention methods.

PLAKON (see [CG91]) is an expert system designed for planning and configuration problems in technical domains. The ontology-based control component of PLAKON (see [Gün92]) intends to solve and relax many problems that might occur in rule based control components: The mixture of different types of knowledge and the large number of rules might lead to problems in the areas of acquisition, maintenance, reasoning, adaptability, consistency and modularity [Gün92]. Following is a list of essential features that were claimed for the ontology-based control component of PLAKON:

- Strict separation of different types of knowledge (ordering, focusing, processing and conflict

resolving knowledge)

- Declarative representation of control knowledge
- Explicit control decisions (particular relevant for backtracking and reasoning methods)
- Modular architecture

The basic idea of the ontology-based control in PLAKON is that model concepts in the ontology can be used to represent prototypical descriptions of object classes. Dynamic instantiations of such classes must be compatible with the model concept description in the ontology. Each operation that can be performed by the ontology-based control of PLAKON corresponds to a specialization of a prototypical part of the model concept description. For example, the specialization of a numeric range to a specific number maps the prototypical range to a discrete value. The selection of the next specialization operation that should be performed belongs to the tasks of the ontology-based control component (ordering knowledge). The selection of the processing methods that should be used to execute the operation is separated from the selection of the next operation because it belongs to a different category of control knowledge (processing knowledge). Additionally, the ontology-based control supports to ignore particular operations, objects, attributes or relations entirely (focusing knowledge). The ontology-based control of PLAKON also supports constraints between entities: The specialization of one entity can have influence on possible specializations of other entities. For example, a milk package should not be a part of the breakfast setting if it is empty. Such restrictions can be expressed by constraints. Conflict situations are handled in PLAKON using conflict resolving rules and different backtracking techniques (including a truth maintenance system). The backtracking is used because the control component may have made wrong decisions based on false assumptions.

The architectural concept that is presented in this chapter is based on the ontology-based control of PLAKON. The most essential aspects were adapted for the control of perception in the kitchen domain. Some other aspects were not adapted because they are beyond of the scope of this thesis (e.g., constraint network, truth maintenance system). Rules are used as a replacement for the constraint network in this control framework. They have the advantage of better performance than constraint networks while being less expressive (multidirectional constraints vs. unidirectional rules). In PLAKON, conflict situations occur when the control component tries to correct a previous decision. This is important for planning and configuration in many technical domains because the construction may fail entirely if some decisions were made on false assumptions. For perception control, there are usually weaker relations between the characteristics of objects. Thus, backtracking is not necessarily required for perception control. In this chapter, the differences to the original ontology-based control of PLAKON, the integration of computational attention methods as well as the integration of perception methods and the interface between the control component and the perception component are discussed.

In the following section, I will discuss the ontology formalism that was used in the context of this thesis as well as I will give a description of how the kitchen domain for the perception

control procedure is modeled (see section 4.1). The control procedure is continuous, a central main loop is executed at regular intervals until the procedure terminates. This control main loop is discussed in section 4.2. Tasks need special treatment because they have influence on the relevance of entities for the perception procedure. It must be possible to overwrite and manipulate the current task in order to support continuous perception. The representation of tasks in the proposed control procedure is introduced in section 4.3. The core of the control mechanism is build upon a set of control strategies, which contain the control knowledge that should be used in particular situations. The dynamic activation (selection) of strategies is discussed in section 4.4. Each strategy covers different types of control knowledge: Focusing knowledge that is used to blank out particular aspects (see section 4.4.1), ordering knowledge that is used for the sequencing of the process (see section 4.4.2) and processing knowledge that is used for the selection and configuration of analysis engines and other processing methods.

4.1 Kitchen Domain Ontology for Perception Procedures

The control procedure that is proposed in this thesis is driven by the definition of model concepts in an ontology. The ontology formalism and implementation used in this thesis (see [Lan14]) is based on a diploma thesis that was written simultaneously to this thesis at the same research institution (artificial intelligence institution of the University Bremen ¹). In this section, I will give a brief formal definition of the ontology formalism as well as I will present the most important aspects of the hand coded kitchen domain ontology that was used in the scope of this thesis.

4.1.1 Ontology Formalism

In general, ontologies are used to represent knowledge about things, their attributes and relations to other things. Model concepts are prototypical descriptions of object classes. Each concept includes a set of attributes where each prototypical value represents the value domain of the particular attribute. For example, the value domain of an attribute *position* that corresponds to a spatial location is a 3 dimensional vector of float values. Furthermore, the model concept description contains knowledge about relations to other concepts. For instance, the type hierarchy is represented with a relation (*is-a* relation). The type hierarchy is important in particular because attributes and relations are inherited along the type hierarchy. The root of the type hierarchy is a common base concept *Object*. Any other model concept must be a child concept of the *Object* concept in the type hierarchy. Figure 4.1 illustrates a basic type hierarchy for the kitchen domain.

¹<http://ai.uni-bremen.de/>

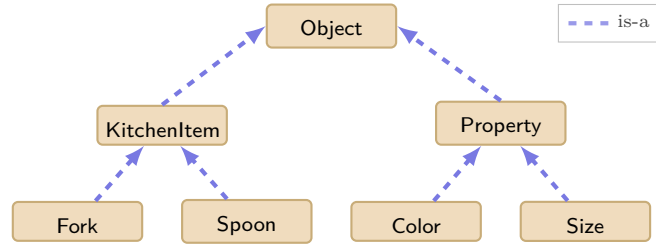


Figure 4.1 A basic type hierarchy of model concepts in the kitchen domain. All model concepts are child concepts of the common base concept *Object*. For example *Spoon is-a KitchenThing is-a Object*.

To formalize this, let $C_{\mathcal{D}}$ be the set of model concepts in ontology \mathcal{D} . The relation mapping $r : C_{\mathcal{D}} \rightarrow C_{\mathcal{D}}^*$ can be used to map a model concept to related model concepts with respect to the relation r . In the following, let $R_{\mathcal{D}} = \{C_{\mathcal{D}} \rightarrow C_{\mathcal{D}}^*\}$ be the set of relation mappings that are defined in ontology \mathcal{D} . Finally, attributes can be defined analogous using the mapping $a : C_{\mathcal{D}} \rightarrow \mathfrak{D}_a$ that maps a concept to the attribute domain \mathfrak{D}_a with respect to attribute a . The domain represents the prototypical attribute value that is defined in the model concept. In the following, let $A_{\mathcal{D}} = \{C_{\mathcal{D}} \rightarrow \mathfrak{D}_a\}$ be the set of attribute mappings which are defined in ontology \mathcal{D} .

Based on above discussion, the ontology \mathcal{D} can then be written as:

$$\mathcal{D} = (C_{\mathcal{D}}, R_{\mathcal{D}}, A_{\mathcal{D}}) \quad (4.1)$$

Let $c \in C_{\mathcal{D}}$ be a model concept. There is always a sequence of model concepts $c_0 \dots c_n$ with $c_0 \dots c_n \in C_{\mathcal{D}}$, $c = c_0, c_n = Object$ and c_i is-a c_{i+1} because *Object* is the common base concept in the type hierarchy.

Instantiations of model concepts are called model instances. They inherit the domains of attributes and relations from the instantiated model concept (model instances are mapped instead of model concepts). In the following, let $I_{\mathcal{D}}$ be the set of dynamic instantiations of model concepts in ontology \mathcal{D} and let $c : I_{\mathcal{D}} \rightarrow C_{\mathcal{D}}$ be the mapping from model instance to the corresponding model concept. The basic idea of the ontology-based control, that was proposed by Günter [Gün92], is that a comparison of dynamic instances with their corresponding model concepts can be used to find unspecified aspects of the entities. Operations that can be performed on model instances are used to specialize the particular instance in order to represent a smaller subset of the object class according to the model concept definition in the ontology. These operations are called agenda items in the context of ontology-based control. They are collected in a data structure that is called agenda (see sections 4.4.1, 4.4.2 and 4.4.3 for more details on this data structure). The area of responsibility of the control procedure includes management of the agenda, selection of items from the agenda, selection of agenda item processing methods and the execution of the operations based on the selection of processing methods.

Name	Inverse	Reflexive	Transitive	Agenda	Property
<i>is-a</i>	<i>is-a-inv</i>	1	1	0	0
<i>is-a-inv</i>	<i>is-a</i>	0	0	0	0
<i>contains</i>	<i>contained-in</i>	0	0	1	0
<i>contained-in</i>	<i>contains</i>	0	0	0	0
<i>location-of</i>	<i>located-at</i>	0	0	0	0
<i>located-at</i>	<i>location-of</i>	0	0	1	0
<i>annotation-of</i>	<i>has-annotation</i>	0	0	0	0
<i>has-annotation</i>	<i>annotation-of</i>	0	0	1	1

Figure 4.2 Relations that are used in the kitchen domain ontology for the control of the perception procedure. For each reflexive relation *rel*, *a rel a* holds true for all entities. For each transitive relation *rel*, *a rel b ∧ b rel c → a rel c* holds true for all entities. The *Agenda* flag is used to tag relations that should be processed by the control component by dynamically relating entities to each other. Finally, the *Property* flag tags relations that are supposed to be used to augment the attributes of entities.

4.1.2 Ontology for Perception in Kitchens

The kitchen domain is highly versatile and the set of characteristic items depends on the culture and other factors (see section 2.2.1). It is beyond the scope of this thesis to model all possible kitchens that could appear in European households. It is sufficient to model kitchen items that occur in one of the reference scenarios which were defined in section 2.3 in order to show that the proposed control mechanism fulfills the requirements that were defined in section 2.4. Furthermore, the ontology has to model annotation types of the perception component: Results of perception methods must be reflected in the ontology in order to be able to use the results in the control procedure.

A relation definition consists of a relation name and a set of boolean properties. The semantic of relations is not important to the control procedure (except for the type hierarchy relation). New relations can be introduced without modification of the source code. The type hierarchy relation – the *is-a* relation – has some special properties like the inheritance of attributes and relation descriptors along the type hierarchy. In the scope of this thesis, containment is expressed using the *contains* relation and the location of kitchen items is expressed using the *location-of* relation. Furthermore, relations can be flagged to be property relations. Property relations are used to attach augmentations to model instances. This fits well with the concept of annotations in the perception component: Annotations are attached to recognized objects in order to augment the set of object properties. In the kitchen domain ontology, annotations are attached to kitchen items using the *has-annotation* relation. Another relation property determines if agenda items should be generated for the particular relation if it is underspecified. The set of relations that I used in the scope of this thesis is shown in figure 4.2.

In the perception component, unstructured sensory input data is analyzed in order to find and identify visible objects. This clustering procedure is based on low-level features of the sensory

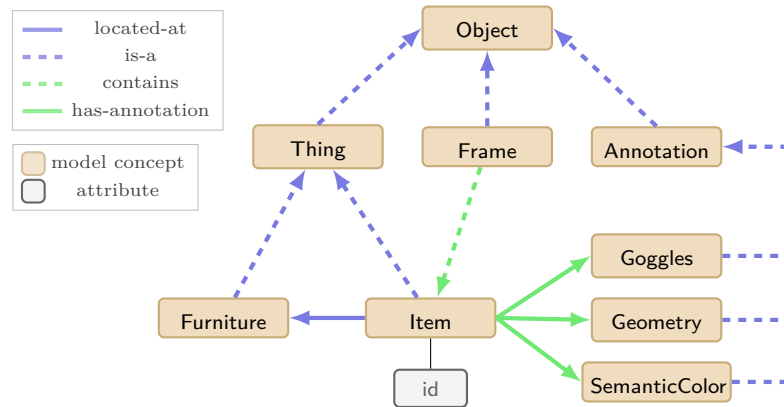


Figure 4.3 Relation hierarchy and attributes of the *Item* model concept. The item class is the base class for all kitchen items that might occur in one of the test scenarios (see section 2.3). The model concept is dynamically instantiated based on the result of a clustering method.

signal (e.g., the perceived depth). This procedure is driven by model concepts in the context of the proposed control architecture. In the context of this thesis, a special concept *Frame* is used to initiate the segmentation procedure. The objective of the clustering phase is to instantiate the *Item* concept for each recognized object. Each *Item* instance corresponds to a cluster that was recognized by the perception component. The clustering is initiated by the relation *Frame contains Item*. Agenda items are generated for this operation because the *Agenda* flag is set to *true* for the *contains* relation (see figure 4.2). Annotations are attached to the *Item* concept using the *has-annotation* relation. The parametrization of *Annotation* concepts initiates perception methods of the perception component in order to obtain the attribute value for a particular attribute and model instance. Finally, the spatial location of recognized kitchen objects is represented using the *located-at* relation. The perception methods are capable to automatically specify this relation for recognized objects by using a region filter that cuts out all points outside of a region that corresponds to a semantic location. As an illustration, the relation hierarchy and attributes of the *Item* concept are shown in figure 4.3.

Specializations of the *Item* concept inherit the relations to the annotation concepts from the definition of the *Item* concept. Generally, model concepts can restrict the inherited relations by specification of attribute specializations of related concepts. In the case of annotations, it is hard to define such specializations because kitchen items can appear in many manifestations with different color, shape and size. Thus, related annotations are usually not restricted by *Item* concept specializations. In the scope of this thesis, there are model concepts defined for each kitchen item that occurs in one of the test scenarios (see section 2.3).

Annotations are used to augment the properties of *Item* concept instantiations. They correspond to methods of the perception component (see section 2.2.3). The annotations which were used

in the context of this thesis are listed below:

- *Geometry*
The *Geometry* annotation contains a 'shape' attribute (one of *flat*, *box*, *round* or *cylinder*) that represents the rough geometric shape of the object and a 'size' attribute (one of *small*, *medium* or *large*) that represents the rough size of the object (according to the author of the perception method who determined which actual sizes correspond to which size category). This annotation model concept corresponds to the perception methods *Cluster3DAnnotator* and *PrimitiveShapeAnnotator*.
- *SemanticColor*
This annotation is used to represent the dominant color of an object. The dominant color is represented by a color attribute (a prioritized list that can contain following constants: *blue*, *yellow*, *red*, *green*, *white* and *black*). Additionally, this annotation contains an attribute for the color histogram of the object. The corresponding perception method is the *ClusterColorHistogramCalculator*.
- *Goggles*
The *Goggles* annotation represents the result of a *Google Goggles*² query (the perception method reverse engineered the *Goggles* communication protocol because there is no public API). The color information of the visible region that corresponds to a particular object is transmitted to the *Goggles* service in order to retrieve information about the visible object. The result of the query contains string attributes for the category and the title of the object which is visible in the cluster region according to *Google Goggles*. The *ClusterGogglesAnnotator* perception method corresponds to this annotation concept.
- *LinemodDetection*
The *LinemodDetection* annotation contains a string attribute that represents the class label of the object and a numeric attribute that represents the confidence that this particular objects belongs to the class that is specified in the class label attribute. This annotation corresponds to the *LinemodAnnotator* perception method.

Finally, a set of furniture objects is represented in the kitchen domain ontology. Furniture objects specify a bounding box that is used to filter away other scene regions. The filtering is done by the perception method *URDFRegionFilter*. In the scope of this thesis, the only furniture that is investigated is the kitchen table (*Table* concept) in the reference kitchen (see section 2.3).

²<http://www.google.com/mobile/goggles>

4.2 The Perception Control Loop

In this section, I will give a description of the control main loop that is executed in order to successively specialize different aspects of the set of instantiated model concepts. This procedure is based on the control main loop that was used in the control procedure of PLAKON (see [Gün92]). In the following description, I will focus on the differences of my approach to the procedure that was proposed by Günter [Gün92].

The initial setup is done based on following steps:

(a) *Instantiation of Known Entities*

In the scope of this thesis, the furniture that occurs in kitchens is expected to be completely specified in advance. Thus, attributes of the table, the fridge and other furniture objects are known in advance. The corresponding model concepts are instantiated and specialized in this step. This has to be done only once in the initialization step. No such mechanism exists in PLAKON.

(b) *Instantiation of Task Entities*

Tasks are defined slightly different in the proposed control procedure: They can contain multiple independent task entities as well as relations between referenced entities (which is not allowed in PLAKON). PLAKON allows dynamic augmentation of the task, but it is not possible to change the target object dynamically. In PLAKON, a so called *task question* agenda processing method is used for the task augmentation. The *task definition* agenda processing method in this control procedure additionally allows to add and remove task entities dynamically (see section 4.3).

The proposed control loop consists of following steps:

1. *Activation of Loop Rules*

Loop rules are rules that belong to the ‘*loop*’ rule group. They can have arbitrary conditions and consequences. This rule group is activated at the beginning of each control loop and multiple rules are allowed to fire for each activation of the rule group. The control procedure in PLAKON does not use such a rule group.

2. *Selection of Active Phase*

Usually control tasks can be segmented into distinct phases where different control knowledge is required for different phases. As in PLAKON, the selection of the active phase is done using control rules. Only the activated rule with highest priority fires in this step. In the proposed procedure, all phase activation rules must belong to the rule group ‘*strategy*’ that is dedicated to the selection of strategies (see section 4.4).

3. *Agenda Generation*

The agenda is generated by comparison of model instances with concept definitions in the ontology. Agenda items are generated for underspecified aspects of the instances (i.e., unspecified attributes, relations, specializable type). As in PLAKON, agenda focus methods (see section 4.4.1) and agenda ordering methods (see section 4.4.2) can be used to exclude and sort agenda items. I introduce hierarchical agendas: A root agenda is used that contains all generated agenda items and child agendas that inherit the set of items from the parent agenda. This is done in order to allow different components to process different aspects of the agenda in parallel.

4. *Selection of Agenda Items*

The selection of agenda items is done identical to the approach in PLAKON: A prioritized sequence of selection criteria is used to sort the agenda and the agenda item with highest priority is selected. In this step, unique agenda items are automatically processed and invalid items are dropped from the agenda. Agenda items are invalid if none of the control methods was able to execute the corresponding operation. The selection of agenda items is discussed in more detail in section 4.4.2.

5. *Selection of Agenda Item Processing Methods*

In PLAKON, the selection of agenda item processing methods is done based on a prioritized list of selection methods that is attached to the control knowledge of phases. The methods are executed in the priority order until a method yields in a consistent result for the operation of the selected agenda item. I propose an augmentation of this selection mechanism that is based on agenda item patterns. In this procedure, multiple prioritized sequences of processing methods can be attached to the control knowledge of a phase. Agenda item patterns are used to select which one of the declared sequences should be activated (see section 4.4.3).

6. *Processing of Agenda Items*

In the context of ontology-based control, processing of agenda items corresponds to the specialization of model instances based on model concept definitions from an ontology. This includes the specialization of instance concepts, specialization of instance attributes and the relations to other instances. In the proposed control mechanism, the selected sequence of processing functions is always processed completely. Methods can process the selected agenda item, an arbitrary other agenda item or none of the generated items at all. Additionally, the perception application domain requires a special set of agenda processing methods which allow parametrization and invocation of perception methods which are used in the perception process. The agenda processing methods are described in more detail in section 4.4.3.

4.3 Ontology-Based Task Representation

The control procedure must be capable to represent different tasks that are required in the kitchen domain. For instance, the robot may have to visually find a red cup on top of a specific table in the kitchen. In the proposed procedure, the task description $\mathcal{T}_\mathfrak{D}$ is an underspecified instantiation of model concepts declared in ontology \mathfrak{D} . $\mathcal{T}_\mathfrak{D}$ consists of following aspects:

- *Task Entities*

Model instances that are targets of the current task are called task entities. Task entities $t \in I_{\mathcal{T}_\mathfrak{D}} \subseteq I_\mathfrak{D}$ are instantiations of model concepts in ontology \mathfrak{D} that are referenced by task description $\mathcal{T}_\mathfrak{D}$. Let $C_{\mathcal{T}_\mathfrak{D}}$ be the set of model concepts that are referenced in the task description. A task entity $t \in I_{\mathcal{T}_\mathfrak{D}}$ is instantiated for each concept $c \in C_{\mathcal{T}_\mathfrak{D}}$. For example, when a concept Cup is referenced in $\mathcal{T}_\mathfrak{D}$, then a task entity $t_{Cup} \in I_{\mathcal{T}_\mathfrak{D}}$ is instantiated with $c(t_{Cup}) = Cup$, where c maps a model instance to the corresponding model concept.

- *Task Entity Relations*

Task entity relations are used to specify relations between instances in $I_{\mathcal{T}_\mathfrak{D}}$. A relation between model instances of $I_{\mathcal{T}_\mathfrak{D}}$ can be interpreted as a (named) mapping $r : I_{\mathcal{T}_\mathfrak{D}} \rightarrow I_{\mathcal{T}_\mathfrak{D}}^*$ that maps a task entity to related instances according to relation r . For example, the red cup relates to the kitchen table using the *located-at* relation. In the following, let $R_{\mathcal{T}_\mathfrak{D}} = \{I_{\mathcal{T}_\mathfrak{D}} \rightarrow I_{\mathcal{T}_\mathfrak{D}}^*\}$ be the set of relations declared in task description $\mathcal{T}_\mathfrak{D}$.

- *Task Entity Attributes*

Task entity attributes are used to specify attribute values for task entities. Attributes of model instances in $I_{\mathcal{T}_\mathfrak{D}}$ can be interpreted as a (named) mapping $a : I_{\mathcal{T}_\mathfrak{D}} \rightarrow \mathfrak{D}_a$ that maps a task entity to the attribute value domain \mathfrak{D}_a . The attribute domain of a is defined in ontology \mathfrak{D} . For instance, the red color of the target cup is represented in the task description by an attribute specialization (the dominant color is set to *red*). In this case, the attribute domain is a selection set of pre-defined color constants (*red, green, blue, ...*). In the following, let $A_{\mathcal{T}_\mathfrak{D}} = \{I_{\mathcal{T}_\mathfrak{D}} \rightarrow \mathfrak{D}_a\}$ be the set of attribute specializations that are declared in task description $\mathcal{T}_\mathfrak{D}$.

Formally, the task description can be represented by following equation:

$$\mathcal{T}_\mathfrak{D} = (I_{\mathcal{T}_\mathfrak{D}}, R_{\mathcal{T}_\mathfrak{D}}, A_{\mathcal{T}_\mathfrak{D}}) \quad (4.2)$$

Contrary to the control procedure in PLAKON where the subject is to specify one particular task entity, the task description of the proposed control procedure can contain multiple task targets. For example, multiple targets for visual search can be specified in the task description.

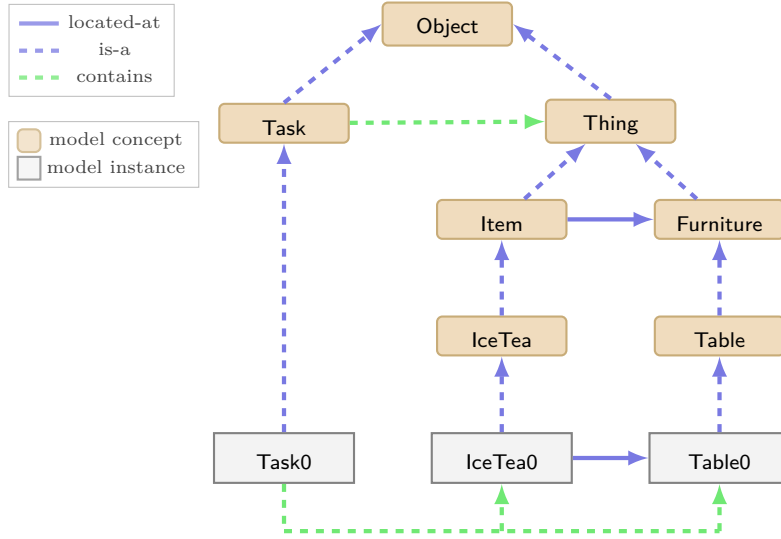


Figure 4.4 Example relation hierarchy for task entities. $Task0$ is the model instance of the $Task$ concept. All other instances that are related to the task instance are task entities. In this example, $IceTea0$ and $Table0$ are task entities because $Task0$ contains $IceTea0$ and $Task0$ contains $Table0$.

Furthermore, the task entities must not be related to each other in the decomposition hierarchy, but they are usually related to each other using spatial relations (e.g., both items are on top of the same table). Furthermore, the task description can reference externally specified instances. In above example, the table might be externally specified in advance because the attributes of the table might not change for different tasks. Finally, the proposed task representation is augmented compared to the PLAKON procedure by the possibility to specify relations between task entities.

In order to expose the information which instances in $I_{\mathcal{D}}$ are task entities to other control methods, the proposed control procedure uses a special task concept $c_{task} \in C_{\mathcal{D}}$. The control procedure instantiates this task concept once. The task instance $i_{task} \in I_{\mathcal{D}}$ with $c(i_{task}) = c_{task}$ is used to tag task entities as referenced in the task description. This is done by relating the task instance i_{task} with all task entities in $I_{\mathcal{T}_{\mathcal{D}}}$ using the *contains* relation: $I_{\mathcal{T}_{\mathcal{D}}} = contains(i_{task})$. This relation can also be expressed with the inverse relation *contained-in*:

$$i_{task} \in contained-in(t) \quad \forall t \in I_{\mathcal{T}_{\mathcal{D}}} \quad (4.3)$$

4.4 Control Knowledge for Perception Procedures

Strategies represent the control knowledge that is used in the proposed control procedure. The perception process can be segmented into different phases where different entities, attributes and relations are relevant for the perception process. In the scope of this thesis, the following set of phases are used:

- *Input-Phase*

In this phase, a sensory frame is written into the data store that is used by the perception component. Initially, the raw sensory data (RGB image and a point cloud) is filtered based on the semantic location of interest in this situation. In the next step, a clustering method is used in order to find and identify visible objects.

- *Annotation-Phase*

The annotation of recognized objects is done in this phase. This includes the decomposition of property relations. For the control procedure, the annotation procedure consists of two steps: Execution of a perception method and the mapping of results to relations and attributes of model instances.

- *Classification-Phase*

The clustering method leads to instantiations of an abstract model concept. The type of these model instances is specialized in this phase. Generally, it is desirable that successive specialization of model instances leaves only a single possible type specialization before all attributes and relations are completely specified. In the application domain of this thesis, where kitchen items may appear in a great variety of sizes, shapes and colors, where sensory input data with signal noise is used and where probabilistic perception methods are used, this (automatic) specialization is not reliable. Thus, a probabilistic approach is used in the context of this thesis.

- *Fusing-Phase*

The goal of the control procedure for the perception component is to focus the attention of the robot on recognized objects that belong to the current task of the robot (task entities). In the context of this thesis, recognized objects must be directly related to the task instance to be a task entity. In the *Fusing-Phase*, recognized objects are fused with task entities under certain conditions for that purpose. The resulting model instance represents the most special aspects of both fused model instances.

Strategies are activated with prioritized activation rules. Only the strategy activation rule with highest priority fires when the conditions of multiple rules are met. Strategy activation rules can

<p><i>Condition</i></p> <ul style="list-style-type: none"> • There is an instance of concept <i>Item</i> assigned to <i>item</i> • - with specializable type • - with unspecified <i>has-annotation</i> relation • There is no instance of concept <i>Item</i> • - with unspecified <i>is-located-on</i> relation <p><i>Consequence</i></p> <ul style="list-style-type: none"> • Activate stage <i>Annotation-Phase</i>

Figure 4.5 Strategy activation rule that activates the strategy *Annotation-Phase* if there is an instantiation of the *Item* concept with unspecified annotations and if the *is-located-on* relation is specified for all *Item* instances (this relation is resolved in the clustering phase).

query model instances and model concepts in the condition part of the rule. In the consequence part, the strategy activation rules usually just activate a particular strategy. Each strategy can have multiple activation rules associated to it. The activation rules represent the meta-control: Knowledge that is used to select the active control knowledge. As an illustration, an activation rule for the *Annotation-Phase* is shown in figure 4.5.

In the proposed control procedure, another rule group for the activation of so called *loop* rules is used. This rule group is activated once at the beginning of each control loop. It is used for the streaming of sensory input data: A special concept is instantiated that leads to an activation of the input phase.

The strategy definition that is used in the scope of this thesis is missing some aspects that were used in the control procedure of PLAKON. This is due to missing features compared to PLAKON: In the scope of this thesis, backtracking and constraints were not investigated. Thus, knowledge regarding those aspects is not represented in strategies. The most important aspects of strategies are focusing knowledge (see section 4.4.1), ordering knowledge (see section 4.4.2) and processing knowledge (see section 4.4.3).

4.4.1 Perception Control Agenda

Agenda items are generated for unspecified aspects of model instances. Dynamic instances of model concepts can have unspecified aspects for following reasons:

- *Specialization*

A model instance $i_0 \in I_{\mathcal{D}}$ can be specialized if the corresponding concept $c_0 = c(i_0) \in C_{\mathcal{D}}$ can be specialized. Thus, if there is a concept $c_1 \in C_{\mathcal{D}}$ with $is-a(c_1) = \{c_0\}$. The set of possible concepts for the specialization operation is defined in the ontology \mathcal{D} .

Example: The robot recognizes a small red object on the kitchen table without knowing the particular object class. This leads to an instantiation of a generic concept that is used as parent concept for all items that might occur in kitchens.

- *Parametrization*

A model instance can be specialized if it has an attribute that can be specialized. The specialized attribute value may be a concrete value or a restriction of the attribute domain. For example numeric ranges can be specialized to specific numbers or to a subset of the range.

Example: The domain of the position of a kitchen item (i.e., the centroid) is a 3-dimensional vector of float ranges. The specification of this attribute is a 3-dimensional vector of concrete float values.

- *Relation*

A model instance can be specialized if there is an underspecified relation. Each relation has a minimum number of instances that are required to be related to the corresponding model concept. If the number of related instances is less than the minimum number of required instances, then the relation is underspecified.

Example: There may be zero or more items on kitchen tables. Thus, this particular relation is specified independent of the number of related instances.

In the proposed control procedure, so called agenda item generators are used to dynamically examine the model instances by comparing them with the corresponding model concepts. Agenda items are generated for each atomic operation that can be performed on the instances regarding to the agenda item generator implementation. To formalize agenda items, let $\mathfrak{A}_s = \{i \in I_{\mathcal{D}} \mid \exists x \in C_{\mathcal{D}} : is-a(x) = \{c(i)\}\}$ be the set of specialization agenda items, $\mathfrak{A}_p = \{(i, a) \in I_{\mathcal{D}} \times A_{\mathcal{D}} \mid a(i) \text{ is defined and not unique}\}$ be the set of parametrization agenda items and let $\mathfrak{A}_r = \{(i, r) \in I_{\mathcal{D}} \times R_{\mathcal{D}} \mid r(i) \text{ is defined and not unique}\}$ be the set of relation agenda items. The set of all expressible agenda items \mathfrak{A} can then be written as:

$$\mathfrak{A} = \mathfrak{A}_s \cup \mathfrak{A}_p \cup \mathfrak{A}_r \quad (4.4)$$

I propose to use a hierarchy of agendas. Each agenda consists of a set of agenda items and a set of selection criteria (see section 4.4.2) that is used to sort the agenda items. The root agenda $\mathfrak{A}_{root} \subset \mathfrak{A}$ contains an unsorted list of all generated agenda items. All other agendas – called agenda views – inherit the items of the corresponding parent agenda. Additionally, agenda views can filter items of the parent agenda based on an agenda item pattern (see section 4.4.1.1). The agenda focus can be defined as boolean mapping $filter : \mathfrak{A} \rightarrow \text{BOOL}$ from agenda items to *true* if the agenda item is filtered and to *false* otherwise. Formally, an agenda view \mathfrak{A}_{view} is defined

as:

$$\mathfrak{A}_{view} = \{x \in \mathfrak{A}_{parent} \mid filter(x) = false\} \quad (4.5)$$

Where \mathfrak{A}_{parent} is the parent agenda of \mathfrak{A}_{view} .

Each strategy declares a set of agenda selection criteria as well as an agenda focus. Thus, each strategy has a distinct agenda view. All agenda views are managed in parallel. This allows fast switching between strategies with the drawback of higher computational costs when a strategy is active for a long time. Contrary, only the agenda of the currently active strategy is managed by the control procedure in PLAKON.

4.4.1.1 Agenda Focus

The agenda focus is used to filter agenda items that are irrelevant in the current stage of the control procedure. For example, only parametrization agenda items might be relevant if the subject of the current control stage is the annotation of various recognized objects on the kitchen table.

The agenda item pattern does not differ notably from the agenda item pattern used in PLAKON. It allows to specify the agenda item type, a model concept, particular slots (attributes and relations) and aggregate concepts (which include related concepts). In the proposed control procedure, the agenda focus pattern is augmented by conjunction and disjunction of agenda foci. Let $P = \{\mathfrak{A} \rightarrow BOOL\}$ be the set of expressible patterns which map agenda items to *true* if the pattern matches and to *false* otherwise. For the pattern focus $f_p : \mathfrak{A} \rightarrow BOOL$, following holds true: $f_p(x) = p(x)$. Finally, let $F = \{\mathfrak{A} \rightarrow BOOL\}$ be the set of expressible agenda foci. The conjunction focus $c \in F$ and the disjunction focus $d \in F$ can be written as:

$$\begin{aligned} c_{fg}(x) &= f(x) \wedge g(x) \\ d_{fg}(x) &= f(x) \vee g(x) \end{aligned} \quad (4.6)$$

Where $f, g \in F$ are arbitrary agenda foci.

4.4.2 Selection of Perception Methods and Recognized Objects

Agenda items represent atomic operations that can be performed on model instances. This allows flexible selection of operations based on control knowledge estimations of selection criteria. Furthermore, the agenda focus (see section 4.4.1.1) can be used to filter agenda items which are irrelevant in the current stage of the control procedure. For each agenda view \mathfrak{A}_{view} there is an ordered sequence of selection criteria $sel_0 \dots sel_n$ defined with $sel_i \in \{\mathfrak{A} \rightarrow \mathbb{R}\}$. The sequence

of selection criteria is ordered by priority. High selection values correspond to high priority for the processing of the particular item. Let $pos : \mathfrak{A} \rightarrow \mathbb{N}$ be a mapping from agenda item to the position of the item in the agenda (0 is the position of the item that will be processed next). The agenda view is sorted so that $i_0 \in \mathfrak{A}$ is positioned before $i_1 \in \mathfrak{A}$ if $sel(i_0) \geq sel(i_1)$. The item comparison procedure is shown in algorithm 1.

Algorithm 1 Agenda Item Comparison

```

1 function COMPARE( $i_0, i_1, sel_0 \dots sel_n$ )           ▷ For  $i_0, i_1 \in \mathfrak{A}, n \in \mathbb{N}, n > 0$ 
2   if  $sel_0(i_0) \neq sel_0(i_1)$  then
3     return  $sel_0(i_0) - sel_0(i_1)$ 
4   else
5     return COMPARE( $i_0, i_1, sel_1 \dots sel_n$ )       ▷ Check next criterion
6 function COMPARE( $i_0, i_1, sel_0$ )
7   return  $sel_0(i_0) - sel_0(i_1)$ 

```

The selection criteria represent ordering knowledge that is used to affect the sequence of processed agenda items. The most essential selection criterion is the selection by agenda item patterns. The pattern used for agenda item selection does not differ notably from the one used in the PLAKON control mechanism. The pattern selection criterion assigns a value of 1 to all agenda items which match the pattern and a value of 0 to all other agenda items. Agenda item patterns may appear multiple times in the sequence of selection criteria. Another basic selection criterion – that was used in the control mechanism of PLAKON – is the selection by continuity where a value of 1 is only assigned to the last modified model instance. A value of 0 is assigned to all other model instances.

Contrary to the control mechanism of PLAKON, the proposed procedure does not require that it is possible to process agenda items. It may be the case that an agenda item cannot be processed in a particular situation. For example, the robot may need to look at an object from another view point in order to find out the type of that object. I propose to use a simple inhibition selection method that assigns negative values to selected agenda items in order to avoid that the control procedure continuously tries to process the same agenda item.

The selection criteria allow to integrate top-down methods of computational attention that were discussed in section 3. This is possible because most methods basically have influence on the ordering and parametrization of methods involved in the perception process. This ordering knowledge can be expressed using selection criteria. For instance, a bottom-up saliency selection method could assign high values to agenda items of model instances with high saliency based on attribute values and relations of the instance. In the following, two of the attention methods are integrated in the expert system architecture using the selection criteria interface. First, a relational task relevance estimation approach (see section 4.4.2.1) and second a probabilistic model instance classification approach (see section 4.4.2.3).

As an illustration, a sequence of selection criteria – as they could be used for the annotation of

Agenda Selection Criteria

1. Prefer items which were processed less often
2. Prefer the last modified model instance
3. Prefer items similar to a task entity
4. Prefer items with relations to task entities
5. Prefer *Relation* agenda items
6. Prefer the *Geometry* model concept
7. Prefer the *SemanticColor* model concept

Figure 4.6 A sequence of selection criteria that could be used for the annotation of recognized entities in the perception process. The list is ordered by priority.

previously recognized kitchen items – is shown in figure 4.6.

4.4.2.1 Relational Task Distance

The kitchen environment can be highly cluttered and many objects without relevance for the current task can appear. For instance, all objects on a specific table are directly relevant for the task when the robot is supposed to select a object that is on top of the table in order to accomplish the current task. Objects with a relation *located-at* to the table are particular relevant in the context of this task. Contrary, items that are not located on top of the table are irrelevant in this scenario (as long as they do not interfere with the actions of the robot).

The relational task distance approach that is proposed in this section is inspired by the computational attention approach that was discussed in section 3.1.4 (task relevance graph). Model instances are interpreted as nodes and relations between model instances are interpreted as edges in the graph. Task relevance can then be estimated by finding a path in the model instance graph that connects the instance of interest with a task entity. The relational task relevance of a model instance is anti proportional to the distance of the best path from the the model instance to an arbitrary task entity. The transition cost between concepts is given by a user defined function that maps relations to a cost value. Contrary to the approach that was discussed in section 3.1.4, this approach does not depend on a particular syntax for tasks (a task consists of object, subject and action in the task relevance graph approach). This allows a more flexible handing of tasks which is required in the kitchen domain (e.g., there is no subject during navigation). In the task relevance graph approach, transitions are weighted based on the co-occurrence of objects. This is useful in order to enhance the paths of entities which occur often together with one of the task objects. The weighting based on object co-occurrence of objects was not investigated in the scope of this thesis because it is sophisticated and time intense to gather representative training data for the kitchen domain.

I propose to use two graphs for the task relevance estimation: A static graph of model concept nodes and relation edges that can be used to find the distance between model concepts and a dynamic graph of model instance nodes and relation edges that can be used to find the distance between model instances. The concept graph and the instance graph are formally defined in following paragraphs.

Concept Graph Ontologies can be interpreted as directed graphs where each node corresponds to a model concept and where each edge corresponds to a relation between concepts. Let \mathfrak{D} be an ontology, $C_{\mathfrak{D}}$ the set of concepts and $R_{\mathfrak{D}} = \{C_{\mathfrak{D}} \rightarrow C_{\mathfrak{D}}^*\}$ the set of relations declared in \mathfrak{D} . Relations are used to map concepts to the set of related concepts. Let further $E_{\mathfrak{D}} \subset C_{\mathfrak{D}} \times R_{\mathfrak{D}} \times C_{\mathfrak{D}}$ be the set of relation edges. Each edge is specified by the source concept, the relation and the target concept. The edges can be defined based on concepts and relations in \mathfrak{D} :

$$E_{\mathfrak{D}} = \{(c_0, r, c_1) \in C_{\mathfrak{D}} \times R_{\mathfrak{D}} \times C_{\mathfrak{D}} \mid c_1 \in r(c_0)\} \quad (4.7)$$

Finally, let $s_{\mathfrak{D}} : E_{\mathfrak{D}} \rightarrow C_{\mathfrak{D}}$ be the mapping from edges to the source nodes of the edges, $t_{\mathfrak{D}} : E_{\mathfrak{D}} \rightarrow C_{\mathfrak{D}}$ the mapping from edges to the target nodes of the edges and $r_{\mathfrak{D}} : E_{\mathfrak{D}} \rightarrow R_{\mathfrak{D}}$ the mapping from edges to the relations that corresponds to the edges. The concept graph $G_{\mathfrak{D}}$ can then be written as:

$$G_{\mathfrak{D}} = (C_{\mathfrak{D}}, E_{\mathfrak{D}}, s_{\mathfrak{D}}, t_{\mathfrak{D}}, r_{\mathfrak{D}}) \quad (4.8)$$

The concept hierarchy is acquired by a knowledge engineer and the task relevance is estimated in applications based on this previously modeled concept hierarchy. Thus, the concept hierarchy does not change during run time in the task relevance estimation process. I propose to compute all shortest paths with respect to an user specified relation weighting in advance using the Floyd-Warshall algorithm [Flo62]. The algorithm has cubic complexity in the number of nodes of the graph ($O(\#C_{\mathfrak{D}}^3)$) and it must only be processed once in the initialization process because the structure of the graph does not change during run time. After initialization, the distance between two concepts can be queried in constant time. In summary, the Floyd-Warshall algorithm initially sets the distance between two different nodes to the minimal weight of relations between the nodes. The distance is set to infinity if there are no relations between the concepts. This can be expressed using the distance function $dist_{C_{\mathfrak{D}}} : C_{\mathfrak{D}} \times C_{\mathfrak{D}} \rightarrow \mathbb{R}$:

$$cost_{C_{\mathfrak{D}}}(c_0, c_1) = \begin{cases} \min_{r \in X} \alpha_r & \text{if } X = \{r \in R_{\mathfrak{D}} \mid c_1 \in r(c_0)\} \neq \emptyset \\ \infty & \text{else} \end{cases} \quad (4.9)$$

Where α_r is a user defined weighting factor for relation r . In the following, the algorithm subsequently allows previously unused nodes on paths between two nodes and updates the minimum distance if a better path was found. This leads to the computation of all shortest paths in $G_{\mathfrak{D}}$.

Instance Graph Model instances are dynamic instantiations of model concepts. Thus, the graph concept can be applied to instances analogues where each node corresponds to a model instance and where each edge corresponds to a relation between instances. Let $I_{\mathcal{D}}$ be the set of dynamic instantiations of model concepts in \mathcal{D} and let $R'_{\mathcal{D}} = \{I_{\mathcal{D}} \rightarrow I_{\mathcal{D}}^*\}$ be the set of relations between instances in $I_{\mathcal{D}}$. The edges between instances $E'_{\mathcal{D}} \subset I_{\mathcal{D}} \times R'_{\mathcal{D}} \times I_{\mathcal{D}}$ can then be defined as:

$$E'_{\mathcal{D}} = \{(i_0, r, i_1) \in I_{\mathcal{D}} \times R'_{\mathcal{D}} \times I_{\mathcal{D}} \mid i_1 \in r(i_0)\} \quad (4.10)$$

Algorithm 2 Dynamic Best Task Path

```

1  $d : I_{\mathcal{D}} \rightarrow \mathbb{R}$                                 ▷ Maps model instances to cost of best task path
2  $tasks \leftarrow \emptyset$                                 ▷ The set of task entities
3  $instances \leftarrow \emptyset$                             ▷ All other model instances
4 function ADD-TASK( $t$ )                                    ▷  $t$  is a task entity
5    $d(t) \leftarrow 0$                                     ▷ Distance to task entity is zero
6   for  $i \in instances$  do                                ▷ Add concept graph edge to all model instances
7     ADD-PATH( $t, i, \gamma + cost_{C_{\mathcal{D}}}(t, i)$ )
8   EXPAND-TASK( $t, \emptyset$ )                                ▷ Add edges to related instances
9    $tasks \leftarrow tasks \cup \{t\}$ 
10 function EXPAND-TASK( $e, visited$ )
11   if  $e \notin visited$  then
12      $visited \leftarrow visited \cup \{e\}$ 
13     for  $r \in R'_{\mathcal{D}}$  do                                ▷ For each relation mapping
14       for  $i \in r(e)$  do                                ▷ For each related instance
15         ADD-PATH( $e, i, cost_{I_{\mathcal{D}}}(e, i)$ )
16         EXPAND-TASK( $i, visited$ )
17 function ADD-ENTITY( $e$ )
18    $d(e) \leftarrow \infty$                                 ▷ Distance to task entity initialized to infinity
19   for  $t \in tasks$  do                                    ▷ Add concept graph edge to all task entities
20     ADD-PATH( $t, e, \gamma + cost_{C_{\mathcal{D}}}(t, e)$ )
21   for  $i \in instances \cup tasks$  do                    ▷ Add edges to related instances
22     for  $r \in R'_{\mathcal{D}}$  do                                ▷ For each relation mapping
23       if  $i \in r(e)$  then
24         ADD-PATH( $e, i, cost_{I_{\mathcal{D}}}(e, i)$ )
25       if  $e \in r(i)$  then
26         ADD-PATH( $i, e, cost_{I_{\mathcal{D}}}(i, e)$ )
27    $instances \leftarrow instances \cup \{e\}$ 
28 function ADD-PATH( $i_0, i_1, \delta$ )
29    $d(i_1) \leftarrow \min(d(i_1), d(i_0) + \delta)$         ▷ Update minimum distance
    
```

Each instance $i \in I_{\mathcal{D}}$ corresponds to a model concept $c \in C_{\mathcal{D}}$ (the type of the instance). Let $c : I_{\mathcal{D}} \rightarrow C_{\mathcal{D}}$ be the mapping from instances to corresponding concepts so that $c = c(i)$ holds true. I propose to add edges – called *concept graph edges* – between distinct model instances based on the best path between the corresponding model concepts in the concept graph $G_{\mathcal{D}}$. For that purpose, a relation $r_C : I_{\mathcal{D}} \rightarrow I_{\mathcal{D}}^*$ is introduced with: $r_C(i_0) = \{i_1\} \forall i_0, i_1 \in I_{\mathcal{D}}, i_0 \neq i_1$.

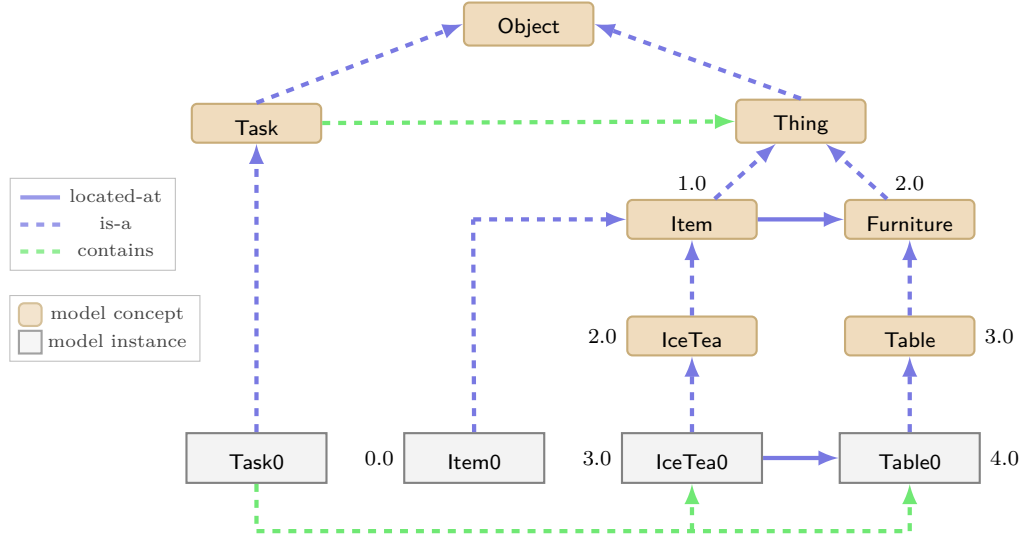


Figure 4.7 Relational task relevance example. The task relevance of model instance $Item0$ is estimated by finding the shortest path to a task entity. In this example, transition costs of 1.0 are assigned to each transition. This leads to a shortest path from the $Item0$ to the $IceTea0$ model instance with a distance of 3.0. The distance to $Item0$ is written next to nodes in the graphic.

Thus, there is a *concept graph edge* for each model instance pair in the instance graph. The set of edges introduced by relation r_C can be written as:

$$E''_{\mathcal{D}} = \{(i_0, r_C, i_1) \in I_{\mathcal{D}} \times R''_{\mathcal{D}} \times I_{\mathcal{D}} \mid i_1 \in r_C(i_0)\} \quad (4.11)$$

Where $R''_{\mathcal{D}} = \{r_C\}$ is the *concept graph relation*. Formally, the instance graph $G'_{\mathcal{D}}$ can then be written as:

$$G'_{\mathcal{D}} = (I_{\mathcal{D}}, E'''_{\mathcal{D}}, s'_{\mathcal{D}}, t'_{\mathcal{D}}, r'_{\mathcal{D}}) \quad (4.12)$$

Where $E'''_{\mathcal{D}} = E'_{\mathcal{D}} \cup E''_{\mathcal{D}}$ is the set of edges, $s'_{\mathcal{D}} : E'''_{\mathcal{D}} \rightarrow I_{\mathcal{D}}$ is the mapping from edges to the source nodes of the edges, $t'_{\mathcal{D}} : E'''_{\mathcal{D}} \rightarrow I_{\mathcal{D}}$ is the mapping from edges to the target nodes of the edges and $r'_{\mathcal{D}} : E'''_{\mathcal{D}} \rightarrow R'_{\mathcal{D}} \cup R''_{\mathcal{D}}$ is the mapping from edges to the relations that correspond to the edges.

It is important that there is a path to task entities for all instances in $I_{\mathcal{D}}$ because the relevance is directly computed based on the path distance. Since r_C adds an edge between all distinct instances, it is obvious that $G'_{\mathcal{D}}$ is fully connected.

The distance function $dist_{I_{\mathcal{D}}} : I_{\mathcal{D}} \times I_{\mathcal{D}} \rightarrow \mathbb{R}$ between related instances is defined as the minimum

cost of relations between them:

$$cost_{I_{\mathcal{D}}}(i_0, i_1) = \begin{cases} \min_{r \in X} \beta_r & \text{if } X = \{r \in R'_{\mathcal{D}} \mid i_1 \in r(i_0)\} \neq \emptyset \\ \infty & \text{else} \end{cases} \quad (4.13)$$

Where β_r is a user defined weighting factor for relation r . Finally, the distance function is redefined based on the distance between the corresponding concepts:

$$cost'_{I_{\mathcal{D}}}(i_0, i_1) = \min(cost_{I_{\mathcal{D}}}(i_0, i_1), \gamma + cost_{C_{\mathcal{D}}}(c(i_0), c(i_1))) \quad (4.14)$$

Where γ is a user defined cost for a transition from the instance graph into the concept graph.

Model instances are dynamically instantiated. Thus, a dynamic procedure for the computation of shortest paths is required. In the proposed algorithm, task entities and none task entities are handled separate. The cost of each entity is initialized to zero for task entities and to the minimum transition costs of the corresponding concept to a concept that corresponds to a task entity in $G_{\mathcal{D}}$ for none task entities. Finally, the costs are updated when instances are related to each other based on the cost function $cost_{I_{\mathcal{D}}}(i_0, i_1)$. The distance between instances is always less than infinity because $G_{\mathcal{D}}$ is fully connected. Algorithm 2 illustrates this procedure.

4.4.2.2 Edit Distance between Model Instances

In the proposed perception control mechanism, a simple similarity metric is used that is inspired by the edit distance [Lev66] between strings where the number of insert and delete operations is counted in order to estimate the similarity between strings. This similarity metric is supposed to be used in order to estimate the similarity of recognized objects and task entities based on a comparison of specified attributes and relations of the task entity.

The edit distance between a task entity $t \in I_{\mathcal{T}_{\mathcal{D}}}$ and a recognized object $x \in I_{\mathcal{D}}$ is computed based on annotations of the task entity (e.g., the specification of the color for the visual search target). Annotations are attached to task entities and recognized objects with the *has-annotation* relation. For each annotation of the task entity t , the minimum distance to an arbitrary annotation of x is computed. The similarity mapping $edit(\cdot)$ between model instances is defined as the normalized sum of these minimum distances:

$$edit(t, x) = \frac{1}{|A_t|} \sum_{a_t \in A_t} \min_{a_x \in A_x} distance(a_t, a_x) \quad (4.15)$$

Where $A_t = has-annotation(t)$ and $A_x = has-annotation(x)$.

The value of annotations is either represented by a string, a nominal value or a continuous value. The distance between feature values is computed as follows:

- *String Features*: The distance between string features is computed based on the Levenshtein distance [Lev66]. The distance is normalized by the maximum number of characters in the compared strings.
- *Nominal Features*: Nominal features are compared by equality. The normalized similarity is maximal (1.0) if both values are equal, otherwise the normalized similarity is minimal (0.0).
- *Continuous Features*: Continuous numeric attributes are normalized by an user defined maximum value. The distance between continuous features is defined as the difference of normalized values.

4.4.2.3 Similarity to Task Entities

Attention leads to selective selection of scene regions, objects and features that are relevant in the context of the current task. In the perception procedure, the type of objects might be unknown and attributes and relations might be unspecified for recently recognized objects (also called proto-objects; see section 3.1.1). Selective selection of proto-objects based on the similarity to task entities can be achieved by a naive Bayes' classifier (see section 3.2). Incremental annotation of proto-objects leads to changes in similarity to task entities. For instance, in the first step after clustering of proto-objects, the type is unknown and none of the attributes and relations are specified for the recognized proto-objects. All proto-objects have identical task similarity in this situation. In a later step, when some annotations has been computed for the proto-objects, there maybe some proto-objects with higher probability to be a task entity then others. For example, a proto-object with dominant blue color is more likely to be a task entity then a proto-object with dominant green color if the search target is a milk carton (at least in Germany, milk cartons are usually bluish).

For the task entity classification selection method, a frequently used naive Bayes' implementation is used [JL95]. Nominal and continuous features are supported by this implementation. Continuous features are handled using a Gaussian normal distribution (see equation 3.15). The mean and variance parameters of the distribution can be obtained from training samples with low computational overhead. Furthermore, the parameters can be dynamically updated without the need to process the complete training set again. The naive Bayes' implementation optionally allows to discretize continuous attributes. Unspecified model instance features are ignored during classification.

Initially, a feature vector $\mathcal{F}_C(c)$ must be obtained for each model concept $c \in C_{\mathcal{D}}$ that is supposed to be used in the classification procedure. In the proposed control procedure, this is limited to model concepts that share a user specified base concept $item \in C_{\mathcal{D}}$ (regarding to the type hierarchy). Thus, there is a model concept sequence $c_0 \dots c_n$ with $c_0 = c, c_n = item$ and $c_{i+1} \in is-a(c_i)$ for all model concepts that support classification. The *is-a* relation is transitive. Thus, $a is-a b \wedge b is-a c \rightarrow a is-a c$ holds true. Therefore, we can write $c is-a item$ for concepts $c \in$

$C_{\mathcal{D}}$ that support classification. Not all attributes of the model concepts are intended to be used as features for the classification procedure. Thus, a method is required that tags model concept attributes as classification features. In the proposed classification based agenda selection method, a special attribute – *features* – is used. The attribute value represents the set of classification feature names for the particular concept. All other attributes are ignored during classification. I propose to use so called property relations in order to augment the feature vector of model concepts for the classification procedure. In the following, let $P_{\mathcal{D}} \subset R_{\mathcal{D}}$ be the set of property relations which are declared in ontology \mathcal{D} .

Definition 4.4.1. The model concept $p \in C_{\mathcal{D}}$ is called property concept of $c \in C_{\mathcal{D}}$ if there is a property relation $r_p \in P_{\mathcal{D}}$ with $p \in r_p(c)$.

Classification features of property concepts are included in the feature vector of the model concept that supports classification.

The feature vector computation procedure is shown in algorithm 3.

Algorithm 3 Model Concept Classification Features

```

1 function  $\mathcal{F}_C(c)$ 
2    $out \leftarrow []$  ▷ The feature vector
3    $\mathcal{F}_C(c, out)$ 
4   return  $out$ 
5 function  $\mathcal{F}_C(c, out)$ 
6   for  $attribute \in features(c)$  do ▷  $attribute \in A_{\mathcal{D}}$ 
7      $out \leftarrow out + [attribute(c)]$ 
8   for  $r_p \in P_{\mathcal{D}}$  do ▷  $r_p$  is a property relation
9     for  $p \in r_p(c)$  do ▷  $p$  is a property concept
10     $\mathcal{F}_C(p, out)$ 

```

Model concepts provide the feature vector domain. The prototypical values of feature attributes define the domain of that particular feature instead of an explicit value. For example, the value of a feature attribute ‘*color*’ that represents the dominant colors of an entity may be given by a selection set of color name constants (e.g., *red*, *green*, ...). Model instances provide dynamic instantiations of the feature vector where the feature attribute values must be specialized to an unique value to be usable for classification. For instance, the ‘*color*’ attribute could be specialized to an explicit color constant like *red*. Feature attributes of model instances that are not unique remain unspecified for the classification procedure (a constant value *Unspecified* is used in this case). In the following, let $\mathcal{F}_I(i)$ be the classification feature vector of instance $I_{\mathcal{D}}$ and let $P'_{\mathcal{D}} \subset R'_{\mathcal{D}}$ be the set of property relations which are declared in ontology \mathcal{D} .

Definition 4.4.2. The model instance $p \in I_{\mathcal{D}}$ is called property instance of $i \in I_{\mathcal{D}}$ if there is a property relation $r'_p \in P'_{\mathcal{D}}$ with $p \in r'_p(i)$.

The feature vector computation procedure is illustrated in algorithm 4.

Such a feature vector is computed for each training instance in order to obtain the parameters of

Algorithm 4 Model Instance Classification Features

```

1 function  $\mathcal{F}_I(i)$ 
2    $out \leftarrow []$  ▷ The feature vector
3    $\mathcal{F}_I(i, out)$ 
4   return  $out$ 
5 function  $\mathcal{F}_I(i, out)$ 
6    $c \leftarrow c(i)$  ▷  $c \in C_{\mathcal{D}}$ 
7   for  $attribute \in features(c)$  do ▷  $attribute \in A_{\mathcal{D}}$ 
8     if  $attribute(i)$  is unique then
9        $out \leftarrow out + [attribute(i)]$ 
10    else
11       $out \leftarrow out + [Unspecified]$ 
12    for  $r'_p \in P'_{\mathcal{D}}$  do ▷  $r_p$  is a property relation
13      for  $p \in r'_p(i)$  do ▷  $p$  is a property instance
14         $\mathcal{F}_I(p, out)$ 

```

the probabilistic model for each model concept that supports classification. This model can be used in the classification selection method in order to estimate the similarity of model instances and task entities. Let $i \in I_{\mathcal{D}}$ be a model instance with i *is-a* item and let $c \in C_{\mathcal{D}}$ be a concept that corresponds to a task entity. Thus, there is a task entity $t \in I_{\mathcal{T}_{\mathcal{D}}}$ with $c = c(t)$. This classification procedure can be expressed by the following mapping $classify : I_{\mathcal{D}} \times C_{\mathcal{D}} \rightarrow \mathbb{R}$ (see equation 3.10):

$$\begin{aligned}
 classify(i, c) &= \mathcal{P}(C=c \mid \mathcal{F}=\mathcal{F}_I(i)) \\
 &= \mathcal{P}(\mathcal{F}=\mathcal{F}_I(i))^{-1} \mathcal{P}(\mathcal{F}=\mathcal{F}_I(i) \mid C=c) \mathcal{P}(C=c)
 \end{aligned}
 \tag{4.16}$$

The prior $\mathcal{P}(\mathcal{F}=\mathcal{F}_I(i))^{-1}$ prefers rare feature values. This makes sense because rare values might be more expressive than frequent values. The other prior ($\mathcal{P}(C=c)$) prefers concepts with a high number of corresponding training instances. Finally, $\mathcal{P}(\mathcal{F}=\mathcal{F}_I(i) \mid C=c)$ prefers feature responses that are likely for the target model concept.

In the proposed classification selection method, the similarity between a model instance $i \in I_{\mathcal{D}}$ and a task entity $t \in I_{\mathcal{T}_{\mathcal{D}}}$ is defined as classification problem. The similarity is maximal if the class of the model instance is already a parent class of the task entity class and it is minimal if both classes are incompatible. In the other case, the classification model is used to estimate the probability that the model instance can be specialized to the model concept that corresponds to the task entity. This can be interpreted as mapping $similarity : I_{\mathcal{D}} \times I_{\mathcal{T}_{\mathcal{D}}} \rightarrow \mathbb{R}$ of model instance and task entity pair to a similarity value in range $[0, 1]$:

$$similarity(i, t) = \begin{cases} 1 & \text{if } c(t) \text{ is-a } c(i) \\ classify(i, c(t)) & \text{if } c(i) \text{ is-a } c(t) \\ 0 & \text{else} \end{cases}
 \tag{4.17}$$

There may be multiple task entities. The selection method estimate is defined as the maximum similarity to a task entity. This can be expressed by a mapping $classify-selection : I_{\mathcal{D}} \rightarrow \mathbb{R}$ that maps a model instance $i \in I_{\mathcal{D}}$ to the maximum task similarity value:

$$classify-selection(i) = \begin{cases} 1 & \text{if } i \in I_{\mathcal{T}_{\mathcal{D}}} \\ \max_{t \in I_{\mathcal{T}_{\mathcal{D}}}} similarity(i, t) & \text{else} \end{cases} \quad (4.18)$$

Summarizing, the proposed classification selection method prefers model instances with high probability to share the object class with a task entity. This method can be used to focus the perception methods on likely task entities by ordering the agenda items based on the task similarity estimate.

4.4.3 Invocation of Perception Methods

An agenda item can be interpreted as a description of an atomic operation that can be performed on the model instance that is referred by the agenda item. It includes information about the operation type, a specific attribute or relation and the allowed domain for the operation. However, it does not include information about the methods that could be used to compute a meaningful value that specializes the item domain. Agenda processing methods are domain independent procedures which are used to specialize dynamic model instances based on the corresponding model concept. The knowledge about agenda item processing methods is included in the control strategy.

In PLAKON, each strategy contains a prioritized list of method descriptions. For each selected agenda item, this sequence is processed in priority order until one of the methods returned a value that is consistent with the domain of the agenda item. In the proposed control architecture, strategies are extended in order to contain a prioritized list of so called flows. Each flow contains a prioritized list of method descriptions. Each method description includes the method name and configuration parameters for the method. This allows to execute methods in different contexts using a different set of configuration parameters. Additionally there is an agenda item pattern defined for each flow. A flow is only activated if the selected agenda item matches this pattern.

To formalize this, let \mathfrak{F} be the set of flows defined in strategy \mathfrak{S} and let \mathfrak{A} be the agenda of \mathfrak{S} . For each flow $flow \in \mathfrak{F}$, there is a corresponding agenda item pattern $p_{flow} \in P$, where $P = \{\mathfrak{A} \rightarrow \text{BOOL}\}$ is the set of expressible patterns which map agenda items to *true* if the pattern matches and to *false* otherwise. $flow$ is only activated for agenda items $item \in \mathfrak{A}$ which match the pattern: $p_{flow}(item) = \text{true}$.

Selected agenda items may turn out to be invalid (empty domain) or unique (domain is a unique value). In the first case, the item is removed from the agenda because the corresponding operation

must not be executed anymore. In the second case, the unique value is applied without the usage of agenda processing methods. Furthermore, there may be no activation pattern that matches the selected item. In this case, the item cannot be processed and must be removed from the agenda. The complete sequence of referenced methods is executed when a flow with a matching activation pattern was found. This step of the control loop is completed if the agenda item domain turned into a unique value after the methods were executed. Otherwise, the agenda item is postponed using the inhibition agenda selection method (see section 4.4.2). This procedure is illustrated in algorithm 5.

Algorithm 5 The Agenda Processing Algorithm

```

1 function PROCESS-AGENDA( $\mathfrak{S}$ )                                ▷  $\mathfrak{S}$  is a control strategy
2    $\mathfrak{A} \leftarrow$  Agenda of strategy  $\mathfrak{S}$ 
3   for  $item \in \mathfrak{A}$  do
4     if  $item$  is invalid then                                ▷ Drop invalid items
5        $\mathfrak{A} \leftarrow \mathfrak{A} \setminus [item]$ 
6     else if  $item$  is unique then                            ▷ Automatically process unique items
7       Apply unique value
8        $\mathfrak{A} \leftarrow \mathfrak{A} \setminus [item]$ 
9     else
10      PROCESS-ITEM( $\mathfrak{S}, item$ )
11      if  $item$  is unique then                                ▷ Exit if the item domain is unique after processing
12        break
13 function PROCESS-ITEM( $\mathfrak{S}, item$ )
14    $has-flow \leftarrow false$                                 ▷ Is there a matching flow?
15    $\mathfrak{F} \leftarrow$  Flow sequence of strategy  $\mathfrak{S}$ 
16   for  $flow \in \mathfrak{F}$  do
17      $p_{flow} \leftarrow$  Activation Pattern of  $flow$ 
18     if  $p_{flow}(item)$  then                                ▷ If the activation pattern matches the agenda item
19        $has-flow \leftarrow true$ 
20       PROCESS-FLOW( $item, flow$ )
21       if  $item$  is unique then                                ▷ Exit if the item domain is unique after processing
22         break
23   if  $has-flow = false$  then                                ▷ If there is no matching flow sequence
24      $\mathfrak{A} \leftarrow \mathfrak{A} \setminus \{item\}$ 
25 function PROCESS-FLOW( $item, flow$ )
26    $X \leftarrow$  Sequence of agenda processing methods of  $flow$ 
27   for  $m \in X$  do                                          ▷ For each processing method of  $flow$ 
28      $CFG \leftarrow$  Configuration parameters of method  $m$  in  $flow$ 
29      $m(item, CFG)$                                           ▷ Execute the agenda processing method  $m$ 
    
```

The domain of agenda items depends on the agenda item type. Supported types and domains of agenda items do not notably differ from the ones supported in PLAKON. The set of supported agenda item types includes the operations:

- *Specialization*

Represents a specialization along the type hierarchy. Thus, a specialization of the model

concept that corresponds to the selected model instance.

- *Parametrization*
Represents a specialization of an attribute value of the selected model instance. This includes the restriction of the attribute value domain and the specification of unique attribute values.
- *Decomposing*
The decompose operation instantiates model concepts based on a particular relation of the selected model instance in order to create the relation between the selected model instance and the instantiated model concepts.
- *Integration*
This operation is similar to the decompose operation. The objective is to relate different model instances to each other. But this operation does not instantiate model concepts: Existing model instances are integrated along a particular relation.
- *Fusing*
The fuse operation integrates a model instance into another existing instance. The resulting model instance represents the most special aspects of both model instances.

In the scope of this thesis, following agenda processing methods are used:

- *Dynamic and Static Default*
Static defaults are supported only for the *Decomposing* operation (where the default cardinality is used) and the *Parametrization* operation (where the default attribute value is used). In both cases, the default value is attached to the particular entities in the ontology. Dynamic defaults include the computation of the minimum and maximum. These methods are only supported for the *Decomposing* operation and for the *Parametrization* operation of parameters with a domain where a maximum and a minimum are defined (for example numeric ranges). Furthermore, there is a method that allows the knowledge engineer to specify a constant or dynamic value. Dynamic values are realized using computable attribute values. The result of the computation is a specialization the computable value. This is used to access available model instances in order to dynamically read attribute values. For example, the configuration parameters of methods can be attached to model instances and then applied using computable attribute values before the method is executed.
- *Control Rules*
This method is used in order to fire active rules of a specific set of rules. The rules must be contained in the same rule group. The name of the rule group is a configuration parameter of this method. Furthermore, it is possible to specify that only a single rule should fire or

that it is allowed to fire multiple rules. Rules have access to the currently selected agenda item and all other instantiated model concepts. The consequences of these rules are not limited to the selected agenda item and the corresponding value domain. Thus, it is allowed that rules affect other agenda items than the selected one.

- *User Question*

User Question methods wait on user input for a valid domain specialization. The only method that is supported shows a Graphical User Interface (GUI) dialog on the screen and waits until the user hits one of the dialog buttons.

- *Task Definition*

Contrary to PLAKON, the proposed control procedure supports the dynamic definition and manipulation of tasks. For example, when a task was completed, it can be entirely replaced by a new task. This is basically realized by manipulating relations between the instance of the task concept and other model instances (see section 4.3). Note that there is only one dedicated instance of the task concept that is used to tag model instances as task entities. Additionally, unrecognized and underspecified task entities are removed when the task is deactivated.

- *Perception Methods*

The perception agenda processing methods correspond to methods of the perception component (see section 2.2.3). There is an agenda processing method for each of the perception methods (the methods are generically exposed to the control component). This set of methods is comparable with simulation methods as they were used in PLAKON. Many of the perception methods used in the perception component provide more information than required according to the selected agenda item. For example, the *CLuster3DGeometryAnnotator* computes the size and the shape of objects. The perception component uses a custom data store – the Common Analysis System (CAS) – that is provided by the UIMA implementation. Mapping of perception method results to model instances is realized using a special CAS agenda processing method.

- *CAS Mapping*

The CAS mapping method reads feature values from the perception data store and uses them to specialize the domain of the selected agenda item. The method also supports to decompose relations based on aggregate feature values which contain a set of other feature values. It is convention that the type name of the aggregate feature must be equal to the name of the instantiated model concept. For example, a feature with the type *Pose* in the perception component corresponds to a model concept *Pose* in the control component. Nesting of aggregate features is allowed too. Furthermore, there is a method that allows to

clear the perception data store and to remove specific features.

- *Classification*

The classification method is only used for the *Specialization* operation where the agenda item domain D is a selection set of model concept specializations: $D \subset C_{\mathcal{D}}$. The result of a classification method selects the model concept with maximum probability to be a specialization of the selected model instance $i \in I_{\mathcal{D}}$. This can be achieved using a Bayes' classifier (see section 3.2.2):

$$\begin{aligned}\lambda &= \arg \max_{c \in D} \mathcal{P}(C=c \mid \mathcal{F}=\mathcal{F}_I(i)) \\ \kappa &= \max_{c \in D} \mathcal{P}(C=c \mid \mathcal{F}=\mathcal{F}_I(i))\end{aligned}\tag{4.19}$$

Where $\mathcal{F}_I(i)$ is the feature vector of object annotations, $\lambda \in C_{\mathcal{D}}$ is the selected model concept and $\kappa \in \mathbb{R}$ is the classification confidence. The agenda processing method allows to specify a minimum classification confidence (the classification threshold). Classifications with a lower confidence are ignored.

- *Fusing*

The fuse method is dedicated to the *Fusing* operation: Two model instances are joined into a single model instance that represents the most special aspects of both fused model instances. In the scope of this thesis, the fuse method is used to integrate recognized objects into the task hierarchy. The task is completed if all task entities are fused with recognized objects or previously known objects (such as the kitchen table). The fuse method is supposed to join two model instances which represent the same object. It is required that the types of both model instances are equal. Type specialization operations must be performed before the fuse operation. Furthermore, it is required that the types are final (i.e., the type cannot be specialized anymore). Note that this makes it impossible to formulate visual search tasks such as “Find the small yellow thing on the kitchen table” where the type of the object is no known in advance. Additionally, model instances can only be fused when their attributes and relations are compatible with each other. For attributes, this means that the edit distance between attributes must be below an user defined threshold. For relations, this means that it must be possible to add the related objects of one model instance to the set of related objects of the other instance.

4.5 Summary and Discussion

In this chapter, I presented an adaptation of the ontology-based control mechanism of PLAKON in the context of perception in the kitchen domain and intended as a framework for computational attention methods. Perception methods are selected and configured based on control knowledge and estimates of computational attention methods. The control knowledge for particular stages in the perception procedure is collected in strategies. Each strategy consists of focusing, ordering and processing knowledge. The activation of strategies is done using rules. The activation rules can be seen as meta-control: Knowledge dedicated to the selection of the active control knowledge.

In the scope of this thesis, the ordering knowledge is particularly relevant because it can be represented by methods of computational attention. Ordering knowledge can express priorities for recognized objects (object-based attention; focusing of particular model instances), attributes of the objects (feature-based attention; focusing of particular attributes of the attended model instance) and regions in the kitchen (region-based attention; focusing of objects with *located-at* relation to particular furniture objects in the kitchen). In section 4.4.2.1, I proposed an attention method that uses the relation hierarchy between model instances and task entities in order to estimate the relational distance between the model instances and the task entities. The task relevance is computed based on the minimum distance to a task entity. This approach is inspired by the attention method that was discussed in section 3.1.4. Probabilistic approaches for computational attention (see section 3.2) are usually defined as classification problem where a set of feature measurements is used to predict the corresponding class label. There are many slightly different approaches to model the attention mechanism in a probabilistic manner (see section 3.2.2, 3.2.3). Most of them are based on Bayes' rule. In the scope of this thesis, a naive Bayes' classifier is used as it was discussed in section 3.2. The classifier is used to prioritize recognized objects with high similarity to task entity types according to the classifier.

Processing knowledge is used to represent the dynamic selection and configuration of perception methods in particular stages of the perception procedure and for particular selected features and recognized objects. This flexible handling of perception methods allows to execute the methods in different contexts and it allows flexible selection of methods based on object types, features or regions in the kitchen. For example, it is possible to define different perception method sequences for the clustering of objects in fridges and for objects on tables.

Implementation Details

The control architecture that I propose in this thesis is implemented in the object oriented programming language Java. Java suits well for the *RoboSherlock* perception component (see section 2.2.3) because it is integrated in a plugin framework that provides a interface for Java software. This allows seamless integration of the perception component in the control component. Furthermore, there are Java implementations of many machine learning algorithms publicly available. In the scope of this thesis, Weka ¹ (Waikato Environment for Knowledge Analysis) is used for the classification of recognized objects.

The implemented control framework is intended as a re-usable and generic framework for various perception tasks. No code has to be written to adapt the framework for other perception scenarios then one of the reference scenarios of this thesis (see section 2.3). It is sufficient that a knowledge engineer modifies the ontology and the control knowledge in order to model different tasks. The framework is modular on different layers: It is embedded into an Open Service Gateway initiative (OSGi) based Java plugin framework, the control framework consists of a set of control components and control knowledge plugins and all perception methods are dynamically loadable plugins in the UIMA based plugin architecture of the perception module. This layered architecture is illustrated in figure 5.1 and different aspects of the modularity are discussed in more detail in section 5.1.

The Java plugin framework that is used supports the registration of services. Java interfaces are used to specify the signature of services. Multiple implementations of the same service interface can be registered. Services are intended to be used in plugins of the framework.

Interfaces and implementations are decoupled for the core components of the control architecture. This allows to hide implementation details from other components (the Java packages are not exported to the Java class loader path of other components). Furthermore, this allows multiple implementations of the interfaces.

The interface pattern is compatible to the service architecture of the plugin framework. Different implementations of the interfaces can be registered and plugins can query the registered services

¹<http://www.cs.waikato.ac.nz/ml/weka/>

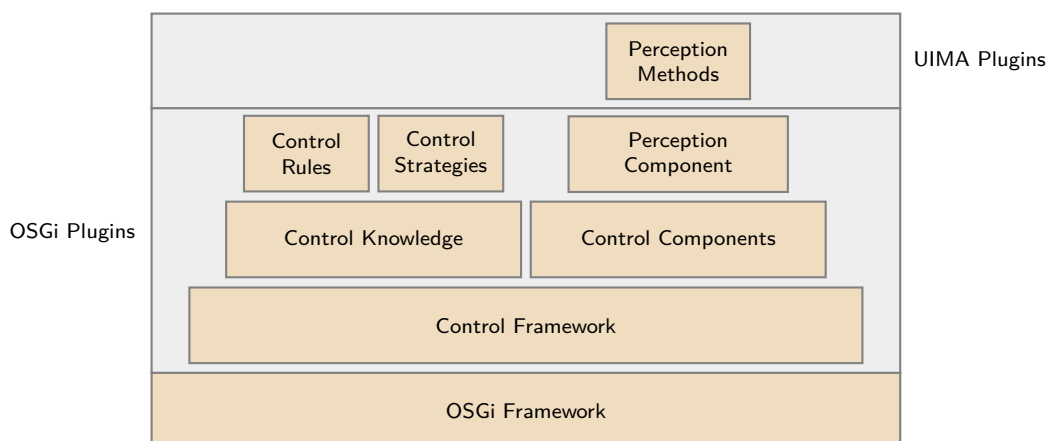


Figure 5.1 Layered architecture of the perception control framework.

for a particular interface.

The GUI is an essential aspect of this control framework because it allows the acquisition of control knowledge by a knowledge engineer as well as the application of the acquired knowledge (i.e., the execution of perception methods based on the acquired knowledge). It is integrated in a docking framework that allows users to set up custom layouts for the GUI.

Expert knowledge is required in order to formulate control rules. The complexity of the rule formulation depends on the syntax of control rules: It is harder to maintain control rules with a sophisticated syntax than it would be to maintain control rules in a syntax that is similar to natural language. Furthermore, the separation of rules into different rule groups improves the maintainability of rules in the control architecture. In section 5.2, I discuss the syntax of control rules and describe the rule editor of the control GUI.

Finally, I will discuss the acquisition and application of control knowledge in section 5.3.

5.1 Modular Architecture

The control framework is modular on multiple layers.

On the top layer, the framework consists of a collection of plugins in a widely used plugin framework for the Java programming language (see section 5.1.1).

The Java plugin framework is also used to make control components extendible by plugins. For example, different implementations of control rules can extend the selection of available rule formalism for the acquisition and application of control knowledge (see section 5.2).

Furthermore, another plugin architecture is used by the perception component *RoboSherlock*. Each of the perception methods of *RoboSherlock* is a shared library that is loaded into this

framework. The plugin architecture of *RoboSherlock* is discussed in section 5.1.2.

5.1.1 Open Service Gateway initiative (OSGi)

The proposed control framework is embedded into OSGi ² which is a widely used plugin framework specification for the Java programming language. There are multiple implementations of the OSGi specification available (*Felix* ³, *Equinox* ⁴, *Knoplerfish* ⁵). I used the *Equinox* OSGi implementation for the validation of the control framework because it allows a seamless integration with the eclipse development platform (eclipse is based on the equinox OSGi implementation). But this framework does not depend on *Equinox*, other OSGi implementations can be used as well.

Plugins are called *bundles* in OSGi. Bundles must implement the *BundleActivator* interface (start and stop methods) and they can import packages from the system or from other bundles. Bundles can also export packages so that other bundles can import them and they can offer multiple services to other bundles. Services are always separated into interface and implementation.

The OSGi system loads the default bundle on startup (all other bundles will be started from within the default bundle) and it defines system packages and system services that will be exported to all other bundles.

System packages can be imported by bundles. Note that all *java.** packages are system packages by default. Additional packages must be exported explicitly by the OSGi framework (for example any *javax.** packages) or by a custom bundle that includes the implementation of the exported packages.

System services are the services which are exposed by the default bundle. Therefore system services are available for all bundles (because all bundles are loaded from within the default bundle). The most commonly used system services include services for logging of messages (the *Pax Logging* service) and configuration of bundles (the *ConfigurationAdmin* and *Pax ConfMan* services).

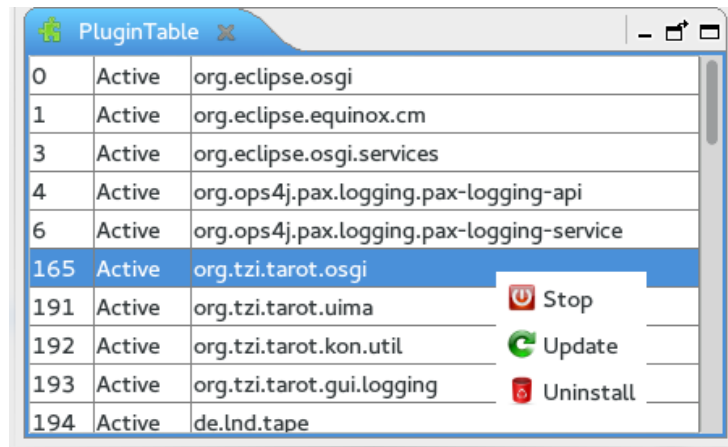
OSGi allows flexible configuration of the set of installed and automatically started bundles. Different initialization levels are used in order to sequence the initialization of bundles. Furthermore, bundles can be dynamically installed, uninstalled, started and stopped. This allows that parts of the application can be unloaded, modified by a programmer and loaded again without termination of the framework. The user interface that was implemented for the management of OSGi bundles is shown in figure 5.2.

²<http://www.osgi.org/>

³<http://felix.apache.org/>

⁴<http://www.eclipse.org/equinox/>

⁵<http://www.knoplerfish.org/>



ID	State	Bundle Name	Actions
0	Active	org.eclipse.osgi	
1	Active	org.eclipse.equinox.cm	
3	Active	org.eclipse.osgi.services	
4	Active	org.ops4j.pax.logging.pax-logging-api	
6	Active	org.ops4j.pax.logging.pax-logging-service	
165	Active	org.tzi.tarot.osgi	Stop, Update, Uninstall
191	Active	org.tzi.tarot.uima	
192	Active	org.tzi.tarot.kon.util	
193	Active	org.tzi.tarot.gui.logging	
194	Active	de.lnd.tape	

Figure 5.2 The Bundle Management GUI. The list of installed bundles. A pop-up menu allows to dynamically start, stop and uninstall the selected bundle.

The control framework consists of 36 custom bundles including a bundle that provides a GUI service implementation (*org.tzi.tarot.gui*), a few bundles that implement the control mechanism (*org.tzi.tarot.control.**), a few ontology related bundles (*org.tzi.tarot.kon.**), bundles dedicated to the export of external libraries (*org.tzi.tarot.weka*, *org.tzi.tarot.drools*, ...) and bundles that provide different GUI components (e.g., the plugin management GUI component shown in figure 5.2).

5.1.2 Unstructured Information Management Architecture (UIMA)

The perception component – *RoboSherlock* – consists of a set of methods dedicated to the perception procedure. The methods are integrated into an architecture for the analysis of unstructured data (UIMA). Each perception method is a shared library that can be loaded dynamically at runtime of the framework. In the context of UIMA, perception methods are called *analysis engines*. The so called *descriptors* of analysis engines are used to expose the set of perception methods and their parameters to the control component. Perception method *descriptors* basically contain the name of the method and a set of configuration parameters (name, type, default value). This information is exposed to knowledge engineers during acquisition of control knowledge.

The *flow controller* component of UIMA is particular relevant for the control procedure. It is used to dynamically select analysis engines for execution. UIMA only defines the interface for flow controller. The flow controller interface is implemented by the control component in order to do the analysis engine selection based on control knowledge and methods of computational attention. Additionally, dynamic reconfiguration of analysis engine is supported by the custom flow controller implementation.

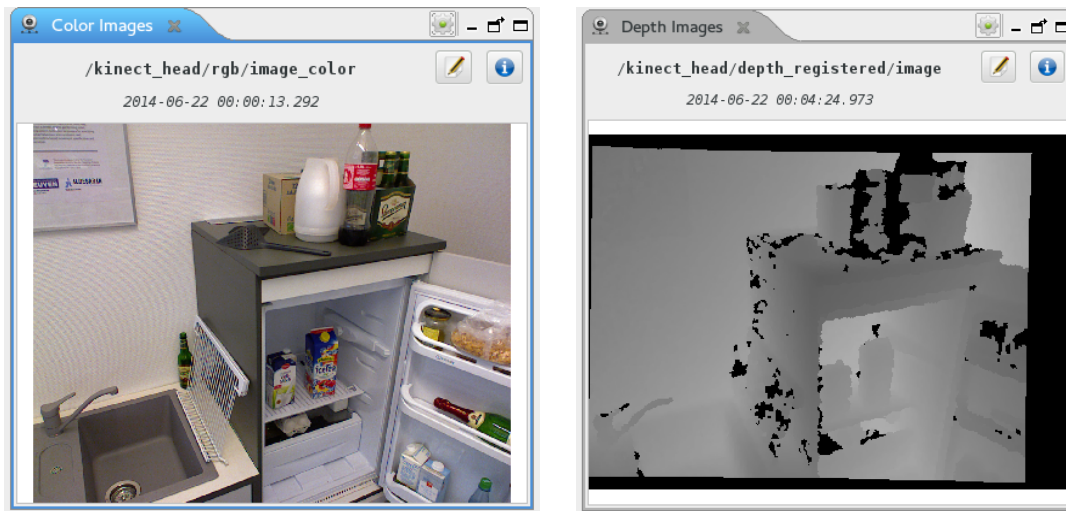


Figure 5.3 GUI components that display RGB and depth sensor messages of the topic `/kinect_head/rgb/image_color` and `/kinect_head/depth_registered/image`. The picture shows the field of view of the RP2 robot in the reference kitchen while the PR2 looks into the fridge.

In UIMA, unstructured data is collected in – so called – *sofas* (subject of analysis). Each *sofa* is intended to be a different view on the same data. *RoboSherlock* uses multiple *sofas* including a *sofa* that represent the RGB color, depth and point cloud as seen from the robots point of view. Additionally, *RoboSherlock* uses some *sofas* for custom UIMA types (UIMA has an extendible type system).

The set of all *sofas* is managed by a container component that is called CAS. The CAS provides analysis engines access to each of the registered *sofa*.

The results of analysis engines is written to *sofas* in the CAS. In the control procedure, the execution of analysis engines is coupled with an adapter component that reads the results from the CAS in order to apply them to model instances. In order to be able to monitor the perception process, the GUI of this framework contains a monitoring component for the CAS.

The CAS contains multiple *sofas* and each *sofa* contains typed feature data (the result of analysis engines). The most relevant *sofas* are dedicated to the sensory input of the system (RGB, depth, point cloud) and to the representation of the scene from the current point of view of the robot. Recognized objects are attached to the *scene sofa*. In order to monitor the perception procedure it is desirable to be able to view the content of the CAS dynamically during execution of the perception control procedure. For that purpose, the framework provides a set of visualization components for features in the CAS. Figure 5.4 shows the visualization components.

Internally, some of the analysis engines require an active Robot Operating System (ROS) message server. The ROS message server is used in order to expose the state that was perceived by sensors of the PR2 robot to the UIMA framework. ROS follows the *publish-subscribe* design pattern:

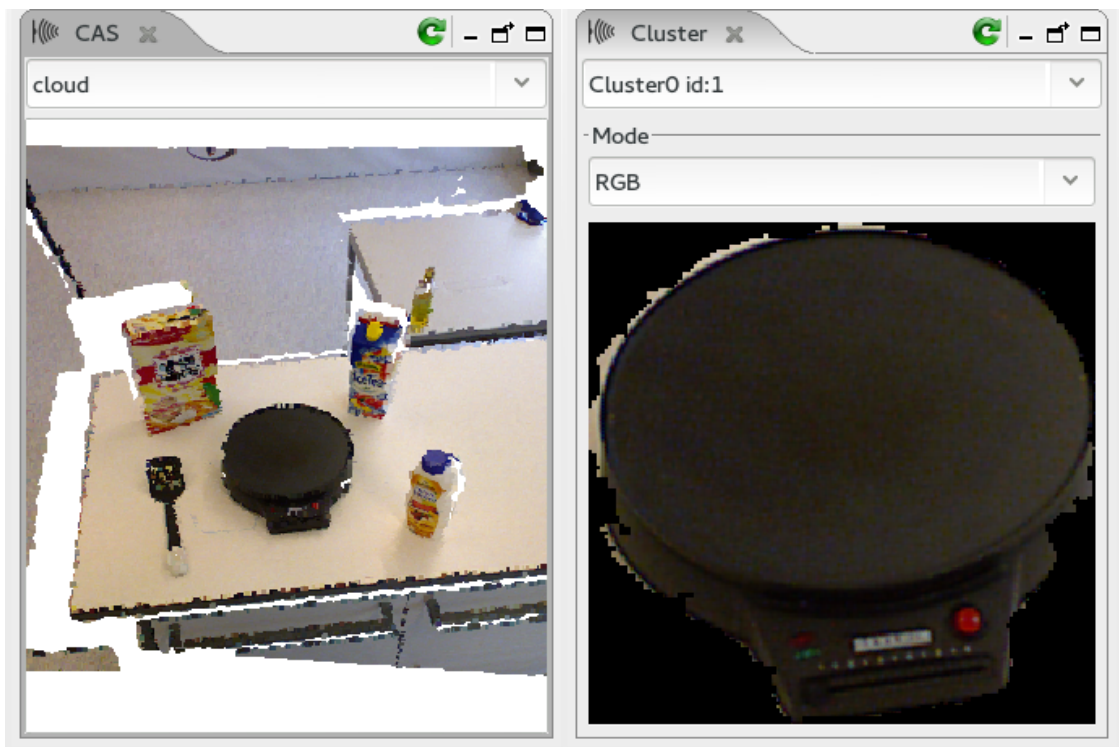


Figure 5.4 CAS visualization user interface. On the left: A GUI component that visualizes point clouds in the CAS; On the right: A GUI component that visualizes recognized objects.

Subscriptions of particular topics are used to notify listeners about new messages. Different topics are used for different types of sensory input. For example, there are different ROS topics for the color and the depth as seen from the robots field of view as well as there is a topic for point clouds where each point corresponds to a point in space as seen from the robots field of view. A simple GUI for the display of ROS messages was integrated into the control framework (see figure 5.3).

5.1.3 Control Framework Services

OSGi services support the re-usability of software components: Common functionality can be encapsulated in services and exposed to other OSGi bundles. Core services of the control framework are used by control related bundles. The most important core services are discussed in the following listing:

- *ConfigurationManager*

The core of the configuration management is the OSGi service *ConfigurationManager*. It supports loading of configuration resources from bundles. Bundles must use this service in order to expose their configuration parameters to the OSGi framework. The OSGi framework provides another interface for the notification of listener about configuration parameter changes (*ManagedService*). The control framework provides a specialization of the *BundleActivator* interface that is intended to be used for bundles with configuration parameters (*ConfigurableBundle*).

Registered configuration parameters are automatically serialized and exposed to a generic configuration editor (see figure 5.5). The modular architecture also allows other components to access and to manipulate configuration parameters of bundles.

- *CommandService*

The *CommandService* provides a central registration point for *command* methods of bundles. Each command method represents functionality of the bundle that is intended to be exported to other bundles. Command methods are grouped into command environments in order to avoid name clashes. A command handle for a particular environment and command can be queried from the *CommandService* implementation.

The set of default command environment definitions includes an environment dedicated to the management of bundles and one that is dedicated to the configuration of bundles. Furthermore, there are command environments defined for the manipulation of ontologies as well as for the management of control knowledge.

Furthermore, there is a GUI component that allows to execute registered commands (see

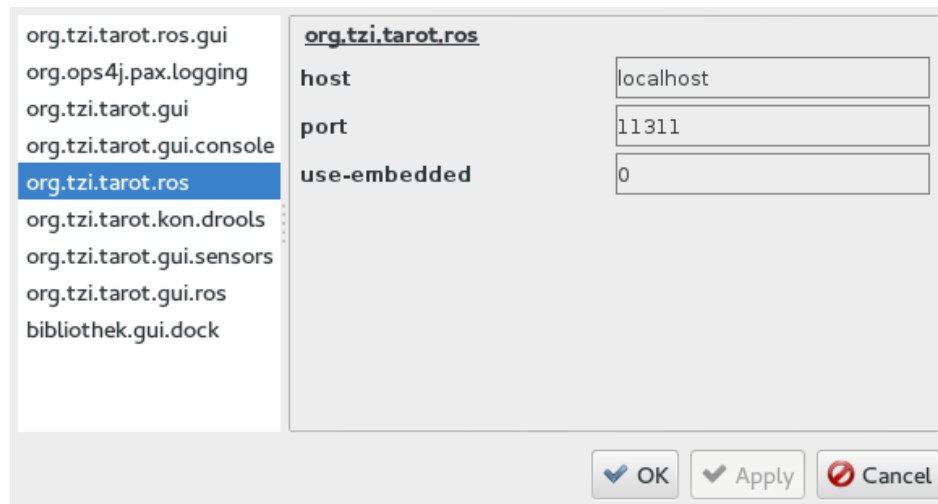


Figure 5.5 Generic configuration parameter editor that uses the *ConfigurationManager* service. The selection list on the left contains a list of bundles with configuration parameters.

figure 5.6).

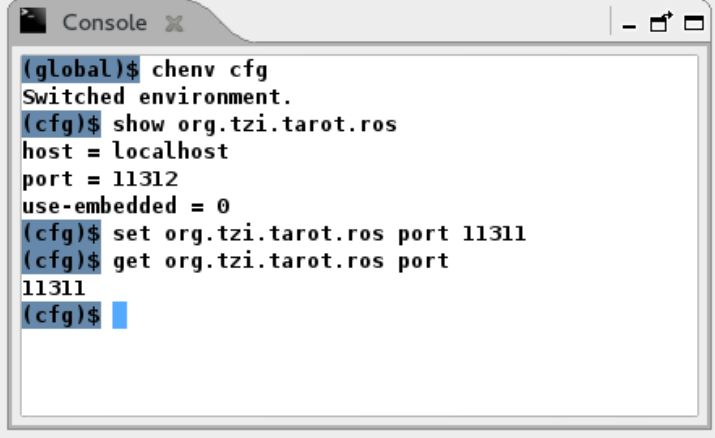
- *MessageService*

The *MessageService* OSGi service implements the *publish-subscribe* design pattern. The service can be used for communication between bundles. ROS uses the same design pattern for sensor messages. The *MessageService* can be used as adapter to the ROS messages: ROS messages are passed through to subscribers of the *MessageService* service. Using the *MessageService*, subscriptions to ROS topics can be established even if no ROS message server is running at that particular moment (the service waits for a ROS message server connection in this case).

5.2 Control Rules

In this ontology-based control framework, control rules are used to select the currently active strategy of control components. Furthermore, there are control rule groups for the initialization of the control procedure and for rules that are supposed to be activated at the beginning of each frame.

Rules are generically defined as a set of rule components and parameters. Each component has a *scope* and a value. Usually, rules can have two different component scopes: Conditions and consequences.



```
(global)$ chenv cfg
Switched environment.
(cfg)$ show org.tzi.tarot.ros
host = localhost
port = 11312
use-embedded = 0
(cfg)$ set org.tzi.tarot.ros port 11311
(cfg)$ get org.tzi.tarot.ros port
11311
(cfg)$
```

Figure 5.6 The console is a graphical user interface for the *CommandService* OSGi service. The picture shows an interaction with the *ConfigurationManager* service.

The core component of the control procedure only declares the interfaces for rules. Rule implementations can be dynamically added and removed from the control component. This is achieved by registration of OSGi services: Each rule implementation must register a typed rule service at the OSGi framework. Furthermore, multiple concurrent rule implementations can be registered at the same time.

Rule implementations must implement a rule factory interface (*HykonRulesFactory*) that is used to instantiate the rules. As for the control knowledge, acquisition and application of control rules is strictly separated: Rule factories are used to instantiate implementations of the *HykonRulesKnowledgeBase* interface and of the *HykonRulesSession* interface. The *HykonRulesKnowledgeBase* interface is used for the acquisition of rules and the *HykonRulesSession* interface is used for the application of rules. *HykonRulesKnowledgeBase* is a container for compiled rules that allows to dynamically add and remove rules. Finally, *HykonRulesSession* provides the interface for the activation (firing) of rules (i.e., the checking of rule conditions and the execution of rule consequences).

Internally, the control component uses the proxy design pattern in order to hide if particular rule factories are registered or not (rule factories can dynamically be registered and unregistered because they are implemented as OSGi bundles). This allows serialization of rules even if the corresponding bundle that exposes the particular *HykonRulesFactory* implementation is not loaded.

Note that the generic rules interface is intended to support later integration of a constraint network into the control framework.

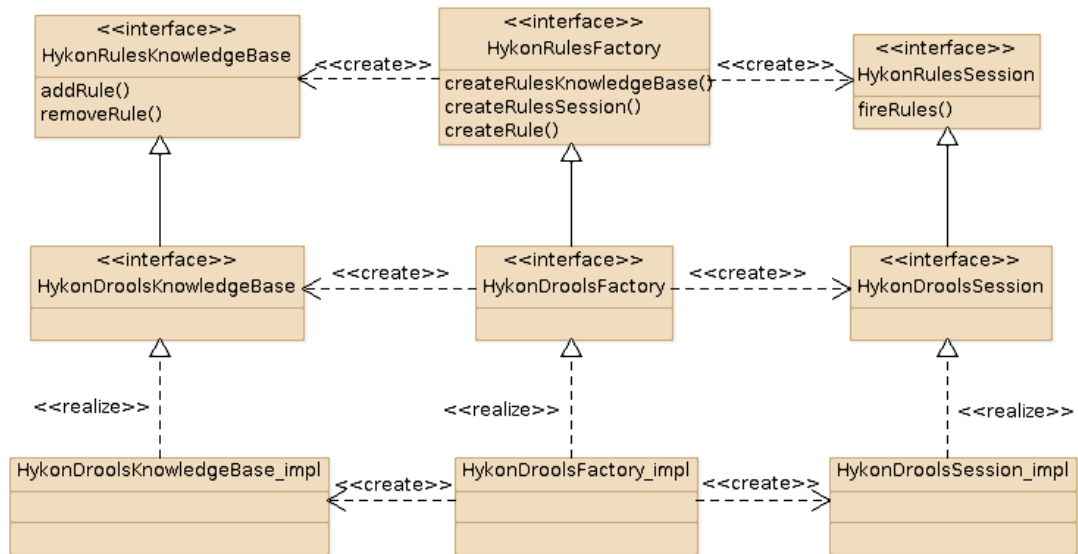


Figure 5.7 UML diagram that shows the *HykonDroolsFactory* implementation of the *HykonRulesFactory* interface.

5.2.1 Drools Control Rules

The control framework currently provides only one rule implementation that is based on *Drools*⁶. *Drools* provides an unified and integrated platform for rules, work flow and event processing. It is implemented in the Java programming language and it is integrated seamless in the control framework. The class hierarchy of the drools rules service implementation is shown in figure 5.7.

Drools collects a set of rules in different named rule packages. Rule packages can include static methods of Java classes as well as they can declare global variables of Java types. This is used in order to give rules access to functionality that is implemented in the Java programming language. Model concepts and model instances are asserted as facts into the working memory of the drools session. The asserted facts can be used in conditions of rules. For example, rules can check for an instantiation of a particular model concept with particular specializations (e.g., the dominant color is red).

A rule is defined by a set of configuration parameters, conditions and consequences. The most important configuration parameters are the rule group (grouping of rules in different contexts), the activation group (only a single rule of particular activation groups can be activated) and the saliency (the rule priority).

The control framework dynamically generates source code for drools rules and drools dynamically compiles the generated rules. Furthermore, the control framework supports to use a *Domain Specific Language* (DSL) in conditions and consequences of rules (see next section).

⁶ <http://drools.jboss.org/>

-Name	
Activate Annotation-Phase0	
-Properties	
agenda-group	strategy
activation-group	Annotation-Phase
-Components	
[condition]	There is an active <input type="text" value="perception"/> runtime assigned to <input type="text" value="runtime"/>
[condition]	There is an instance of concept <input type="text" value="Annotation"/> assigned to <input type="text" value="annotation"/>
[condition]	The instance <input type="text" value="annotation"/> has specializable attributes
[consequence]	Activate stage <input type="text" value="Annotation-Phase"/> of runtime <input type="text" value="runtime"/>
<input type="button" value="+"/> <input type="button" value="-"/> <input type="button" value="A"/> <input type="button" value="A"/>	

Figure 5.8 DSL-based rule editor. The editor displays a strategy selection rule (the agenda group is set to *strategy*) that is activated when there is an instance of the *Annotation* model concept with unspecified attributes. The consequence of the rule activates the strategy *Annotation-Phase*.

Domain Specific Language (DSL) Expert knowledge about the syntax and the control framework is required in order to write rules. This can be relaxed by a Domain Specific Language (DSL) that hides the sophisticated syntax of drools rules from the knowledge engineer. A term in a DSL definition contains a natural language pattern (with placeholders) and a mapping to drools rules syntax. In the control framework, rules can contain instantiations of DSL patterns (i.e., the placeholders are replaced with concrete values) in conditions and consequences of the rule. Furthermore, it is supported to dynamically register DSL definitions for drools rules.

The core of the drools rules service defines a DSL that is an interface to model concepts and model instances. It allows conditions to check for the existence of model instances with particular specializations and it allows consequences to execute *Specialization*, *Parametrization*, *Relation* and *Fusing* operations. The control core adds another DSL definition that is used for the selection of strategies. For that purpose, control components (called *runtimes*) are asserted as facts to the drools session. These facts can be used to access the active strategy instantiation (called *stage*) and they can be used to select the currently active *stage*. The formulation of control rules is supported by an rule editor that allows to formulate rules as a composition of DSL terms. The editor is shown in figure 5.8.

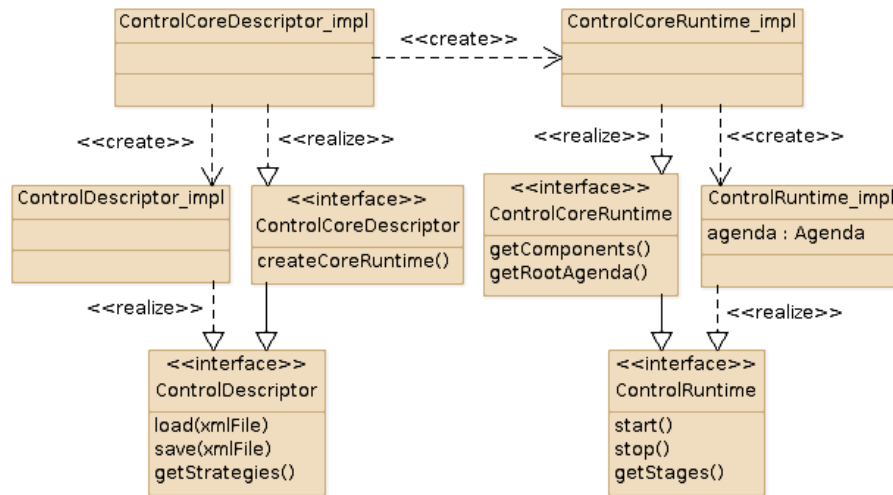


Figure 5.9 UML diagram that shows the *ControlDescriptor* and the *ControlRuntime* interfaces that are used for the acquisition and the application of control knowledge.

5.3 Acquisition of Control Knowledge

The control architecture consists of a core component that is implemented as a container for multiple other control components. Each of the control components is separated into *descriptor* and *runtime*. The control *descriptor* is used during acquisition of control knowledge and the control *runtime* is used during application of control knowledge. Each control descriptor has a custom set of control strategies, control rules and overwrites for agenda processing methods associated to it. Additionally, a serialization interface must be implemented by control descriptors. Control *runtimes* can be started and stopped. The core *runtime* is responsible for the management of component *runtimes*. All component *runtimes* are started and stopped together with the core *runtime*. Furthermore, the core *runtime* contains the root agenda. All component *runtimes* inherit the agenda items from the root agenda. The purpose of component *runtimes* is to process items of the agenda view that they inherited from the core component. The architecture is shown in figure 5.9.

The control framework provides a GUI for the acquisition of control knowledge that is intended to be used by knowledge engineers. There are different GUI components for the acquisition of focusing knowledge (agenda focus patterns; see figure 5.10), ordering knowledge (agenda selection criteria; see figure 5.11) and processing knowledge (control flows; see figure 5.12). For more details on these components of the control framework see section 4.4.1.1 (focusing knowledge), section 4.4.2 (ordering knowledge) and section 4.4.3 (processing knowledge).

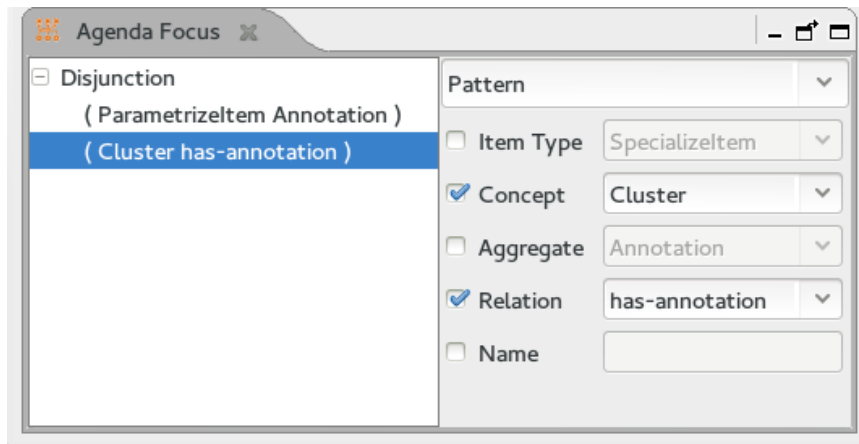


Figure 5.10 User interface for the acquisition of focusing knowledge. The picture shows an agenda focus that selects all agenda items for the *Parametrization* operation of the *Annotation* concept and all items for the *has-annotation* relation of the *Cluster* concept. All other agenda items are ignored.

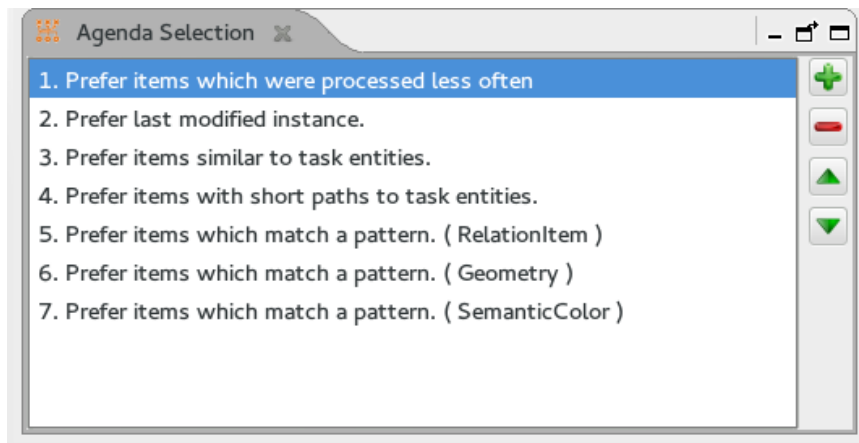


Figure 5.11 User interface for the acquisition of ordering knowledge. The ordering knowledge is expressed as prioritized list of agenda selection criteria. For example, the selection criterion 5 is a agenda item pattern selection criterion. It prefers all *Relation* operations over other operations.

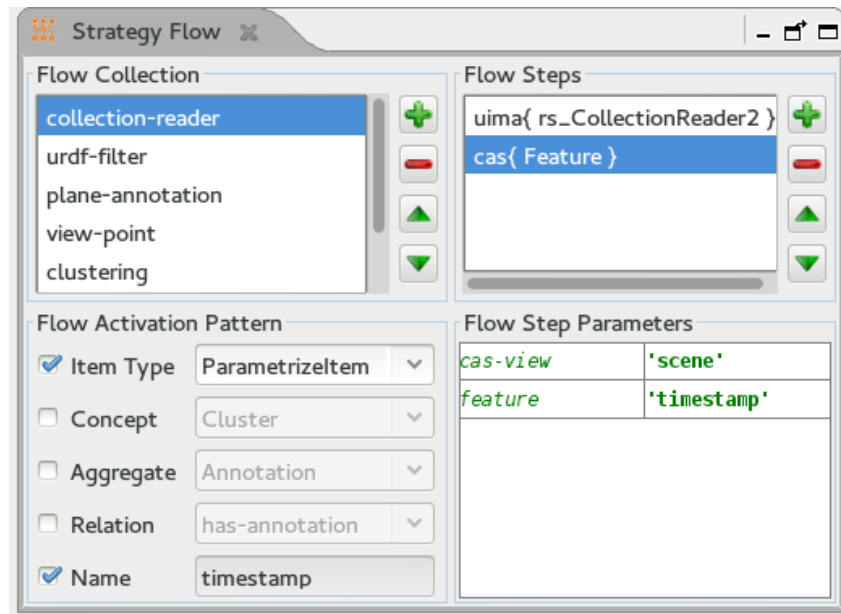


Figure 5.12 User interface for the acquisition of processing knowledge. Each strategy contains a collection of flows (top left) and each flow defines an activation pattern (bottom left) and a sequence of agenda processing methods (top right). Finally, each method can be configured by a set of parameters (bottom right). The picture shows a flow that is activated by *Parametrization* operations for the attribute *timestamp*.

5.4 Application of Control Knowledge

The application of control knowledge is driven by agenda processing methods. It is possible to dynamically register agenda processing methods using the OSGi service interface. The control core defines the service interface — *AgendaProcessing* — that must be implemented by custom agenda processing methods. The service provides access to information about the agenda processing methods that are supported by the service.

The *AgendaProcessing* service interface follows the factory design pattern: It is an interface for the creation of *descriptors* and *runtimes* of the agenda processing service. As usual, *descriptors* are used during acquisition of control knowledge and *runtimes* are used during application of control knowledge.

The acquisition part of the agenda processing service allows the global configuration of methods. The global method configuration is applied for *runtimes* of the agenda processing service when they are initialized.

Each agenda processing *runtime* is started before it is used and stopped when it is not used anymore. The purpose of agenda processing *runtimes* is to process agenda items with one of the methods that is provided by the service. Additionally, the service allows dynamic configuration of agenda processing methods.

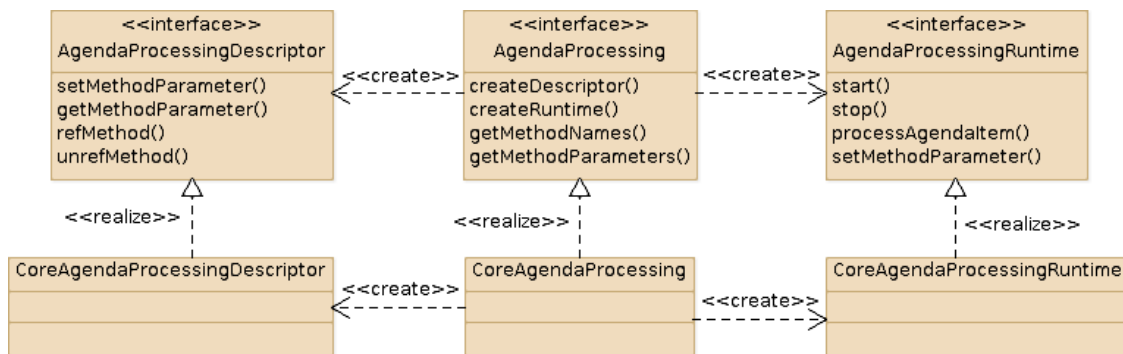


Figure 5.13 UML diagram that shows the *CoreAgendaProcessing* implementation of the *AgendaProcessing* interface.

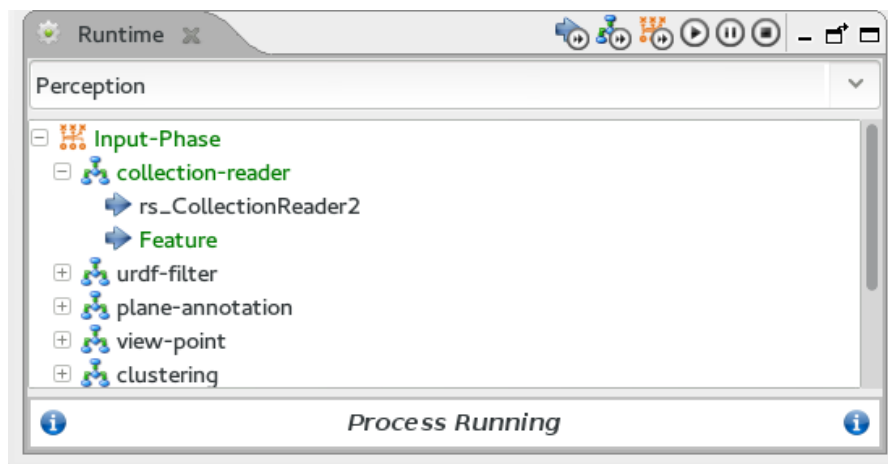
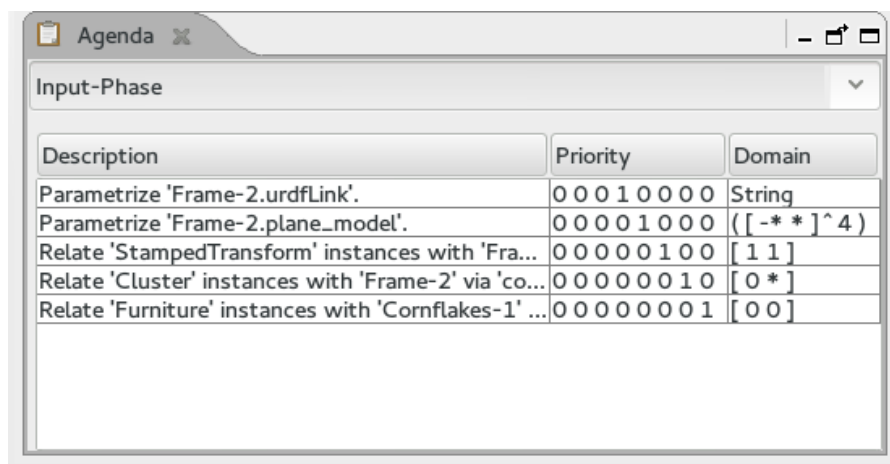


Figure 5.14 Strategy monitoring user interface that shows the selection of control strategies, control flows and agenda processing methods. The user interface allows to start, stop and pause the control process as well as it allows to process a single strategy, flow and agenda processing method.

The core of the control framework only implements a *core* agenda processing service that implements common functionality like the parametrization of attributes with an user defined value. This architecture is illustrated in figure 5.13.

The user interface of the control framework supports monitoring of the control process. Validation of control knowledge is simplified by this user interface because the knowledge engineer has the possibility to start, to stop and to pause the control process at any point (see figure 5.14). Additionally, the agenda of different control stages is shown to the knowledge engineer (see figure 5.15).



The screenshot shows a window titled "Agenda" with a dropdown menu set to "Input-Phase". Below the menu is a table with three columns: "Description", "Priority", and "Domain". The table contains five rows of agenda items, each with a binary priority string and a domain value.

Description	Priority	Domain
Parametrize 'Frame-2.urdfLink'.	00010000	String
Parametrize 'Frame-2.plane_model'.	00001000	([-*]^4)
Relate 'StampedTransform' instances with 'Fra...	00000100	[11]
Relate 'Cluster' instances with 'Frame-2' via 'co...	00000010	[0*]
Relate 'Furniture' instances with 'Cornflakes-1' ...	00000001	[00]

Figure 5.15 Agenda monitoring user interface that shows a prioritized list of agenda items for a particular stage of the control procedure. Additionally, this interface can be used by the knowledge engineer to manually process agenda items (using the *user question* agenda processing method). The priority column represents the estimates of agenda selection criteria.

Empirical Investigations

In this chapter, it is investigated how well the perception control framework performs as a top-down attention mechanism. The evaluation is based on real data that was gathered from the sensors of the PR2 (see section 2.2.2) in the reference kitchen (see section 2.2.1).

In section 6.1, modalities of the empirical investigations are discussed. The validation methods which are used in the empirical investigations are described in this section. Furthermore, system parameters with influence on the attention mechanism are identified and discussed. Finally, the acquisition of the training data for the Bayes' classifier (see section 4.4.3) is discussed in this section.

The classification of recognized objects is an important aspect for visual search tasks. Such tasks can only be fulfilled when the control mechanism is able to identify one of the recognized objects as task entity. The perception control framework uses a Bayes' classifier that uses the outcomes of perception methods as feature vectors. For the success of the perception control framework in visual search tasks, it is important that the Bayes' classifier is able to reliably predict the types of recognized objects. Additionally, the framework uses a selection method for control decisions that is inspired by computational attention (see section 4.4.2.3). The method uses the classifier in order to prefer recognized objects with annotations which are common for one of the task targets. The Bayes' classifier is investigated empirically in section 6.2 in order to make sure that it is suitable for the classification of kitchen items in a small validation domain. This chapter continues with an investigation of the influence of system parameters on the perception control framework in section 6.3. For example, the prioritization of features which are used for classification is such a system parameter. The objective of the experiments which are discussed in section 6.3 is the justification of a standard configuration for following validation experiments. Finally, the perception control framework is investigated for varying tasks in reference scenarios (see section 2.3) in section 6.4.

6.1 Experiment Modalities

In this section, the modalities for the empirical investigations in this chapter are described. The validation is based on methods which can be used to investigate particular properties of the perception control framework and its components. The set of validation methods which are used in this chapter is discussed in section 6.1.1. The experiment procedures are described in section 6.1.2 followed by a description of system parameters in section 6.1.3. Finally, the acquisition of the training data that is used for the validation of the perception control framework is discussed in section 6.1.4.

6.1.1 Validation Methods

Many computational attention systems are evaluated from a psychophysical perspective by comparing the results with ground-truth eye fixation data that was gathered using an eye tracking device. From a technical perspective, computational attention systems can be evaluated based on their usefulness in applications. The psychophysical accuracy of attention systems in technical domains (such as mobile robotics) is not necessarily important for the success of the computational attention mechanism in a particular application (e.g., as a front end for object recognition).

For top-down attention systems, the current task is clearly specified and validation methods can easily determine if a task target is selected by the perception control framework or not (hit rate). Note that the attention methods are not responsible for the recognition of objects. They can only be used to prefer particular regions, objects and features. For the object recognition, a naive Bayes' classifier is used which can be evaluated in terms of detection rate and rejection rate. Additionally, this classifier is used in one of the selection methods in order to support selection based on similarity to task entity classes.

Hit Rate In the annotation phase, each control decision specializes aspects of recognized objects. A hit is defined as a control decision that specializes the subject of the current task and a miss is defined as a control decision that specializes a model instance that is not subject of the current task. In the following, let CH denote the number of control decision hits and let CM denote the number of control decision misses. The hit rate $\phi_{hit} \in \mathbb{R}$ can be written as:

$$\phi_{hit} = \frac{CH}{CH + CM} \quad (6.1)$$

Analogues, the miss rate $\phi_{miss} \in \mathbb{R}$ can be written as:

$$\phi_{miss} = \frac{CM}{CH + CM} \quad (6.2)$$

The objective of selection methods is to reduce the overall number of control decisions which are needed to fulfill a visual search task. Assumed that a task is fulfilled, a low hit rate indicates that only few features were needed for the detection of the task targets. Furthermore, a low number of control decisions (compared to the number of possible decisions) indicates that the selection methods are able to concentrate the perception methods on recognized objects based on task demands. Note that the number of control decisions is maximal for tasks which cannot be fulfilled because there is no mechanism implemented that is able to stop the visual search early without positive result (i.e., the perception is controlled so that it looks at each detail in such cases). Thus, only fulfilled tasks are considered in this chapter.

NSS Score The hit rate does not take into account the number of recognized objects. The number of recognized objects has influence on the hit rate because the number of possible control decisions is proportional to the number of recognized objects. In order to validate that the methods perform better than by random chance, the NSS score (which is used for the validation of many attention systems – see section 3) is adapted for the scope of this thesis.

Let $D = \{D_0, \dots, D_n\}$ be the set of possible control decisions with $D_i = 1$ for hits and $D_i = 0$ misses. These values represent the “saliency” of the control decisions. The mean μ_D over all possible control decisions can be written as:

$$\mu_D = \frac{1}{n} \sum_i^n D_i \quad (6.3)$$

A high mean value (bigger than 0.5) indicates that most possible control decisions are hits while a low mean value (smaller than 0.5) indicates that there are more possible control decisions which are misses.

The variance σ_D over all possible control decisions can be written as:

$$\sigma_D = \frac{1}{n} \sum_i^n (D_i - \mu_D)^2 \quad (6.4)$$

The variance represents the balancing between hits and misses. A low value indicates that the selection is dominated by either hits (high mean) or misses (low mean) while a high value indicates that hits and misses are balanced.

Let $D' = \{D'_0, \dots, D'_m\} \subseteq D$ be the set of selected control decisions. The NSS score is defined as the average of normalized saliency over all selected control decisions. For the normalization, each response is transformed to have zero mean and a unit standard deviation. The NSS score ψ_{NSS} can be written as:

$$\psi_{NSS} = \frac{1}{m} \sum_i^m \frac{D'_i - \mu_D}{\sigma_D} \quad (6.5)$$

$\psi_{NSS} = 0$ indicates that the selection method does not perform better than by random chance because the average saliency of selected control decisions is not higher than the mean saliency over all possible control decisions. On the other hand, values greater than zero indicate that the saliency of selected control decisions is higher than the average saliency over all possible control decisions (i.e., a selection that is better than by chance).

Pruning Rate The pruning rate ϕ_{prune} measures the amount of unconsidered control decisions. Let n be the number of selected control decisions and let m be the number of all control decisions that can be performed for a particular set of objects. The pruning rate can be written as:

$$\phi_{prune} = \frac{m - n}{m} \quad (6.6)$$

Detection and Rejection Rate The classification based selection and processing methods can be validated in terms of detection rate. The detection rate measures the rate of correct classifications (true positives) over the number of training samples for the target class (true samples). This does not take into account how the methods perform for training samples of other classes (false samples). The rejection rate measures the rate of correct rejections (true negatives) over the number of false samples. This is illustrated in following table:

	Positive Classification	Negative Classification
True Sample	True Positive (TP)	False Negative (FN)
False Sample	False Positive (FP)	True Negative (TN)

The detection rate ϕ_{det} can be written as:

$$\phi_{det} = \frac{TP}{TP + FN} \quad (6.7)$$

$\phi_{det} = 1$ indicates perfect detection (i.e., every true sample is correctly classified) while lower values indicate that there are rejected true samples.

Analogous, the rejection rate ϕ_{rej} can be written as:

$$\phi_{rej} = \frac{TN}{TN + FP} \quad (6.8)$$

$\phi_{rej} = 1$ indicates perfect rejection (i.e., every false sample is correctly rejected) while lower values indicate that there are false samples which are incorrectly classified as the target class.

AUC Score Detection and rejection rate depend on the classification threshold τ . The classification threshold determines the minimum confidence that is required in order to accept a classification. All classifications are accepted for $\tau = 0$ while only classifications with maximum confidence are accepted for $\tau = 1$.

The ROC (see section 3) curve can be obtained by plotting the the detection rate ϕ_{det} against the false positive rate $1 - \phi_{rej}$ for different classification thresholds. The area under this curve is the AUC score ψ_{AUC} . $\psi_{AUC} = 1$ indicates perfect prediction while $\psi_{AUC} = 0.5$ indicates that the model does not perform better then by random chance. The classification threshold is a sensitive system parameter of the perception control framework because visual search targets are identified based on the outcomes of a classifier.

6.1.2 Experiment Procedure

If not stated differently in the experiment description, the experiment procedure is as follows: Initially, the description of the semantic map is parsed and model concepts are instantiated for each entity in the semantic map (e.g., the kitchen table). Furthermore, the task description is parsed and model concepts are instantiated for each task entity. The task description contains additional knowledge about task entities such as the color of the object or the location of the object (i.e., a spatial relation to a semantic map entity).

Real data that was gathered from the sensors of the PR2 in the reference kitchen (see section 2.3) is used for the validation. The sensory input data initiates the clustering perception procedure which is reliable when the objects of interest are located on top of a supporting plane such as the top of the kitchen table. For the validation of the perception control framework, samples are only used if the clustering result is acceptable (other samples are manually removed from the validation sample set). The clustering procedure is also controlled by the proposed perception control framework but this procedure is not relevant for object-based attention because an object representation is only available after clustering. Thus, the validation concentrates on the annotation and classification phase.

The annotation phase is initiated by the instantiation of model concepts which correspond to recognized kitchen items. The subject of the annotation phase is the specialization of the recognized objects (i.e., computing annotations for the recognized object). The first attended object is selected randomly and the attention shifts based on control decisions in the following. The classification phase is activated as soon as the confidence for a particular object classification exceeds the classification threshold. Finally, model instances which correspond to recognized objects are integrated into task entities after classification if the types are equal and attributes and relations are compatible. The procedure terminates when each of the task entities is associated to a recognized object.

For the validation of the perception control framework, experiments are divided as follows:

- A The first class of experiments is dedicated to the investigation of different annotations and their influence on classification. The validation of the suitability of annotations for classification is separated from the validation of selection methods. In these experiments, no selection

methods are used and the only feature of recognized objects is the investigated annotation. The validation samples (see section 6.1.4) are divided into training set ($\frac{2}{3}$ of all samples) and testing set ($\frac{1}{3}$ of all samples). The training samples are used to train the probabilistic model of the Bayes' classifier. In these experiments, the testing samples are successively processed for each of the validation object classes. The investigated annotation is computed for each sample and the recognized object is classified based on this feature in the next step. For the validation, the classification results (TP, TN, FP and FN) are logged in order to compute the detection rate and the rejection rate for a particular classification threshold and validation object class. The classification threshold is varied as follows: 0.0, 0.1, 0.2, . . . 1 (i.e., 10 different classification thresholds).

- B The experiments of this class are dedicated to the investigation of different system parameters of selection methods and their influence on the NSS score of the perception control framework in different reference scenarios (see section 2.3). The set of annotations which is used for these experiments is defined in section 6.2.5. In this class of experiments, only the investigated selection method is considered by the control framework. Thus, the selection of the next control decision only depends on the investigated selection method and system parameter. In these experiments, the reference scenarios are successively processed and control decisions are logged for later investigation. The experiment terminates as soon as the classification confidence of the target object exceeds the classification threshold. The investigated parameter is varied for each of the reference scenarios in order to examine the influence of this parameter on the NSS score of the investigated selection method.

- C The last class of experiments is dedicated to the investigation of the NSS score of the perception control framework with varying tasks in different reference scenarios (see section 2.3). The set of annotations, the set of selection methods, the prioritization of selection methods and system parameters which are used for these experiments are defined in sections 6.3.4 and 6.2.5. On the top level, these experiments are repeated for all visible objects as targets for the visual search task. Furthermore, the experiments are repeated with different expected annotations for the task entities (i.e., knowledge about the annotations of task entities). The parametrization of task entities is limited to the *Color Ratio* annotation (i.e., search for a particular color) and the *Goggles* annotation (i.e., search for a particular label). Finally, the kitchen table is segmented into border and center area in order to represent additional knowledge about the spatial location of task entities. The experiment is repeated for tasks where the spatial location of task entities on the kitchen table is known in advance.

6.1.3 System Parameters

In this section, different system parameters with influence on the NSS score of the perception control framework are identified and described. Additionally, the default values which are used for the validation are justified for some parameters in this section. Appropriate values for other system parameters are empirically investigated in section 6.3.

Classification Features Classification methods are used in the perception control framework in order to estimate the probability that a recognized object belongs to a object class of an task entity (i.e., the visual search target class). The classification relies on a set of features which are obtained by perception methods of *RoboSherlock* (see section 2.2.3).

The set of considered annotations is as follows:

- *Semantic Size:*
This feature roughly divides objects into *small* and *big* objects based on the maximum distance between points which belong to the object. Thus, the semantic size is a nominal feature. This feature might be useful to distinguish between cutlery such as spatulas and larger kitchen items such as corn flakes packages.
- *Semantic Shape:*
This feature roughly divides objects into different basic shapes such as *flat*, *box*, *round* or *cylinder*. Thus, the semantic shape is a nominal feature. This feature might be useful to distinguish between boxy kitchen items such as a milk package and round kitchen items such as a pancake maker.
- *Color Ratio:*
The color of recognized objects is decomposed into 6 channels: *blue*, *yellow*, *red*, *green*, *white* and *black*. The color ratio (i.e., the number of channel responses over the number of pixels) is computed for each of the channels. Thus, the color ratio feature is a vector of 6 continuous values between 0 and 1. It is expected that this feature is useful for the classification of objects with strong dominant colors such as the pancake maker.
- *Goggles:*
The *Goggles* web service is able to provide different information for a particular ROI that corresponds to a recognized object. The information may include visible text and the company name for a visible company logo. Thus, the *Goggles* feature is a string feature and recognized objects may be annotated with multiple *Goggles* features. This feature might be useful for the classification of textured objects with text and company logos such as milk packages and ice tea packages.

- *LINE-MOD*:

The *LINE-MOD* feature is the label of the class with highest classification confidence according to the *LINE-MOD* classification algorithm. Thus, this feature is a nominal feature of all possible class labels. The *LINE-MOD* classification algorithm might work well for untextured objects such as pancake makers and cups.

The set of annotations is empirically investigated in section 6.2.

Classification Threshold The Bayes' classifier relies on a classification threshold τ that is used in order to determine if a particular classification should be accepted or rejected. Each classification option has a corresponding confidence. Classifications are accepted if the corresponding confidence is higher than the classification threshold. In the scope of this thesis, the classification threshold is a sensible system parameter because it has influence on the success of the control framework for visual search task. The classification threshold is empirically investigated in section 6.2.

Preference of Novel Objects Initially, no annotations are attached to recognized objects. The resulting classification threshold is very low. This may result in a concentration on the first attended object because annotations enhance the classification confidence compared to objects without annotations. The control framework allows to define a constant classification confidence for objects without annotations. For the validation, the classification confidence of novel objects is set to 0.3. Thus, novel objects are preferred as long as the classification confidence of other objects does not exceed this threshold.

Edit Distance Weight The *task entity similarity* method (see section 4.4.2.3) estimates the minimum similarity to task entities based on the edit distance between model instances and the confidence of the Bayes' classifier. The edit distance d_{edit} (see section 4.4.2.2) is used to prefer recognized objects with parametrization similar to task entities. The *task entity similarity* method uses a weight parameter ω in order to weight the edit distance against the task classification confidence c :

$$task\text{-similarity} = \omega d_{edit} + (1 - \omega)c \quad (6.9)$$

For $\omega = 0$, the parametrization of task entities is completely ignored and only the object classes are considered. On the other hand, the edit distance is not able to enhance parameters which are discriminative for the object class. Thus, the weight should be chosen so that both aspects are reflected in the task similarity estimate. The edit distance weight parameter is investigated in experiment B1 (see section 6.3.1).

Minimum Task Relation Similarity The *task entity similarity* method selects control decisions (e.g., relation decisions) based on the edit distance between model instances and the confidence of the Bayes' classifier. The method allows to define a minimum similarity (s_{min}) for relation decisions when the corresponding relation is specified in the task description. For example, knowledge about the color of task targets is attached to the task entity using the *has-annotation* relation. In this case, the s_{min} yields in a preference of relation decisions which are used to annotate the recognized object with the visible color ratio. On the other hand, the minimum similarity of task relations should not dominate the edit distance and the classifica-

tion confidence in all cases in order to allow to concentrate on a particular object before this annotation is computed for each visible object.

Relation Weighting The transition cost of paths in the *relational task distance* approach is based on an user defined weighting for relations. It is possible to define individual weights for the concept graph and the instance graph. The set of used relations is as follows: *is-a* relation (defines the taxonomy of model concepts), *contains* relation (used to attach task entities to the instantiation of the task concept – see section 4.3), *located-at* relation (defines the spatial location of recognized objects) and *has-annotation* relation (used to attach annotations to recognized objects). Taxonomy edges are only added to the concept graph. Thus, the taxonomy weight for the instance graph can be set to ∞ . The *has-annotation* relation augments an object. Thus, it is plausible to assume a transition cost of 0.0 for this relation. Additionally, a transition cost of 0.0 can be assumed for the *contains* relation because it is always used for relations with task entities. Finally, the *located-at* relation expresses a stronger semantic relation between objects then the taxonomy.

Connections via the concept graph represent weaker semantic relations then connections in the instance graph. Thus, the transition cost should be increased in the concept graph compared to the instance graph. For the validation, the concept graph is completely ignored because recognized objects are always connected directly to a task entity (i.e., the kitchen table). In order to be able to obtain different distance estimates for recognized objects, a relation is introduced that represents spatial pooling on the table. The table is roughly segmented into border area and center area and special relations (*at-border* and *at-center*) are used to represent the spatial position of objects on the table. For the spatial pooling, a transition cost of ∞ is used for the *located-at* relation because it can not distinguish between different objects on the same table. Finally, the cost of transitions via the *at-border* relation is set to 0.0 and the cost of transitions via the *at-center* relation is set to ∞ if the task target is located at the border of the table. Otherwise, the weight of the *at-border* relation is set to ∞ and the weight of the *at-center* relation is set to 0.0.

Selection Methods The perception control framework provides a set of methods which are dedicated to the selection of control decisions (see section 4.4.2). The framework allows the combination of different selection criteria. Each criterion can be interpreted as a preference on particular objects, regions or features. The selection, configuration and prioritization of selection criteria in the annotation phase has influence on the NSS score of the perception control framework. The set of considered selection criteria is as follows: *Relational task distance* method (see section 4.4.2.1), *Task entity similarity* method (see section 4.4.2.3), the selection based on patterns of control decisions (see section 4.4.2) and the selection based on continuity (see section 4.4.2).

The relational task distance selection method restricts the attention to semantic locations and the task entity classification selection method restricts the attention to particular object classes and features. Thus, it is plausible to assign a higher priority to the relational task distance selection method. The attention methods are the primary selection methods. Thus, the priority of other selection methods should be lower than the priority of the attention methods. Furthermore, it is evident that the priority of the continuity selection method must be higher than the priority of pattern selection methods in order to allow the concentration on modified model instances. Nevertheless, it is not obvious that the continuity selection method yields in better results. Finally, the sequence of pattern selection criteria is used to define the ordering of considered annotations. It is desirable to prefer discriminative annotations (i.e., features with low variance) in order to quickly exceed the classification threshold.

6.1.4 Acquisition of Training Data

The classification of recognized objects is based on a probabilistic model. The model is learned based on a set of training samples which are acquired using a RGB-D sensor in the reference kitchen. Training samples are acquired for each kitchen item that appears in one of the reference scenarios (see section 2.3). The items are recorded from different view angles because the perception methods may not be rotation invariant (e.g., the *Goggles* method can only find visible text). As an illustration, the training samples for corn flakes packages are shown in figure 6.1. The result of the *LINE-MOD* method (i.e., the class label) is a feature for the Bayes' classifier. Thus, the *LINE-MOD* method and the naive Bayes' classifier must use different sets of training samples.

The set of considered object classes for validation is as follows: Corn flakes package, ice tea package, milk package, pancake tube, pancake maker, spatula and cups with different colors (blue, green and yellow). The validation set includes 30 samples from different view angles for each object class (i.e., in total 210 validation samples).



Figure 6.1 Training samples for a *Kellogg's Corn Flakes* package from different view angles.

6.2 Classification Experiments

The main objective of the proposed perception control framework is the focus of attention which is modeled as a set of selection criteria in the control framework. Visual search tasks require that the system is able to classify visible objects based on perceived features. Additionally, top-down attention mechanisms may utilize the knowledge about the statistics of object classes in order to focus on likely targets. The Bayes' classifier that is used in the control framework is validated in this section based on a set of experiments in which the detection and rejection rate is gathered for a set of training images (see section 6.1.4).

6.2.1 Experiment A1 - Goggles Annotation

In this experiment, the suitability of the *Goggles* annotation for the classification of the validation object classes is investigated (the experiment procedure is described in section 6.1.2). The annotations are used by the task entity classification method (see section 4.4.2.3). Thus, it is important that the results of the *Goggles* method are suitable for the Bayes' classifier that is used in the selection method.

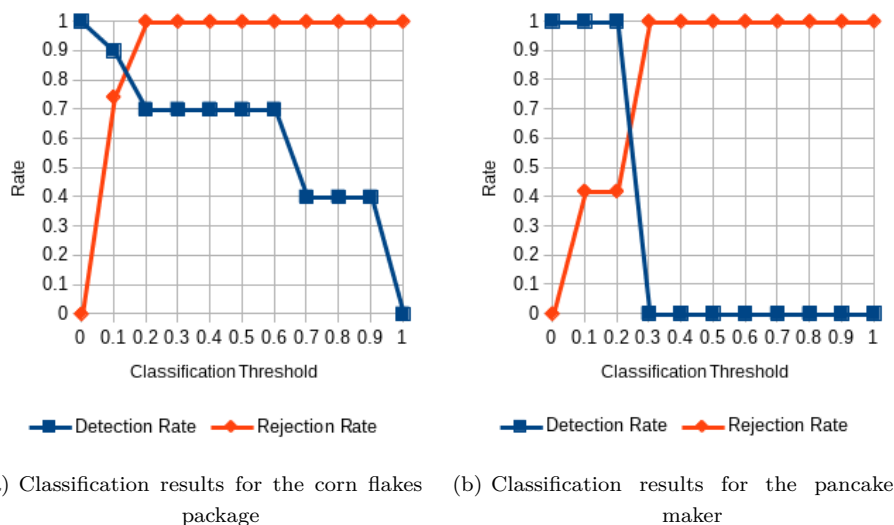


Figure 6.2 Sample results of experiment A1.

Assumptions and Experimental Setup In this experiment, no selection methods are used and the only features which are used for classification are *Goggles* annotations. Each *Goggles* feature is a string that represents the company name, visible text or something else that is visible according to the *Goggles* web service.

Hypothesis It is expected that the *Goggles* feature is not rotation invariant. Thus, it is expected that the feature values vary for training samples from different view angles and that the recognition rate is rather low. Additionally, it is expected that the results from different angles are lexically similar (e.g., a missing word or a wrong letter) and that the Bayes' classifier is able to use this lexical similarity for the classification of textured objects such as corn flakes packages and ice tea packages when the company name or logo is visible (usually the front side of the package). On the other hand, it is expected that this feature does not perform well for objects without visible company name or logo such as the pancake maker.

Results The classification based on the *Goggles* feature is better than by chance for all of the validation object classes. As expected, the best result is achieved for a textured object: For cornflakes packages, the classifier achieves a detection rate of 70% and a rejection rate of 100% for a classification threshold of 0.6 (see figure 6.2). The detection rate for other textured objects is worse than expected. For example, the detection rate of the milk package is at 0.3 for classification thresholds between 0.2 and 0.7. As expected, the detection rate of untextured objects quickly reaches zero for growing classification thresholds. For instance, the detection rate

for the pancake maker is at 0.0 starting from a classification threshold of 0.3 (see figure 6.2). Furthermore, the rejection rate quickly reaches 100% for growing classification thresholds. For all validation object classes, the rejection rate is at 100% starting from a classification threshold of 0.3. The results of this experiment are completely listed in appendix B.1.

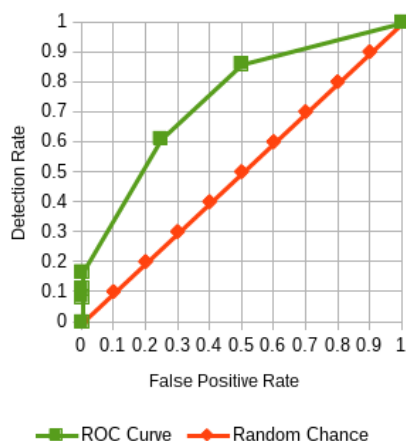


Figure 6.3 The ROC curve of the classification based on the *Goggles* annotation.

Discussion The *Goggles* annotation is more suitable for the classification of textured objects than for untextured objects because the *Goggles* service does not often return results for untextured objects. The best results are achieved for the corn flakes package because the *Goggles* service is able to recognize the company logo for testing samples where the package is visible sideways and it is able to recognize some text when the front side of the package is visible. This is not the case for the ice tea package and the milk package. Furthermore, the *Goggles* service does not always return results for testing samples where text is clearly visible for a human observer. For example, it seems like the service has some issues with the font that is used on the ice tea package. Thus, the detection rate is rather low over all training samples (e.g., about 0.16 for a classification threshold of 0.6). Nevertheless, the rejection rate is rather high over all training samples (e.g., about 1.0 for a classification threshold of 0.6) because the service returns empty results in most cases (which yields in a low classification confidence). Overall, the classification based on the *Goggles* annotation is better than by chance (see figure 6.3). Thus, it can be used by the Bayes' classifier which is used in the perception control framework.

6.2.2 Experiment A2 - LINE-MOD Annotation

The *LINE-MOD* method is able to predict class labels for recognized objects based on set of training samples (about 20 per object class). The suitability of the *LINE-MOD* annotation for the classification of the validation object classes is investigated in this experiment (the experiment procedure is described in section 6.1.2).

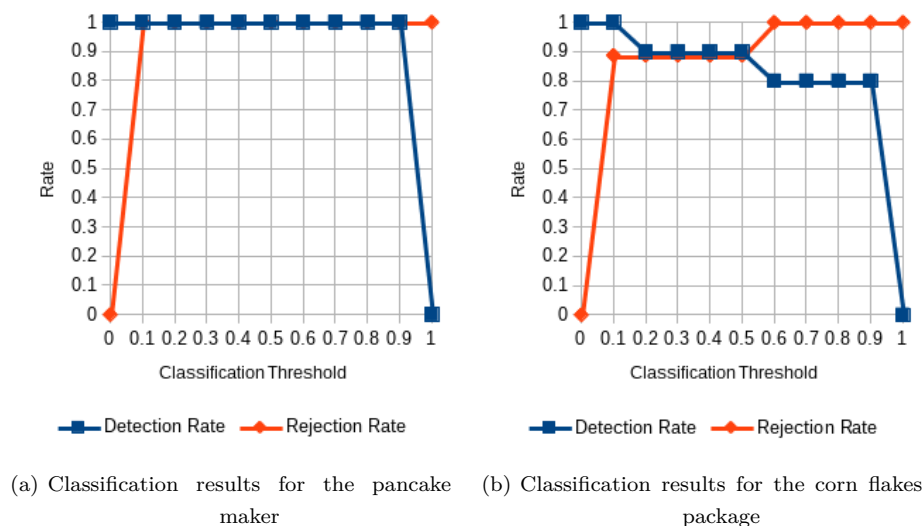


Figure 6.4 Results of Experiment A2.

Assumptions and Experimental Setup In this experiment, no selection methods are used and the only feature that is used for classification is a class label (nominal feature) that was predicted by the *LINE-MOD* method. The *LINE-MOD* method is trained for the set of validation object classes based on a set of training samples (about 20 samples per validation object class). This set of training samples is distinct from the set of training samples that is used to train the Bayes' classifier.

Hypothesis It is expected that the *LINE-MOD* annotation is rotation invariant because it was trained with samples from different view angles. Thus, the predicted class label should be consistent for most view angles of the testing samples (contrary to the *Goggles* method where the visibility of text and logos depends on the view angle). Furthermore, it is expected that the predicted class label of untextured objects such as pancake makers or cups can be better used for classification than the class label of textured objects such as milk packages.

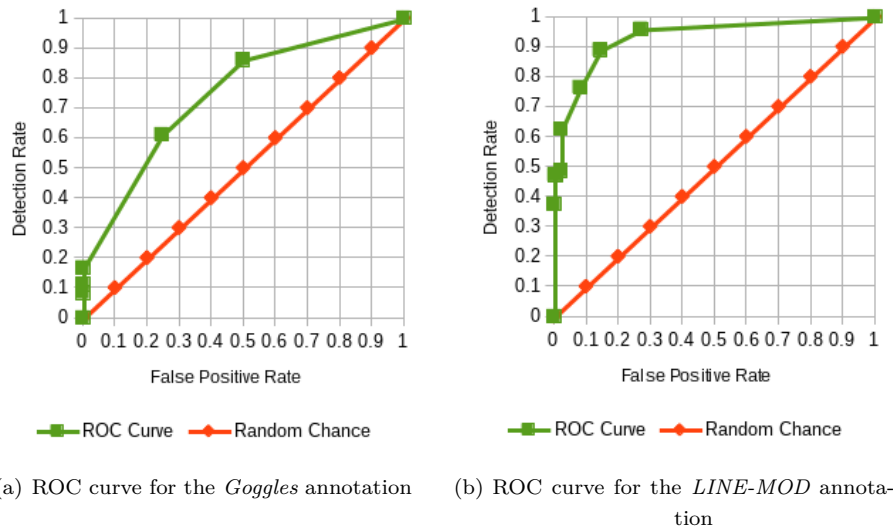


Figure 6.5 Comparison of experiment results for experiments A1 and A2.

Results As expected, the best result is achieved for an untextured object. The classification of the pancake maker is in fact perfect for the set of testing samples (see figure 6.4) and the detection rate for cups is about 67% up to a classification threshold of 0.9. Nevertheless, the result of for the spatula is worse. For the spatula, the detection rate is at 0% starting from a classification threshold of 0.4. Surprisingly, the detection rate and the rejection rate are rather high for the corn flakes package (80% detection rate and 100% rejection rate for a classification threshold of 0.9) and the ice tea package (70% detection rate and 100% rejection rate for a classification threshold of 0.8). Furthermore, the rejection rate quickly raises for growing classification thresholds. For all validation object classes, the rejection rate is above 98% starting from a classification threshold of 0.6. The results of this experiment are listed completely in appendix B.2.

Discussion The *LINE-MOD* annotation yields in good detection rate (above 60% for a classification threshold of 0.8) for about half of the validation object classes (pancake maker, cup, ice tea package and corn flakes package). For other object classes, the rejection threshold quickly reaches 100% (for classifications thresholds starting from 0.4). The best results are achieved for the pancake maker where the testing samples can be predicted perfectly up to a classification threshold of 0.9. This might be related to the consistent shape of the pancake maker from different view angles. Surprisingly, the *LINE-MOD* annotation suits well for the classification of the ice tea package and the corn flakes package in this validation domain. Actually, the *LINE-MOD* method predicts wrong class labels for the ice tea package and the corn flakes package. For the corn flakes package, the *LINE-MOD* method predicts the class label that corresponds to the

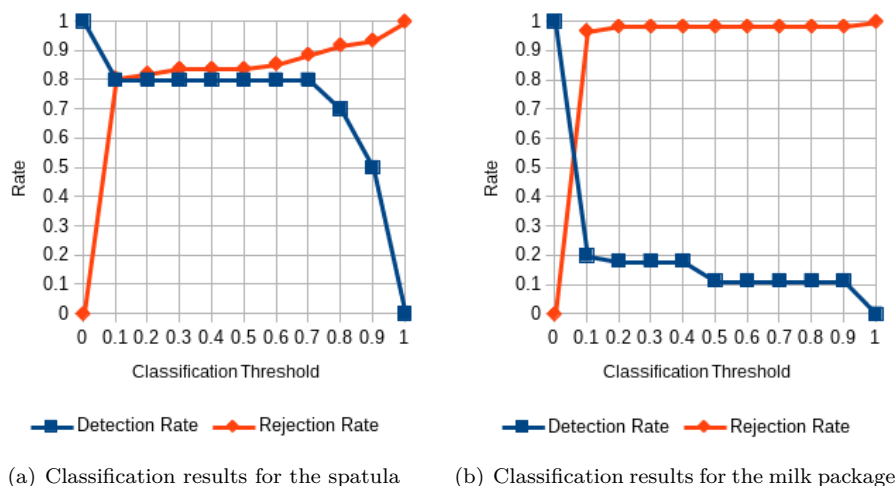


Figure 6.6 Results of Experiment A3.

pancake tube or the class label that corresponds to the milk package most often. For the ice tea package, the *LINE-MOD* method predicts the class label that corresponds to the pancake tube in most cases. These false predictions seem to work well in this validation domain but it is expected that this is not the case in larger domains. Overall, the classification based on the *LINE-MOD* annotation is better than by chance and better than classification based on the *Goggles* feature (see figure 6.5).

6.2.3 Experiment A3 - Color Ratio Annotation

In this experiment, the suitability of the *SemanticColor* annotation (see section 2.2.3) for the classification of the validation object classes is investigated (the experiment procedure is described in section 6.1.2).

Assumptions and Experimental Setup In this experiment, no selection methods are used and the only feature that is used for classification is a color ratio as computed by the *SemanticColor* method. Color is represented as a 6 dimensional vector of continuous values between 0 and 1 where the dimensions correspond to different color channels (*blue, yellow, red, green, white* and *black*).

Hypothesis It is expected that the classification based on the *Color Ratio* feature is better than by chance and that it works similar well for textured and untextured objects. On the other

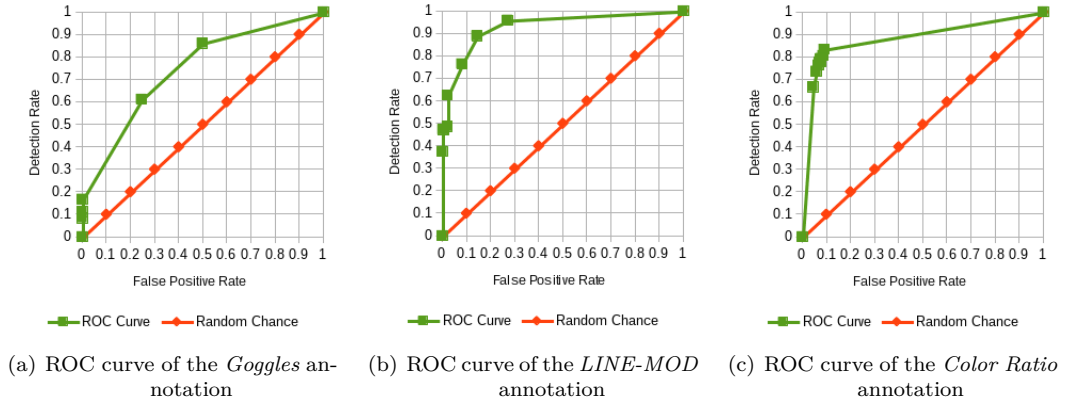


Figure 6.7 Comparison of experiment results for experiments A1, A2 and A3.

hand, it is expected that there are more false predictions compared to the *Goggles* annotation and *LINE-MOD* annotations (i.e., lower rejection rate).

Results The overall results of the classification based on the *Color Ratio* feature are surprisingly good. As expected, the results are similar for textured and for untextured objects. For example, the spatula can be classified with 70% detection rate up to a classification threshold of 0.7 (see figure 6.6) and the ice tea package can be classified with 70% detection rate up to a classification threshold of 0.9. The best result is achieved for the pancake maker where the detection rate is at 90% for a classification threshold of 0.9. As expected, the number of false positives is higher for the *Color Ratio* compared to previously discussed annotations. Nevertheless, the number of false positives is still low for classification thresholds above 0.5. For example, there are only 2 false positives for the pancake maker out of 62 false samples for most tested classification thresholds. The results of this experiment are listed completely in appendix B.3.

Discussion The *Color Ratio* feature performs better than expected because the dominant color is in fact a discriminative feature in the set of validation samples (e.g., all plain yellow objects in training samples are cups). Thus, the *Color Ratio* feature is suitable for the classification in such a small domain. Nevertheless, the feature might be less useful for the classification when the domain is augmented. The detection rate and rejection rate of the classification based on the *Color Ratio* feature depends less on the classification threshold compared to the *LINE-MOD* and *Goggles* annotation (see figure 6.7). The average detection rate lies between 0.6 and 0.9 for all classification thresholds and the average false positive rate lies between 0.0 and 0.1 for all classification thresholds. Thus, the feature is suited to be used for classification in this validation domain.

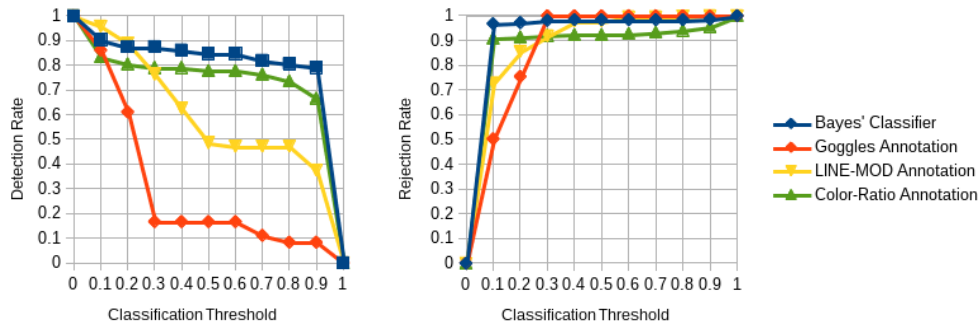


Figure 6.8 Classification results of the Bayes' classifier in comparison to the results of classification based on individual features.

6.2.4 Experiment A4 - Bayes' Classifier

In this section, the suitability of the Bayes' classifier for the classification of the validation object classes is investigated (the experiment procedure is described in section 6.1.2).

Assumptions and Experimental Setup In this experiment, no selection methods are used and the only considered processing method is the Bayes' classifier. All available features are considered in this experiment. For each training sample, all selected features are computed. Thus, the prioritization of annotations is irrelevant for this experiment. The classification phase is initiated when all annotations are computed and the experiment terminates as soon as the classification result is computed.

Hypothesis It is expected that the Bayes' classifier (which combines the outcomes of different perception methods) outperforms a classification that is based on the outcome of individual perception methods in terms of detection rate and rejection rate.

Results The overall results of the Bayes' classifier show that the selected set of features can be used to detect objects with a rate of 80% up to a classification threshold up to 0.9 while the rejection rate is at 100% for classification thresholds above 0.2 (see figure 6.8). The best results are achieved for the pancake maker with perfect detection and rejection and for the ice package where the detection rate is at 80% for a classification threshold of 0.9. The worst detection is achieved for the pancake tube where the detection rate is at 60% for a classification threshold of 0.9. The results of this experiment are listed completely in appendix B.4.

Discussion The results show that the Bayes' classifier achieves a better detection rate than classification which is based on individual annotations. The detection rate is always higher than

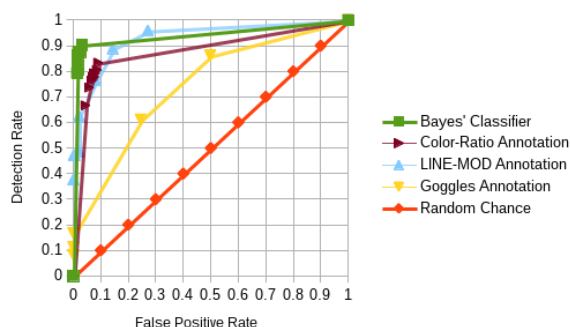


Figure 6.9 ROC curve for the Bayes' Classifier in comparison to the ROC curves for classification based on individual features.

the detection rate for individual annotations except of compared to the *LINE-MOD* annotation with a classification threshold of 0.1. Nevertheless, the detection rate is higher for all tested classification thresholds above 0.1. Furthermore, the area under the ROC for the Bayes' classifier is bigger than the area under the ROC for individual annotations (see figure 6.9). Thus, the trade-off between detection rate and false positive rate is best for the Bayes' classifier compared to individual annotations.

6.2.5 Classification Parameters

It is expected that evidence which discriminates the validation object classes from each other enhances the recognition rate of the Bayes' classifier. In experiments A1, A2 and A3, it was investigated how well particular annotations can be used for the classification in the validation domain. The selection of the next control decision is influenced by the outcome of the Bayes' classifier. Thus, the selection of annotations is relevant for the NSS score of the perception control framework. All discussed annotations can be used to predict validation object classes better than by chance and are selected by default to be used in the Bayes' classifier. It is expected that the combination of the *LINE-MOD* feature (which works well for untextured objects) and the *Goggles* annotation (which works only for textured objects) enhances the overall detection rate of the Bayes' classifier. Additionally, the *Semantic Size* and the *Semantic Shape* are selected by default to be used by the perception control framework for following experiments. The experiments showed that the classification threshold should be chosen between 0.6 and 0.9 in order to obtain a good trade-off between detection rate and rejection rate. If not stated different, the classification threshold is set to 0.75 for following experiments.

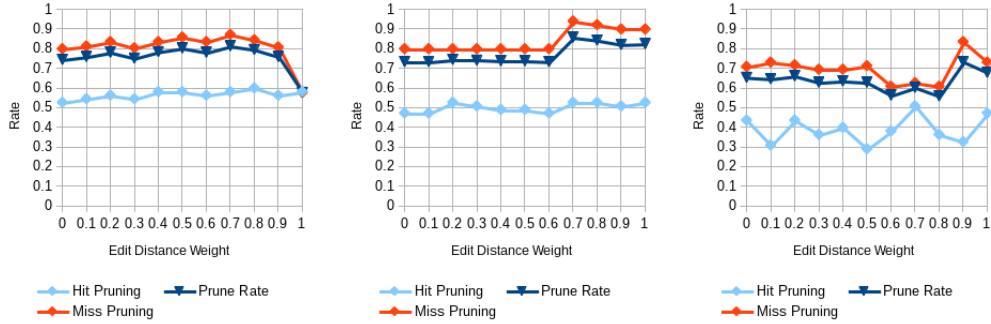


Figure 6.10 Pruning results for a visual search for a cornflakes package in the default scene where only the target class is known (left), where the label of the package is known in addition (middle) and the pruning results for the spatula in the default scene where the color of the spatula is known (right).

6.3 Configuration Experiments

The class of configuration experiments is dedicated to the discussion of suitable system parameters. Different aspects which are relevant for the hit rate of the perception control framework are investigated empirically in following sections.

6.3.1 Experiment B1 - Edit Distance Weight

The task entity classification selection method (see section 4.4.2.3) depends on an user defined weighting between classification confidence and edit distance between model instances. The weighting is expressed by the edit distance weight parameter $d_{edit} \in [0, 1]$.

Assumptions and Experimental Setup In this experiment, the only considered selection method is the task entity classification selection method. The edit distance weight parameter is varied as follows: 0.0, 0.1, \dots 1.0 (i.e., the experiment is repeated for 10 different values of d_{edit}) and the minimum task relation similarity s_{min} is set to 0.6.

Hypothesis It is expected that d_{edit} has no influence on the pruning of control decisions for visual search tasks where only the target class is known. For $d_{edit} = 1.0$, it is expected that the selection is not better then by random chance for tasks where only the class is known. Furthermore, it is expected that additional knowledge about task entities yields in a higher prune rate compared to task without additional knowledge about task entities. Finally, it is expected that the selection of control decisions is better then by chance in each case.

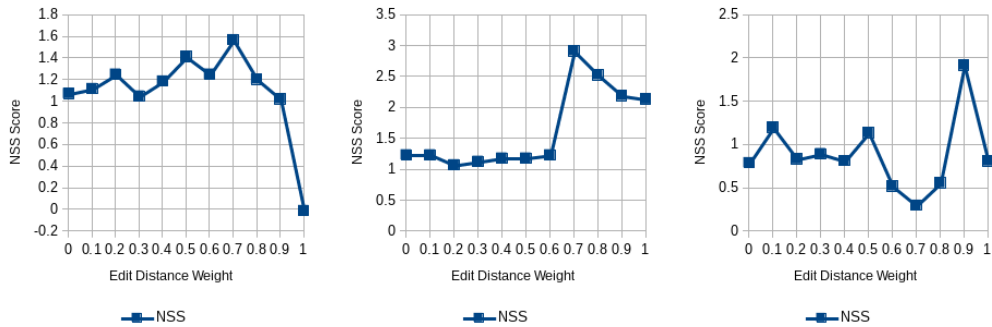


Figure 6.11 NSS score results for a visual search for a cornflakes package in the default scene where only the target class is known (left), where the label of the package is known in addition (middle) and the pruning results for the spatula in the default scene where the color of the spatula is known (right).

Results The results show that the control procedure is able to prune about 80% of all control decisions for the cornflakes package in the default scene (see figure 6.10). About 55% of all hit decisions and about 75% of all miss decisions are not selected by the control framework. On average, 2.8 hits are needed in order to identify the corn flakes package. The NSS score shows that the selection of control decisions is better than by chance for weights below 1.0 when no additional knowledge about the corn flakes package is available. The results of visual search for a labeled corn flakes package in the demo scene are similar to the results without additional knowledge up to a edit distance weight threshold of 0.7. Both, pruning rate (about 90%) and NSS score (between 2 and 3) are higher for edit distance weights above 0.7 when the label of the cornflakes package is known. The results of visual search for a black spatula in the default scene are worse than the results for cornflakes packages. About 70% of all control decisions were not selected in the control process. The selection does not benefit from the additional information about the color of the object up to a edit distance weight threshold of 0.9 where the pruning rate exceeds 80%.

Discussion This experiment shows that additional knowledge about annotations of target objects can enhance the pruning rate and the NSS score of the perception control framework. Knowledge about the annotation of the search target does not yield in a improvement of the rejection rate for edit distance weights below a threshold. The threshold is produced by the preference of relations which are specified in the task (minimum task relation similarity). The value of s_{min} is set to 0.6 for this experiment. Thus, control decisions for relations which are specified in the task are preferred over all control decisions with a task similarity below 0.6. For the cornflakes package, the task similarity exceeds 0.6 for edit distance weights above 0.7. For the spatula, the task similarity exceeds 0.6 for edit distance weights above 0.9. This means that the distance between the actual annotation and the annotation which is defined in the task

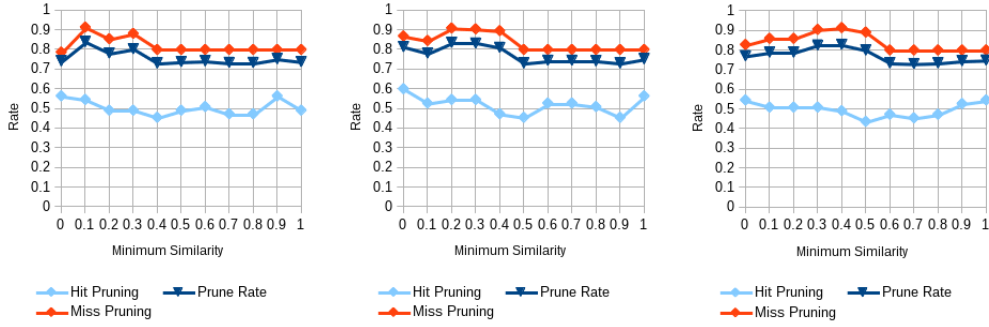


Figure 6.12 The influence of the s_{min} on the pruning rate of the perception control framework for the cornflakes package in the default scene. The label of the cornflakes package is known in advance. From left to right, the images show the result for $d_{edit} = 0.4$, $d_{edit} = 0.5$ and $d_{edit} = 0.6$.

description is larger for the spatula than for the cornflakes package (i.e., the specified label is a better match for the outcome of the *Goggles* method than the specified color for the outcome of the *Color Histogram* method). Furthermore, the experiment shows that the selection is not better than by chance when $d_{edit} = 1.0$ and no additional knowledge is available because the edit distance is the same for all objects in this case. Overall, the edit distance weight should be chosen below 1.0 so that tasks without additional knowledge can be performed better than by random chance. Furthermore, the experiment showed that the distance weight should be chosen above 0.6 because the edit distance is not able to shift away the attention from the relations which are specified in the task due to the minimum task relation similarity (0.6).

6.3.2 Experiment B2 - Minimum Task Relation Similarity

In the last experiment, the minimum task relation similarity (s_{min}) was set to 0.6. The experiment results showed that the specification of annotations for task entities only yields in a better NSS score for edit distance weights above 0.6. In this experiment, it is investigated how the system performs with a fixed edit distance weight below 0.6 and with a varying s_{min} parameter.

Assumptions and Experimental Setup In this experiment, the only considered selection method is the task entity classification selection method. s_{min} is varied as follows: 0.0, 0.1, ..., 1.0 (i.e., the experiment is repeated for 10 different values of s_{min}). Additionally, the experiment is repeated for different edit distances weights ($d_{edit} = 0.4$, $d_{edit} = 0.5$ and $d_{edit} = 0.6$).

Hypothesis It is expected that knowledge about annotations of objects yields in better pruning rate for particular minimum task relation similarity values. Furthermore, it is expected that the

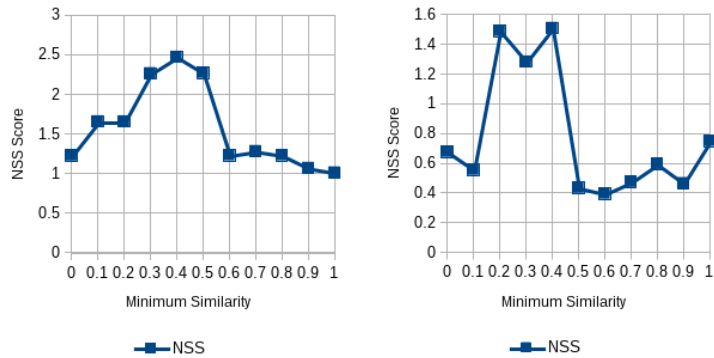


Figure 6.13 The influence of the minimum task relation similarity parameter on the pruning rate of the perception control framework for the cornflakes package (left image) and the spatula (right image) in the default scene for $d_{edit} = 0.6$. The label of the cornflakes package and the color of the spatula are known in advance.

range of the best values for s_{min} depends on the uniqueness of the user specified annotation. For unique values, the range might be wider compared to values with high ambiguity.

Results For the cornflakes search target, the results show that the pruning rate is about 90% for 3 different values of s_{min} (see figure 6.12) This hold true for all tested values of d_{edit} . Furthermore, the results show that the best values of s_{min} (in terms of NSS score) depend on the selected value of d_{edit} . The center of the range of best values is at $s_{min} = 0.4$ for $d_{edit} = 0.6$ and at $s_{min} = 0.3$ for $d_{edit} = 0.5$. The NSS score is enhanced compared to other samples within the range of best values for s_{min} . For example, the NSS score is slightly below 2.5 within the range and between 1.0 and 1.5 for other values of s_{min} . For the spatula search target, the maximum pruning rate is about 80%. For $d_{edit} = 0.4$ and $d_{edit} = 0.5$ there is only a single value for s_{min} where the best pruning rate is achieved.

Discussion The results for the cornflakes package are better then the results for the spatula for following reasons: The classification of the cornflakes package is more reliable and the user specified annotation is more accurate for the cornflakes package. For the cornflakes package, the task description contains the expected result of the *Goggles* perception method (the label “corn flakes”). In this scenario, the outcome of the *Goggles* method is equal to the label that was specified by the user in most cases. For the spatula, a color ratio is specified in the task description that corresponds to a dominant black color. The actual feature outcome of the *Color Ratio* method is similar to the user specified feature but not equal (i.e., a low edit distance). Additionally, the pancake maker has also a dominant black color. Thus, both objects can be preferred by the edit distance based selection. This yields in lower NSS score and lower pruning rate for the spatula compared to the cornflakes package. For both objects, best results are

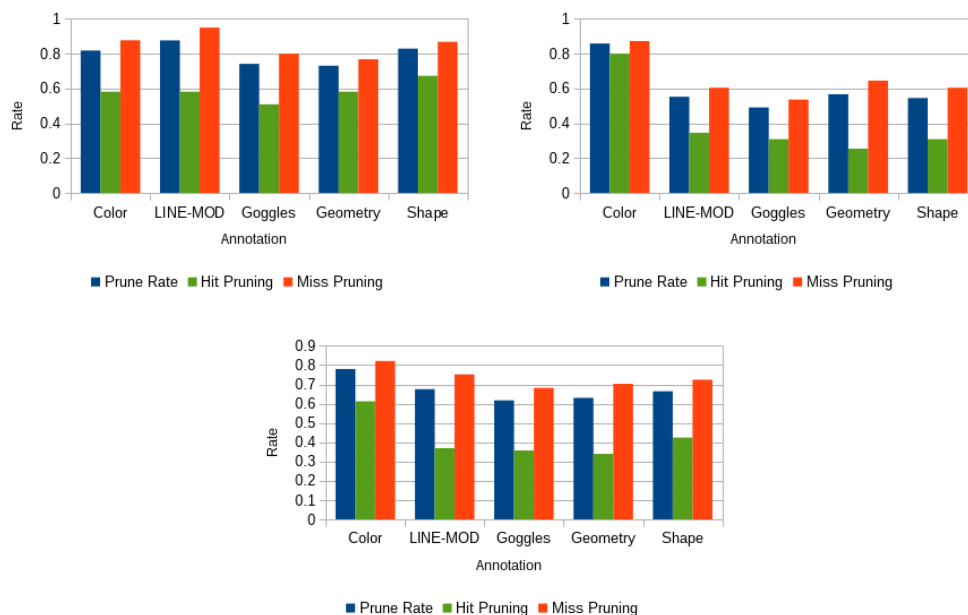


Figure 6.14 Pruning results of experiment B3. From left to right the results correspond to the cornflakes package, the ice tea package and the average over all tested object classes.

achieved for $d_{edit} = 0.6$ and s_{min} values between 0.3 and 0.4 (NSS score is slightly below 2.5 for the cornflakes package and slightly below 1.5 for the spatula).

6.3.3 Experiment B3 - Prioritization of Annotations

In this section, the influence of the pattern selection method on the hit rate of the perception control framework is investigated. The pattern selection method is used for the prioritization of control decisions based on the corresponding annotation (e.g., preference of control decisions which correspond to the *LINE-MODE* annotation). Thus, the investigated system parameter in this experiment is the preference for annotations.

Assumptions and Experimental Setup In this experiment, the investigated selection method is the pattern selection method. Additionally, the *task entity classification* method is used to prefer objects which are similar to task entity classes. The edit distance weight is set to 0.0 for this experiment so that the edit distance has no influence on the control procedure. The experiment is repeated with a preference for each of the considered annotations (*Color Ration*, *LINE-MOD*, *Goggled*, *Semantic Size*, *Semantic Shape*).

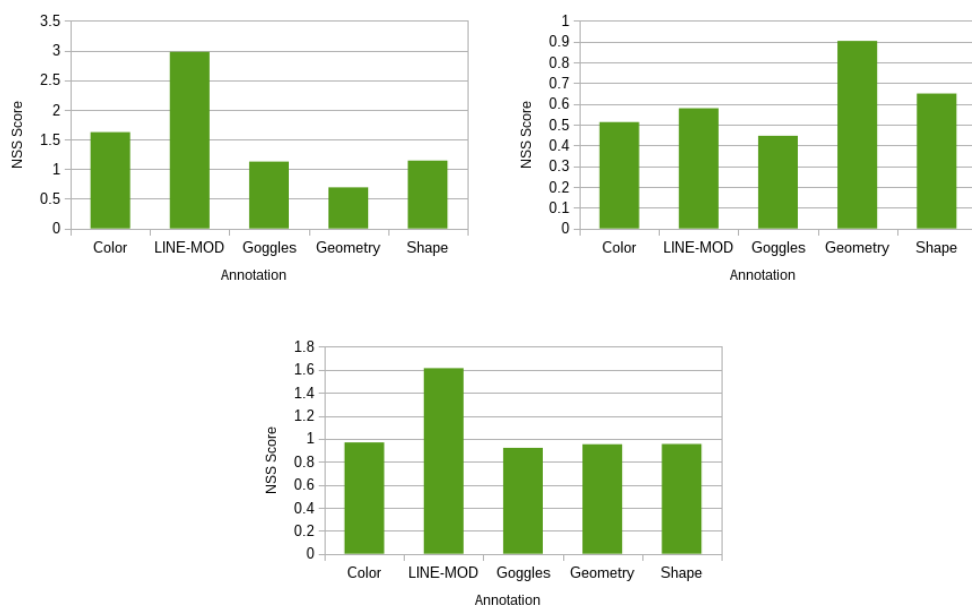


Figure 6.15 Overall results of experiment B3. From left to right the results correspond to the cornflakes package, the ice tea package and the average over all tested object classes.

Hypothesis It is expected that the preference of discriminative annotations yields in better hit rate because the classification threshold is exceeded with less control decisions. Furthermore, it is expected that the preference of the *Semantic Size* and the *Semantic Shape* annotations yields in a lower NSS score then the preference of the *Goggles* annotation, the *LINE-MOD* annotation and the *Color Ratio* annotation. Furthermore, it is expected that the preference of the *LINE-MOD* annotation yields in best results in terms of NSS score because it showed best result in the classification experiments (see section 6.2.2). Finally, it is expected that the *Color Ratio* attribute suits well for the pruning of control decisions.

Results As expected, preference of the *LINE-MOD* annotation yields in best *NSS* score in this experiment (see figure 6.15). Overall, the *NSS* score is at 1.61 for the *LINE-MOD* annotation while the *NSS* score of other annotations is slightly below 1.0. For example, the *NSS* score is low for the *LINE-MOD* annotation when the search target is an ice tea package (0.58). For cornflakes packages, the preference of the *LINE-MOD* annotation yields in a hit rate above 60% while the hit rate is below 50% for the other annotations. Finally, the pruning rate is best for the *Color Ratio* annotation (see figure 6.14). In this experiment, the preference of the color annotation yields in a hit pruning rate of 61% (i.e., 61% of all hits were not selected) and in a miss pruning rate of 82% (i.e., 82% of all misses were not selected). The result of other annotations is about 20% worse for the hit pruning rate and about 10% worse for the miss pruning rate.

Discussion In this experiment, the influence of the preference of different annotations on the NSS score and the pruning rate was investigated. On average, the annotations perform very similar to each other. The *NSS* score is slightly below 1.0 for all annotations except of the *LINE-MOD* annotation. This result is dominated by the outstanding *NSS* score for the *LINE-MOD* annotation when the search target is a cornflakes package (the hit rate is above 50% and cornflakes packages are identified with 2 decisions on average). For other search targets, the *LINE-MOD* annotation yields in a *NSS* score that is about the average *NSS* score of the considered annotation. Thus, the *LINE-MOD* annotation can safely be preferred over other annotations if the maximization of the *NSS* score is the objective. Additionally, the *Color Ratio* annotation showed good results. It is the second best annotation in terms of *NSS* score and the best annotation in terms of hit pruning rate and miss pruning rate. This is the case because there are not much ambiguities for the color of the validation object classes. One of the ambiguities is the black color of the spatula and the pancake maker. Additionally, the dark blue cup and the dark green cup may appear as black to the *Color Ratio* perception method. This yields in the lowest *NSS* score for the *Color Ratio* when the search target is a spatula (i.e., many objects with similar color were considered). Nevertheless, the overall *NSS* score is slightly above average for the *Color Ratio* annotation. Summarizing, the experiment showed that it might be desirable to prefer the *LINE-MOD* annotation over the other annotations. Furthermore, it might be desirable to prefer the *Color Ratio* annotation over all other annotations except of the *LINE-MOD* annotation.

6.3.4 System Parameters

The task similarity method depends on two system parameters: The edit distance weight d_{edit} and the minimum similarity s_{min} for relations which are specified in the current task. The edit distance measures the distance between annotations which are defined in task entities with annotations of recognized objects. Objects with low edit distance are preferred. The edit distance is combined with the classification confidence in order to yield a task similarity value that represents knowledge about the common appearance of the target object as well as knowledge about particular annotations of the task target. The weighting factor d_{edit} is sensible for the success of the attention mechanism because high values yield in ignorance of the classification confidence while low values yield in ignorance of annotations which are specified in the task. The edit distance weight parameter correlates with the minimum task relation similarity s_{min} that is used to prefer annotations which are defined in the task description. Experiments showed that good results can be achieved with $d_{edit} \in [0.5, 0.6]$ and $s_{min} \in [0.3, 0.4]$. In the following, $s_{min} = 0.4$ and $d_{edit} = .6$ are used.

The perception control framework provides a selection method that compares agenda items with an user defined pattern. For the validation of the framework, the pattern selection method was

investigated in experiment B3 (see section 6.3.3) where the method was used for the preference of particular annotations. The experiment showed that the preference of the *LINE-MOD* annotation yields in the best *NSS* score results and that the preference of the *Color Ratio* annotation yields in the best pruning rate for the tested validation scenarios. Thus, the *LINE-MOD* annotation is preferred over all other annotations in the following. Additionally, the color annotation is preferred over all other annotations. It is expected, that the preference of these features enhances the *NSS* score and the pruning rate in some scenarios.

6.4 Control Experiments

In previous experiments, components of the perception control framework and their parametrization were investigated individually. Based on these experiments, default values of system parameters were defined and justified in section 6.3.4. In this section, the default parameters are used for a set of reference scenarios (see section 2.3) with varying tasks in order to investigate the hit rate for particular tasks in the reference scenarios.

6.4.1 Experiment C1 - The Default Scenario

This experiment validates the perception control framework for the default scene where multiple objects are visible and visually separated from each other (the experiment procedure is described in section 6.1.2).

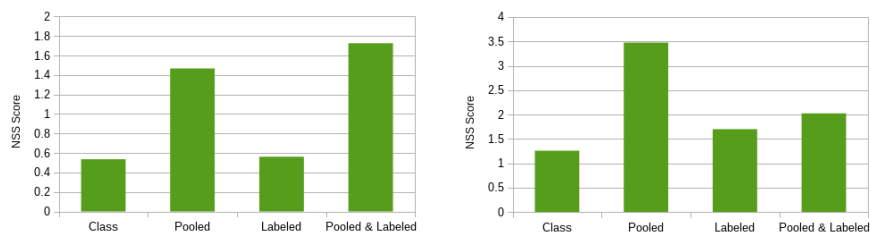


Figure 6.16 The *NSS* score for the ice cornflakes package (on the left) and the ice tea package (on the right).

Assumptions and Experimental Setup In this experiment, the *default* scenario is used (see figure 2.7) and the number of hits and misses is counted for visual search tasks with varying knowledge. Knowledge about the common appearance of object classes is used in all cases. Additionally, the experiment is repeated for tasks where a feature of the object is known in

advance. For the cornflakes package and the ice tea package, the expected *Goggles* annotation is specified in the task description. For both objects a label is clearly visible (i.e., “corn flakes” and “ice tea”) which is used as the expected outcome of the *Goggles* perception method. For the spatula, the dominant color (black) is defined in the task description. Finally, the experiment is repeated with additional knowledge about the spatial location of objects (i.e., spatial pooling of the kitchen table).

Hypothesis It is expected that the hit rate is better than a selection by chance and it is expected that the hit rate is enhanced for tasks where additional information about the task entity is known (e.g., the color of the entity or the location on the table).

Results For the cornflakes package, the prune rate is between 0.8 and 0.9 for all cases except for the case where the label of the package is known in which case the prune rate is slightly below 0.8 (see figure 6.16). Overall, best results were achieved for the pooling technique where the NSS score exceeds 2.5 (the score is below 2.0 without spatial pooling). The results for the ice tea package show that the prune rate lies between 0.7 and 0.9 where the best result (0.83) was achieved for the scenario where spatial pooling is used and where the label of the ice tea package is known in advance. The NSS score is about 0.6 without spatial pooling and about 1.6 with spatial pooling. Knowledge about the label of the package yields in a slight improvement for both cases. For the spatula, spatial pooling greatly enhances the hit rate when no color of the spatula is specified (above 0.7). The prune rate lies between 0.6 and 0.8 for the spatula where the best results were achieved when the spatial pooling was used.

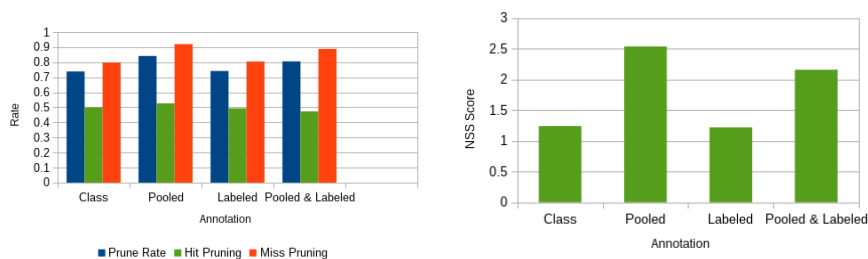


Figure 6.17 Overall results of experiment C1.

Discussion Overall, the NSS score is above 1.0 for each case which is a good result. It shows that the selection of control decisions is better than by chance. Thus, the methods that were used suit well for the scope of computational attention. The prune rate and the NSS score are generally higher for objects where the Bayes’ classifier can reliably predict the type of the object. The knowledge about annotations of objects improved the results for the spatula and the ice tea package but the result for the cornflakes package was worse for this case compared to the

selection based on the Bayes' classifier. The main reason for this is that the Bayes' classifier is able to reliably identify the cornflakes package based on the *LINE-MOD* annotation which is preferred over all other annotations when no spatial pooling is used.

6.5 Summary and Discussion

In this chapter, the proposed perception control framework was validated for a set of reference scenarios in the kitchen domain. The domain includes a small set of object classes (9) which are chosen so that the versatility of kitchen items is represented rudimentarily. For example, there are two objects with a plain black color (i.e., the spatula and the pancake maker). Furthermore, there are dark cups in the training data. This yields in more ambiguity for the color annotation of dark objects.

One of the most essential components of this framework is the Bayes' classifier that is used to predict the object type based on annotations of the object. It was validated separately in this chapter in terms of rejection rate and detection rate. The results showed that the Bayes' classifier can reliably detect the 9 objects of the validation domain (the detection rate was above 0.8 in the experiments). The set of features that is used by the Bayes' classifier is a system parameter for the perception control framework. The selection of features has influence on the attention mechanism because features usually perform good for a subset of the object classes only (e.g., the *Goggles* perception method works only for textured objects). Experiments showed that the *LINE-MOD* feature should be preferred over the other features so that the number of control decisions can be reduced. This is due to high confidence that is usually associated to the outcomes of the *LINE-MOD* perception method.

The most important selection method that is used in this framework combines the confidence value of the Bayes' classifier with a distance estimate based on an user defined weighting. The distance metric is used in order to estimate the similarity of task entities and recognized objects. It is important that the influence of the classification confidence and the distance estimate is balanced so that none of the aspects dominates the other. In this chapter, a good value for the weight was found based on a set of experiments where the parameter value was varied. This standard configuration of the control framework was used to investigate the influence of different types of knowledge on visual search tasks in the default scenario. The spatial pooling based on relations between instances always enhanced the pruning rate and the NSS score while additional knowledge about annotations of objects may yield in a worse result when the user specified feature does not represent the actual outcome of the perception method accurately.

Conclusion and Perspective

The objective of this thesis was the implementation of a perception control framework which is able to incorporate the outcomes of multiple knowledge based attention methods for the realization of a top-down attention control procedure in the context of mobile robotics. Furthermore, it was intended to use different types of knowledge about task relevant objects (i.e, the object class, attributes of the object and relations to other objects). For that purpose, the annotation phase of the perception procedure was segmented into atomic operations where each operation corresponds to the computation of an annotation for a particular recognized object. The perception control framework is responsible for the selection, configuration and execution of such operations. For the control framework, each of the possible operations represents a particular control decision in the problem solving process. The main objective of the control mechanism is the reduction of the number of control decisions which are required in order to fulfill visual search tasks.

Each visual search task defines one particular search target. In the most basic scenario, the task description only contains information about the type of the object (e.g., “Where is the cup” or “Where is the milk package?”). Knowledge based focus of attention requires knowledge about the common appearance of object classes in this case (e.g., the shape of cups). In the perception control framework, this knowledge is represented in a probabilistic model that is used by a Bayes’ classifier in order to predict the object class based on annotations of the object. The overall detection rate of the Bayes’ classifier is between 0.8 and 0.9 and the rejection rate is between 0.9 and 1.0 for completely specified objects depending on the classification threshold (see section 6.2). Thus, the predictions of the Bayes’ classifier can be used to reliably identify objects in the reference scenarios (see section 2.3). Nevertheless, the number of objects in the validation domain is rather low (only 9 different object classes) and it remains unclear how the Bayes’ classifier performs for a larger set of validation object classes. The performance of the classification based selection of control decisions highly depends on how well particular object classes can be predicted by the Bayes’ classifier. It was shown that the perception control framework is able to prune away more than 80% of all possible control decisions for objects, such

as the cornflakes package, where the classifier prediction is reliable (see figure 6.10). Furthermore, it was shown that the selection of control decisions was better than by chance for objects, such as the spatula, where the prediction of the classifier is less reliable (around 70% of all decisions were pruned in the experiment).

It was shown that the perception control framework can utilize knowledge about expected annotations of task targets (e.g., “Where is the yellow cup” or “Where is the boxy milk package?”) in order to reduce the number of required control decisions for particular scenarios (see section 6.3.1 and 6.3.2). For that purpose, an edit distance metric between model instances was defined. Basically, the edit distance measures the distance between features of annotations which are defined by the task entity. The distance between features is computed using common distance metrics such as the Levenshtein distance [Lev66] for string features. The edit distance estimate is combined with the classification confidence in order to yield a “task similarity” value that includes knowledge about the common appearance of objects and knowledge about expected annotations of task entities. In section 6.3.1 it was shown that the knowledge about annotations of the target object can yield in a better pruning rate and NSS score for particular configurations and scenarios. For the cornflakes package, the pruning rate was improved from about 80% to about 90% when knowledge about the label on the cornflakes package was used in the control procedure (see figure 6.10).

Another type of knowledge that can be utilized by the perception control framework in order to reduce the number of required control decisions for visual search tasks is knowledge about relations between objects. In the proposed framework, this knowledge can also be represented in task descriptions for visual search tasks (e.g., “Is there a milk package on the table?” or “Is there a yellow cup next to the dish on the kitchen table?”). Nevertheless, special methods are required in order to dynamically infer such relations between visible objects. The *RoboSherlock* perception framework provides a method to infer the semantic location (e.g., on top of the kitchen table) of objects based on the perceived depth and a bounding box that corresponds to the attended location. It is not possible to infer more complex spatial relations with this method (e.g., “Is the cup right of the dish?”). In the scope of this thesis, knowledge about spatial relations between objects is restricted to the spatial pooling technique that is provided by *RoboSherlock*. Additionally, for the validation of the control framework, the kitchen table was segmented into border area and center area in order to allow visual search tasks such as “Is there a cornflakes package at the border of the kitchen table?”. The knowledge about relations between objects is represented by a special selection method that computes the semantic distance between recognized objects and task entities based on the hierarchy of relations between both objects. The spatial pooling scenario is trivial for this procedure because recognized objects are directly connected with task entities in this case (i.e., the distance of the path between recognized objects and task entities is equal to the transition cost of the spatial relation between them).

Perspective In the scope of this thesis, the perception control framework was investigated with static recordings of scenes in the reference kitchen. Dynamic scenes and scenes where the robot navigates were not considered in this thesis because dynamic environments require additional functionality such as the retraction of control decisions based on a backtracking algorithm (e.g., a truth maintenance system). Nevertheless, there are interesting scenarios for visual search which involve navigation of the robot or moving objects such as the guidance of the gaze direction of the robot. For example, manipulation of the gaze direction could be used in order to gather sensory data from a slightly different perspective when the classification confidence is slightly below the classification threshold for the selected object.

In the current implementation, an edit distance metric between instances is used in order to select control decisions based on knowledge about annotations of the target object. The major disadvantage of this method is that the user has to guess the outcomes of perception methods. It is impossible to specify these values exactly in advance because the outcome of perception methods varies for different frames with the same visible objects. The method works fine as long as the edit distance between the user specified feature value and the actual feature value is small but it does not represent the statistics of features accurately. Thus, it is desirable to utilize a probabilistic approach instead.

Furthermore, it would be interesting to investigate regression methods for the prediction of feature values. The execution of perception methods can be omitted if the confidence of the predicted feature value exceeds an user specified threshold.

The implemented control procedure acts as a framework where multiple attention methods can be combined. In the current implementation, a set of attention inspired methods is used in order to represent different aspects of knowledge about the current task. It would be interesting to investigate the control framework for different attention methods such as standard bottom-up models. The proposed perception control framework can be utilized to investigate the performance of different combinations of attention methods as well it can be utilized to compare different attention methods with each other.

Appendix

A.1 List of Figures

2.1	Visual search efficiency.	6
2.2	Labeling results for a household scene.	7
2.3	PR2 looking at household items.	12
2.4	PR2 Field of View.	13
2.5	Hierarchical behavior decomposition.	16
2.6	The reference kitchen	17
2.7	The reference scenes	18
3.1	Basic top-down biasing architecture.	24
3.2	Holistic scene characteristics using low-level features.	26
3.3	The stages of attention in the <i>Guided Search Model (GS)</i>	28
3.4	Rensink’s triadic architecture of attention.	29
3.5	Task graph and task relevancy estimation.	35
4.1	A Kitchen Domain Type Hierarchy.	54
4.2	Relations that are used in the kitchen domain ontology.	55
4.3	The Item Model Concept.	56
4.4	Task Relations.	61
4.5	Strategy activation rule.	63
4.6	Sequence of selection criteria.	67
4.7	Relational Task Relevance.	70
5.1	Architecture of the Perception Control Framework.	82
5.2	Bundle Management GUI.	84
5.3	ROS Messages GUI.	85
5.4	CAS Visualization User Interface.	86

5.5	Generic Configuration Editor.	88
5.6	Console GUI.	89
5.7	Drools Rules Class Hierarchy.	90
5.8	Rules GUI.	91
5.9	Control Core Class Hierarchy.	92
5.10	Acquisition of Focusing Knowledge.	93
5.11	Acquisition of Ordering Knowledge.	93
5.12	Acquisition of Processing Knowledge.	94
5.13	Core Agenda Processing Class Hierarchy.	95
5.14	Strategy Monitoring.	95
5.15	Agenda Monitoring.	96
6.1	Corn Flakes Training Samples	108
6.2	Sample results of experiment A1	109
6.3	ROC curve of Experiment A1	110
6.4	Results of Experiment A2	111
6.5	Comparison of experiments A1 and A2	112
6.6	Results of Experiment A3	113
6.7	Comparison of Experiments A1, A2 and A3	114
6.8	Detection and Rejection Rate of Experiment A4.	115
6.9	ROC Curve for the Bayes' Classifier.	116
6.10	Pruning Results of Experiment A1.	117
6.11	NSS Score Results of Experiment A1.	118
6.12	Pruning Rate Results of Experiment B2	119
6.13	NSS Score Results of Experiment B2	120
6.14	Pruning Results of Experiment B3.	121
6.15	Overall Results of Experiment B3.	122
6.16	NSS Score Results of Experiment C1.	124
6.17	Overall results of Experiment C1.	125
B.1	Results of Experiment A1 for the Corn Flakes Package.	139
B.2	Results of Experiment A1 for the Cups.	139
B.3	Results of Experiment A1 for the Ice Tea Package.	140
B.4	Results of Experiment A1 for the Milk Package.	140
B.5	Results of Experiment A1 for the Pancake Maker.	140
B.6	Results of Experiment A1 for the Pancake Tube.	141
B.7	Results of Experiment A1 for the Spatula.	141
B.8	Overall Results of Experiment A1.	141
B.9	Results of Experiment A2 for the Corn Flakes Package.	142

B.10 Results of Experiment A2 for the Cups.	142
B.11 Results of Experiment A2 for the Ice Tea Package.	143
B.12 Results of Experiment A2 for the Milk Package.	143
B.13 Results of Experiment A2 for the Pancake Maker.	143
B.14 Results of Experiment A2 for the Pancake Tube.	144
B.15 Results of Experiment A2 for the Spatula.	144
B.16 Overall Results of Experiment A2.	144
B.17 Results of Experiment A3 for the Corn Flakes Package.	145
B.18 Results of Experiment A3 for the Cups.	145
B.19 Results of Experiment A3 for the Ice Tea Package.	146
B.20 Results of Experiment A3 for the Milk Package.	146
B.21 Results of Experiment A3 for the Pancake Maker.	146
B.22 Results of Experiment A3 for the Pancake Tube.	147
B.23 Results of Experiment A3 for the Spatula.	147
B.24 Overall Results of Experiment A3.	147
B.25 Results of Experiment A4 for the Corn Flakes Package.	148
B.26 Results of Experiment A4 for the Cups.	148
B.27 Results of Experiment A4 for the Ice Tea Package.	149
B.28 Results of Experiment A4 for the Milk Package.	149
B.29 Results of Experiment A4 for the Pancake Maker.	149
B.30 Results of Experiment A4 for the Pancake Tube.	150
B.31 Results of Experiment A4 for the Spatula.	150
B.32 Overall Results of Experiment A4.	150
B.33 Results of Experiment B1 for the Corn Flakes Package.	151
B.34 Results of Experiment B1 for a Labeled Corn Flakes Package.	151
B.35 Results of Experiment B1 for the Spatula.	151
B.36 Results of Experiment B1 for a Occluded Corn Flakes Package.	152
B.37 Results of Experiment B2	152
B.38 Results of Experiment B2	152
B.39 Results of Experiment B2	153
B.40 Results of Experiment B2	153
B.41 Results of Experiment B2	153
B.42 Results of Experiment B2	154
B.43 Results of Experiment B3 for the Corn Flakes Package.	154
B.44 Results of Experiment B3 for the Ice Tea Package.	155
B.45 Results of Experiment B3 for the Spatula.	155
B.46 Overall results of Experiment B3.	156
B.47 Results of Experiment C1 for the Cornflakes Package.	157

B.48 Results of Experiment C1 for the Ice Tea Package.	157
B.49 Results of Experiment C1 for the Spatula.	158
B.50 Overall results of Experiment C1.	158

A.2 Bibliography

- [BAA11] A. Borji, M. N. Ahmadabadi, and B. N. Araabi. “Cost-sensitive learning of top-down modulation for attentional control.” In: *Mach. Vis. Appl.* 22.1 (2011), pp. 61–76.
- [BAK] J. Burianek, A. Ahmadyfard, and J. Kittle. *Soil-47 the surrey object image library*. <http://www.ee.surrey.ac.uk/Research/VSSP/demos/colour/soil47/>. Accessed: 2014-02-19.
- [CG91] R. Cunis and A. Günter. “PLAKON - Übersicht Über Das System”. In: *Das PLAKON-Buch, Ein Expertensystemkern Für Planungs- Und Konfigurierungsaufgaben in Technischen Domänen*. London, UK, UK: Springer-Verlag, 1991, pp. 37–57. ISBN: 3-540-53683-3.
- [Cla05] R. N. Clark. “Notes on the Resolution and Other Details of the Human Eye”. <http://clarkvision.com/articles/eye-resolution.html>. Accessed: 2014-02-19. 2005.
- [CS02] M. Corbetta and G. L. Shulman. “Control of Goal-Directed and stimulus driven attention in the brain”. In: *Nature Reviews Neuroscience* 3 (3) (2002), pp. 201–215.
- [CZ07] O. Chum and A. Zisserman. “An Exemplar Model for Learning Object Classes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2007.
- [DD95] R. Desimone and J. Duncan. “Neural Mechanisms of Selective Visual Attention”. In: *Annual Review of Neuroscience* 18.1 (1995), pp. 193–222.
- [Dha03] L. I. N. Dhavale. “Realistic avatar eye and head animation using a neurobiological model of visual attention”. In: *Proc. SPIE*. SPIE Press, 2003, pp. 64–78.
- [EI10] L. Elazary and L. Itti. “A Bayesian model for efficient visual search and recognition”. In: *Vision Research* 50.14 (June 2010), pp. 1338–1352.
- [Eve+] M. Everingham, A. Zisserman, C. Williams, and L. Van Gool. “The Pascal Visual Object Classes Challenge 2006 (VOC2006) Results”. In: (). Accessed: 2014-02-19.
- [Flo62] R. W. Floyd. “Algorithm 97: Shortest Path”. In: *Commun. ACM* 5.6 (June 1962), pp. 345–.
- [FRC10] S. Frintrop, E. Rome, and H. I. Christensen. “Computational Visual Attention Systems and Their Cognitive Foundations: A Survey”. In: *ACM Trans. Appl. Percept.* 7.1 (Jan. 2010), 6:1–6:39.
- [Fri06] S. Frintrop. *VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search*. Vol. 3899. Lecture Notes in Computer Science. Springer, 2006.

- [GBS05] J.-M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. “The Amsterdam Library of Object Images”. In: *Int. J. Comput. Vision* 61.1 (Jan. 2005), pp. 103–112.
- [GHV09] D. Gao, S. Han, and N. Vasconcelos. “Discriminant Saliency, the Detection of Suspicious Coincidences, and Applications to Visual Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.6 (2009), pp. 989–1005.
- [Gün92] A. Günter. *Flexible Kontrolle in Expertensystemen zur Planung und Konfigurierung in technischen Domänen*. Dissertationen zur künstlichen Intelligenz. Infix, 1992. ISBN: 9783929037036.
- [GV04] D. Gao and N. Vasconcelos. “Discriminant saliency for visual recognition from cluttered scenes”. In: *In Proc. NIPS*. 2004, pp. 481–488.
- [Hay85] B. Hayes-Roth. “A Blackboard Architecture for Control”. In: *Artif. Intell.* 26.3 (Aug. 1985), pp. 251–321. ISSN: 0004-3702.
- [Hin+13] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. “Model Based Training, Detection and Pose Estimation of Texture-less 3D Objects in Heavily Cluttered Scenes”. In: *Proceedings of the 11th Asian Conference on Computer Vision - Volume Part I. ACCV’12*. Daejeon, Korea: Springer-Verlag, 2013, pp. 548–562. ISBN: 978-3-642-37330-5.
- [HKP07] J. Harel, C. Koch, and P. Perona. “Graph-based visual saliency”. In: *Advances in Neural Information Processing Systems 19*. MIT Press, 2007, pp. 545–552.
- [IB09] L. Itti and P. F. Baldi. “Bayesian Surprise Attracts Human Attention”. In: *Vision Research* 49.10 (May 2009), pp. 1295–1306.
- [IK01] L. Itti and C. Koch. “Feature Combination Strategies for Saliency-Based Visual Attention Systems”. In: *Journal of Electronic Imaging* 10.1 (Jan. 2001), pp. 161–169.
- [IKN98] L. Itti, C. Koch, and E. Niebur. *A Model of Saliency-based Visual Attention for Rapid Scene Analysis*. 1998.
- [JL95] G. H. John and P. Langley. “Estimating Continuous Distributions in Bayesian Classifiers”. In: *Eleventh Conference on Uncertainty in Artificial Intelligence*. San Mateo: Morgan Kaufmann, 1995, pp. 338–345.
- [JT05] F. Jurie and B. Triggs. “Creating efficient codebooks for visual recognition”. In: *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1*. IEEE, 2005, pp. 604–610.
- [Lan14] C. Landgraf. *Extending The KnowRob Upper Ontology With Respect To Qualitative Spatial Knowledge Representation*. 2014.
- [Lev66] V. Levenshtein. “Binary Codes Capable of Correcting Deletions, Insertions and Reversals”. In: *Soviet Physics Doklady* 10 (1966), p. 707.
- [Lia+06] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. “Comprehensive learning particle swarm optimizer for global optimization of multimodal functions.” In: *IEEE Trans. Evolutionary Computation* 10.3 (2006), pp. 281–295.

- [Low99] D. G. Lowe. “Object Recognition from Local Scale-Invariant Features”. In: *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*. ICCV '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1150–.
- [MNE00] A. Maki, P. Nordlund, and J. Eklundh. “Attentional Scene Segmentation Integrating Depth and Motion”. In: *Computer Vision and Image Understanding* 78 (June 2000), pp. 351–373.
- [NI05] V. Navalpakkam and L. Itti. “Modeling the influence of task on attention”. In: *Vision Research* 45.2 (Jan. 2005), pp. 205–231.
- [NI06] V. Navalpakkam and L. Itti. “An Integrated Model of Top-Down and Bottom-Up Attention for Optimizing Detection Speed”. In: *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*. CVPR '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 2049–2056.
- [NNM96] S. A. Nene, S. K. Nayar, and H. Murase. *Columbia Object Image Library (COIL-20)*. Tech. rep. CUCS-005-96. Department of Computer Science, Columbia University, Feb. 1996.
- [Pal99] S. E. Palmer. *Vision science : photons to phenomenology*. MIT Press, 1999, p. 810.
- [Pet+05] R. J. Peters, A. Iyer, L. Itti, and C. Koch. *Components of bottom-up gaze allocation in natural images*. 2005.
- [PI07] R. J. Peters and L. Itti. “Beyond bottom-up: Incorporating task-dependent influences into a computational model of spatial attention”. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition. Minneapolis, MN, 2007.
- [Rao+02] R. P. Rao, G. J. Zelinsky, M. M. Hayhoe, and D. H. Ballard. “Eye movements in iconic visual search”. In: *Vision Research* 42 (2002), pp. 1447–1463.
- [Ren00] R. A. Rensink. “The dynamic representation of scenes”. In: *Visual Cognition* 7.1-3 (Jan. 2000), pp. 17–42.
- [RM04] L. W. Renninger and J. Malik. “When is scene identification just texture recognition?” In: *Vision Research* 44.19 (Sept. 2004), pp. 2301–2311.
- [SC99] D. J. Simons and C. F. Chabris. “Gorillas in our midst: Sustained inattentive blindness for dynamic events”. In: *Perception* 28 (1999), pp. 1059–1074.
- [SI07] C. Siagian and L. Itti. “Rapid Biologically-Inspired Scene Classification Using Features Shared with Visual Attention”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.2 (2007), pp. 300–312.
- [Siv+05] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. “Discovering Objects and their Localization in Images.” In: *ICCV*. IEEE Computer Society, 2005, pp. 370–377.
- [SWP05] T. Serre, L. Wolf, and T. Poggio. “Object recognition with features inspired by visual cortex”. In: *In CVPR*. 2005, pp. 994–1000.
- [TBG05] B. W. Tatler, R. J. Baddeley, and I. D. Gilchrist. “Visual correlates of fixation selection: effects of scale and time”. In: *Vision Research* 45.5 (2005), pp. 643–659.

- [TG80] A. M. Treisman and G. Gelade. “A Feature-Integration Theory of Attention.” In: *Cognitive Psychology* 12 (1980), pp. 97–136.
- [Tor03] A. Torralba. “Modeling global scene factors in attention”. In: *Journal of Optical Society of America* 20(7) (2003), pp. 1407–1418.
- [Tso89] J. K. Tsotsos. “The Complexity of Perceptual Search Tasks”. In: *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 2. IJCAI’89*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 1571–1577.
- [WCF89] J. M. Wolfe, K. R. Cave, and S. L. Franzel. “Guided search: An alternative to the feature integration model for visual search.” In: *Journal of Experimental Psychology: Human Perception & Performance* 15(3) (1989), pp. 419–433.
- [WK06] D. Walther and C. Koch. “Modeling attention to salient proto-objects”. In: *Neural Networks* 19.9 (Nov. 2006), pp. 1395–1407.
- [Wol94] J. M. Wolfe. “Guided Search 2.0- a revised model of visual search”. In: *Psychonomic Bulletin and Review* 1(2) (1994), pp. 202–238.
- [Yan+13] V. Yanulevskaya, J. Uijlings, J. Geusebroek, and N. Sebe. “A proto-object-based computational model for visual saliency”. In: *Journal of Vision* (2013).
- [Yar67] A. L. Yarbus. “Eye movements during perception of complex objects”. In: *Eye Movements and Vision*. Plenum Press, 1967. Chap. VII, pp. 171–196.
- [Zha+08] L. Zhang, M. H. Tong, T. K. Marks, H. Shan, and G. W. Cottrell. “SUN: A Bayesian framework for saliency using natural statistics”. In: *Journal of Vision* 8.7 (2008).

A.3 List of Abbreviations

AUC Area under the ROC curve, p. 24, 25, 101

CAS Common Analysis System, p. 78, 85, 86

DSL Domain Specific Language, p. 91

GD Gaussian Distribution, p. 41–43, 45, 50

GGD Generalized Gaussian Distribution, p. 43–45

GUI Graphical User Interface, p. 78, 82

MGD Multivariate Gaussian Distribution, p. 46, 50

NSS Normalized Scanpath Saliency, p. 24, 25, 99, 102, 103, 106, 116, 118–126, 128

OSGi Open Service Gateway initiative, p. 81, 83, 87–89, 94

PDF Probability Density Function, p. 37, 41, 42, 46

PR2 Personal Robot 2, p. 7, 8, 10, 13, 14, 17–20, 97, 101

ROC Receiver Operating Characteristics, p. 24, 101, 110, 116

ROI Region of Interest, p. 6, 15, 19, 103

ROS Robot Operating System, p. 85, 87, 88, 159

SIFT Scale-Invariant Feature Transform, p. 30, 37, 41, 42

UIM Unstructured Information Management, p. 14, 15

UIMA Unstructured Information Management Architecture, p. 51, 78, 81, 84, 85

Empirical Results

B.1 Experiment A1

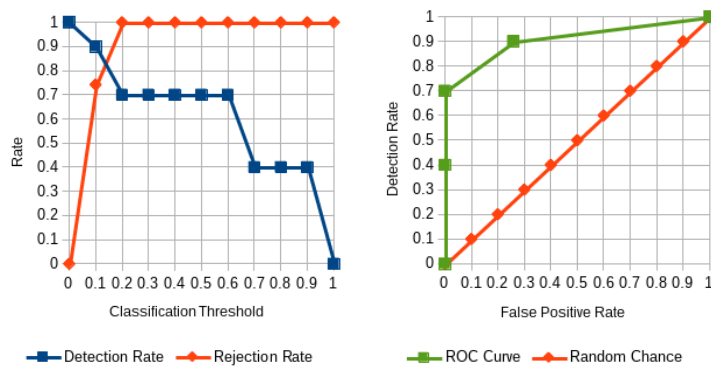


Figure B.1 Classification results of the *Goggles* feature for the corn flakes package.

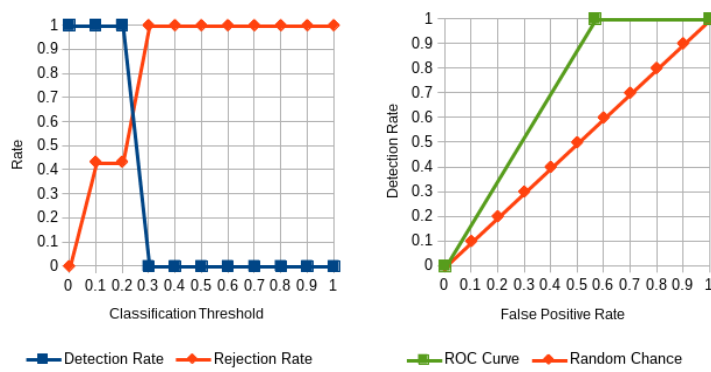


Figure B.2 Classification results of the *Goggles* feature for the cups.

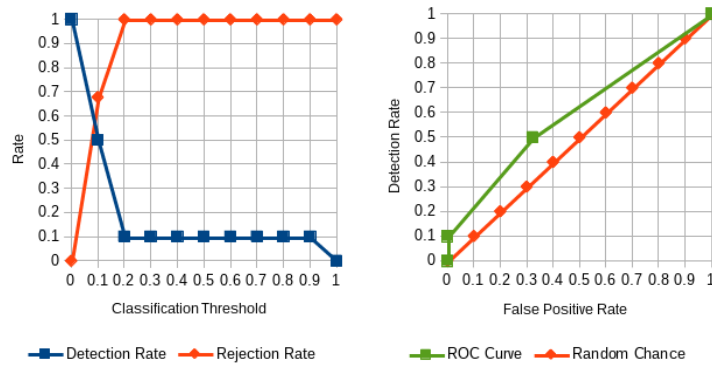


Figure B.3 Classification results of the *Goggles* feature for the ice tea package.

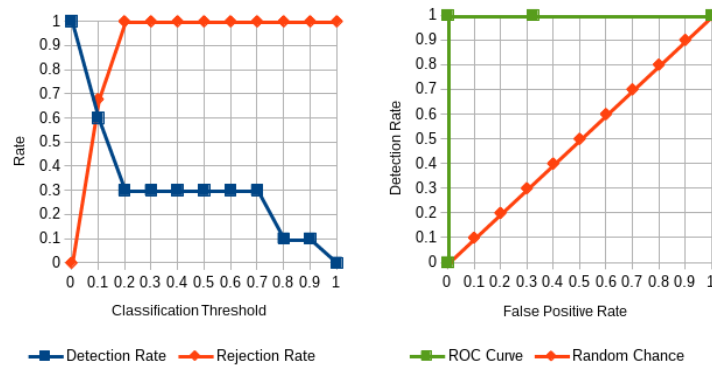


Figure B.4 Classification results of the *Goggles* feature for the milk package.

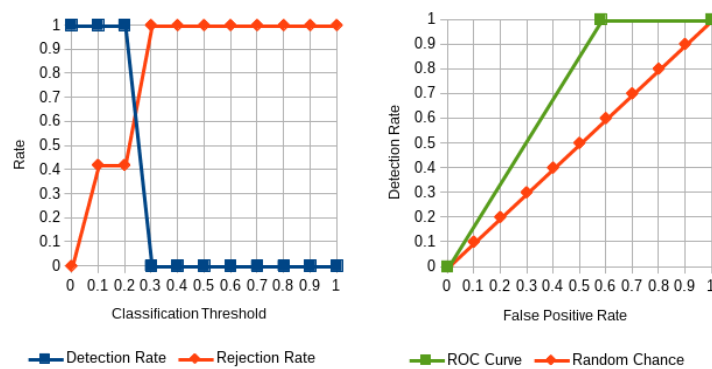


Figure B.5 Classification results of the *Goggles* feature for the pancake maker.

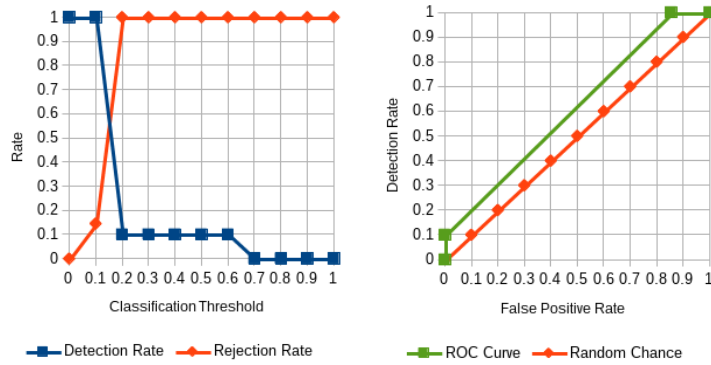


Figure B.6 Classification results of the *Goggles* feature for the pancake tube.

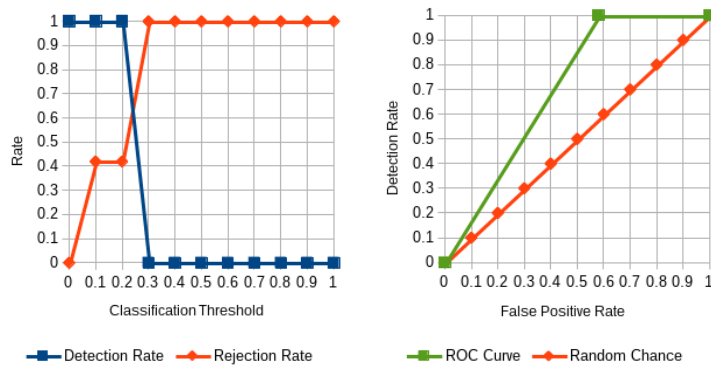


Figure B.7 Classification results of the *Goggles* feature for the spatula.

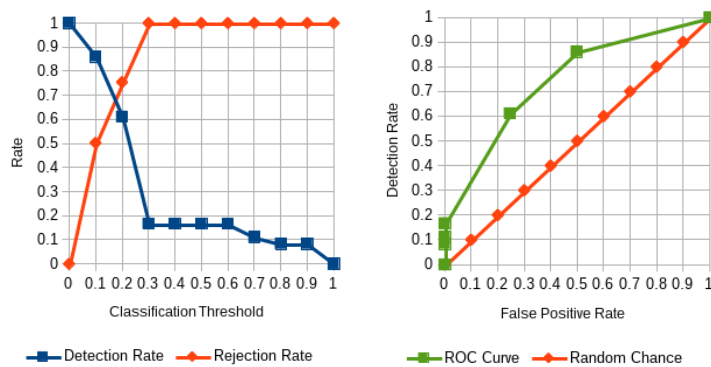


Figure B.8 Overall classification results of the *Goggles* feature.

B.2 Experiment A2

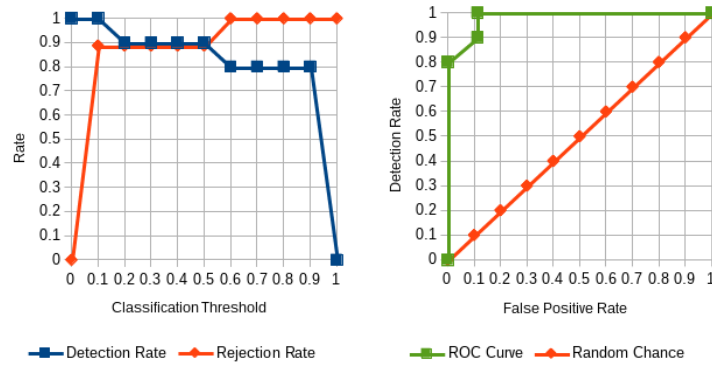


Figure B.9 Classification results of the *LINE-MOD* feature for the corn flakes package.

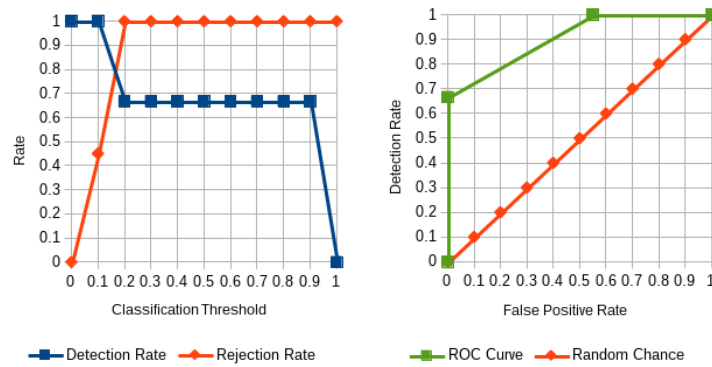


Figure B.10 Classification results of the *LINE-MOD* feature for the cups.

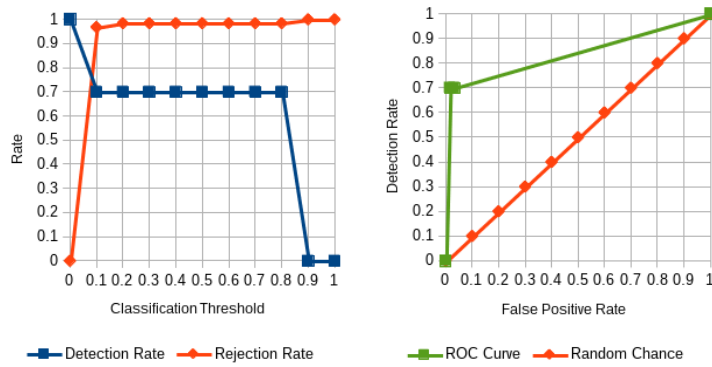


Figure B.11 Classification results of the *LINE-MOD* feature for the ice tea package.

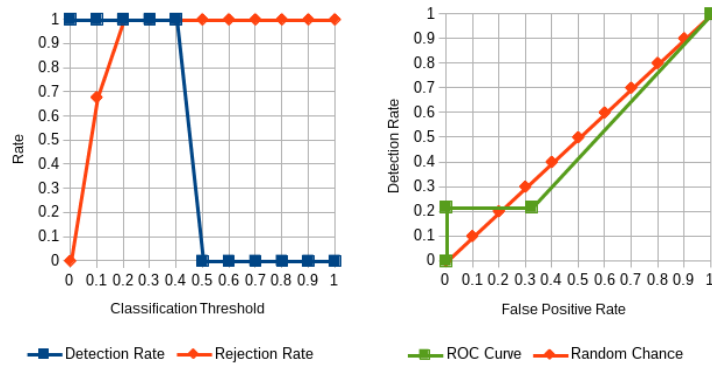


Figure B.12 Classification results of the *LINE-MOD* feature for the milk package.

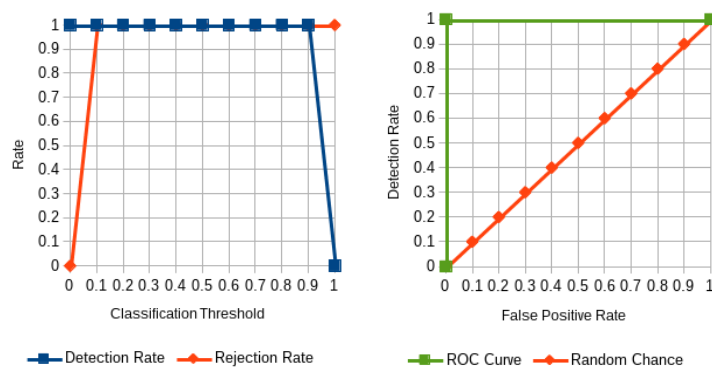


Figure B.13 Classification results of the *LINE-MOD* feature for the pancake maker.

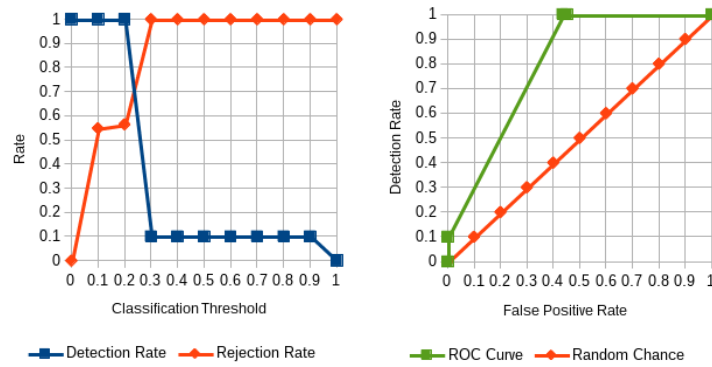


Figure B.14 Classification results of the *LINE-MOD* feature for the pancake tube.

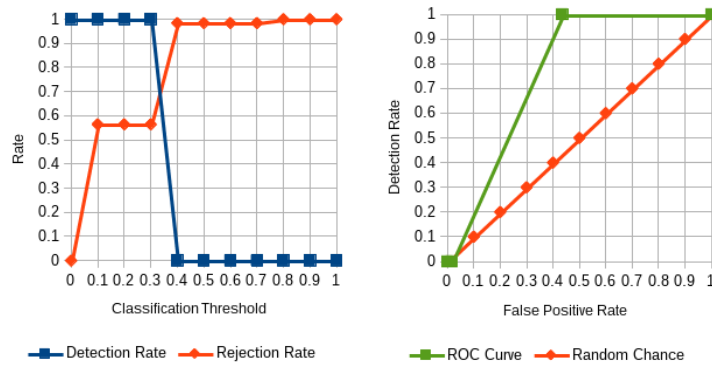


Figure B.15 Classification results of the *LINE-MOD* feature for the spatula.

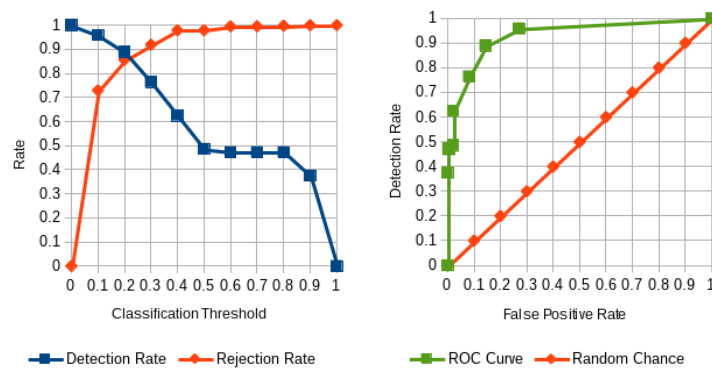


Figure B.16 Overall classification results of the *LINE-MOD* feature.

B.3 Experiment A3

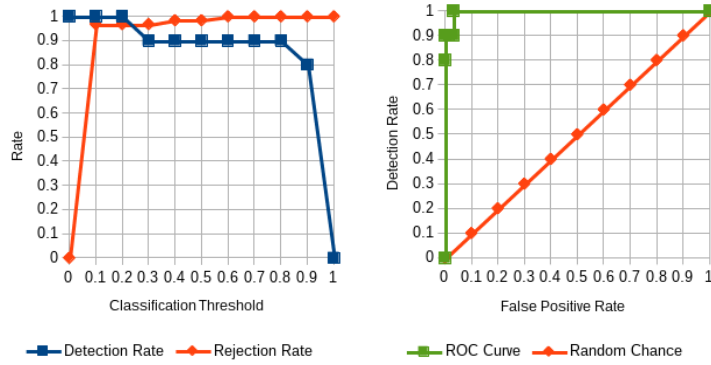


Figure B.17 Classification results of the *Color Ratio* feature for the corn flakes package.

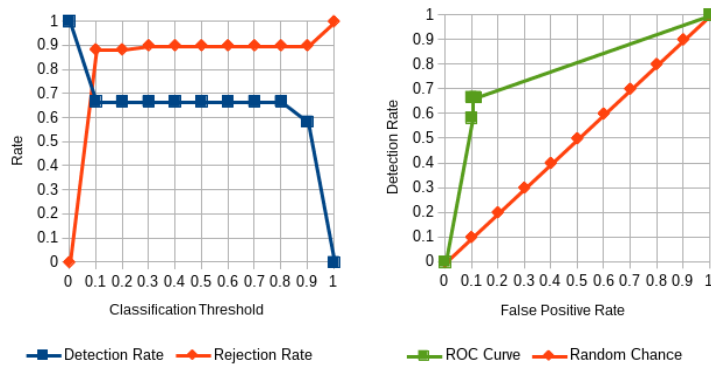


Figure B.18 Classification results of the *Color Ratio* feature for the cups.

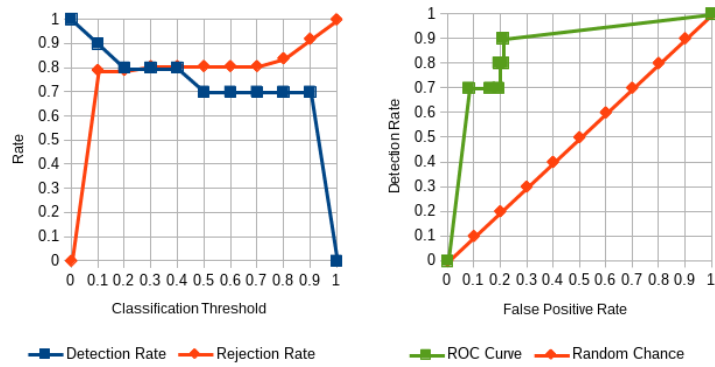


Figure B.19 Classification results of the *Color Ratio* feature for the ice tea package.

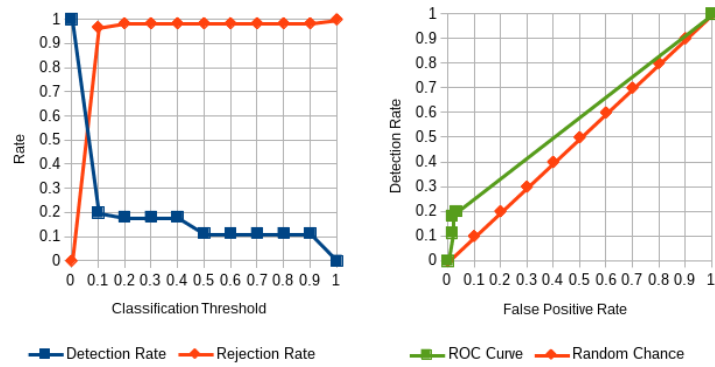


Figure B.20 Classification results of the *Color Ratio* feature for the milk package.

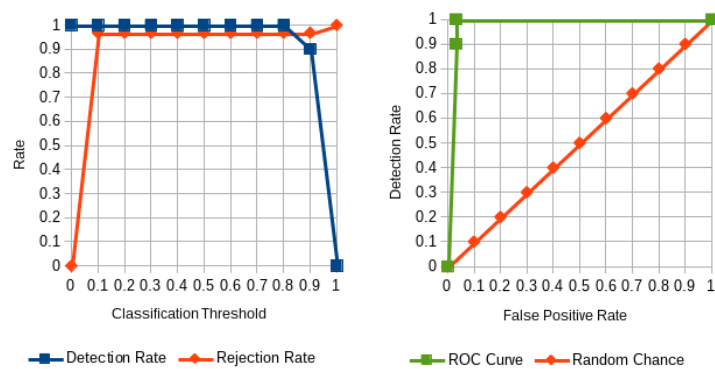


Figure B.21 Classification results of the *Color Ratio* feature for the pancake maker.

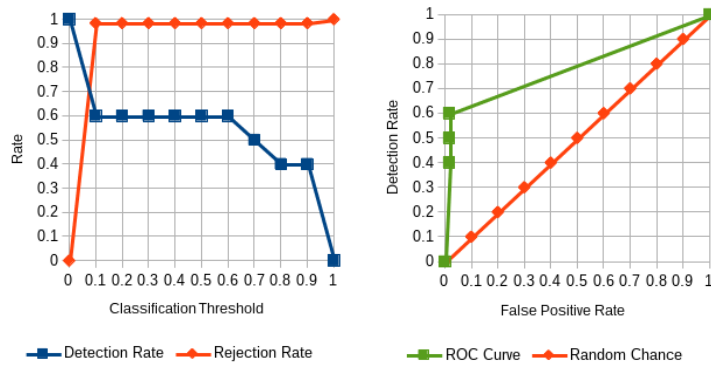


Figure B.22 Classification results of the *Color Ratio* feature for the pancake tube.

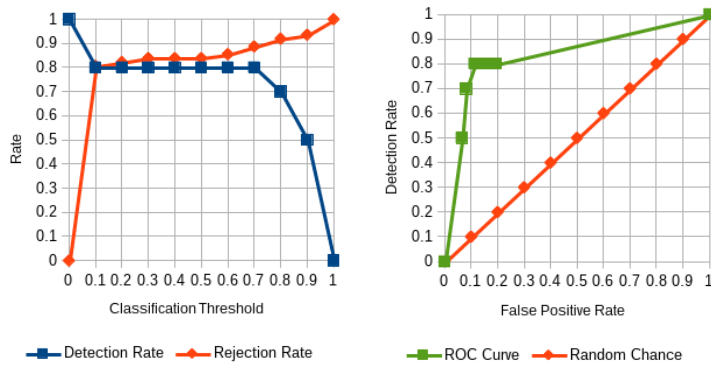


Figure B.23 Classification results of the *Color Ratio* feature for the spatula.

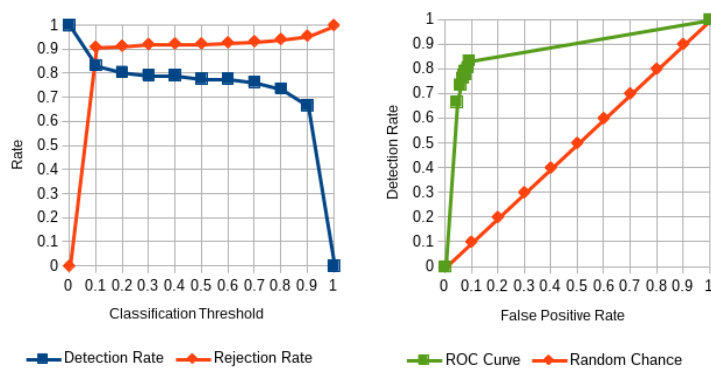


Figure B.24 Overall classification results of the *Color Ratio* feature.

B.4 Experiment A4

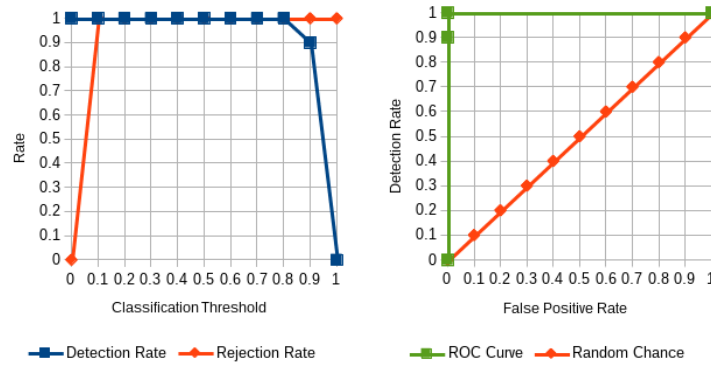


Figure B.25 Classification results of the Bayes' classifier for the corn flakes package.

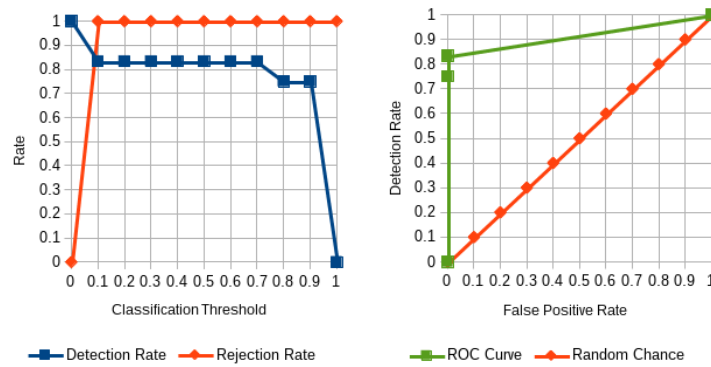


Figure B.26 Classification results of the Bayes' classifier for the cups.

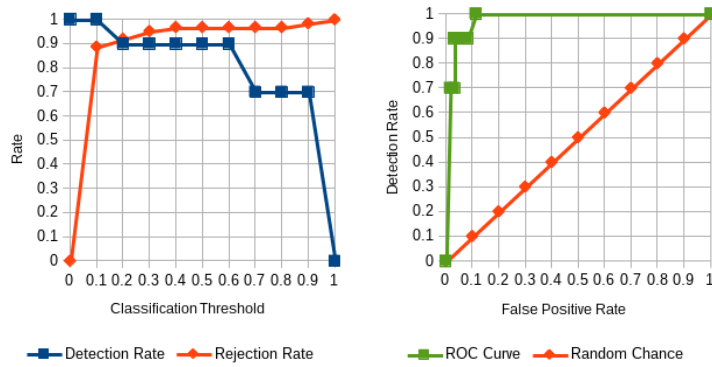


Figure B.27 Classification results of the Bayes' classifier for the ice tea package.

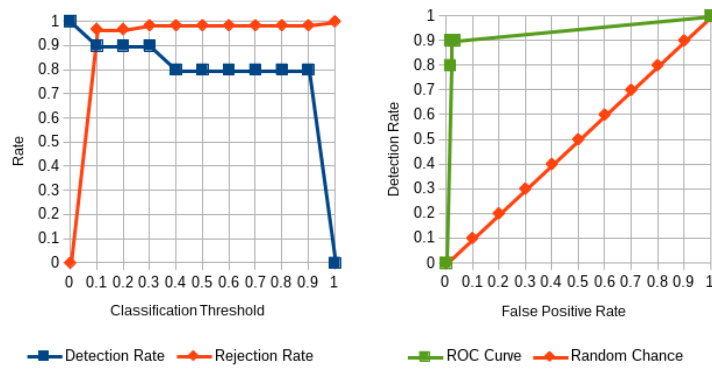


Figure B.28 Classification results of the Bayes' classifier for the milk package.

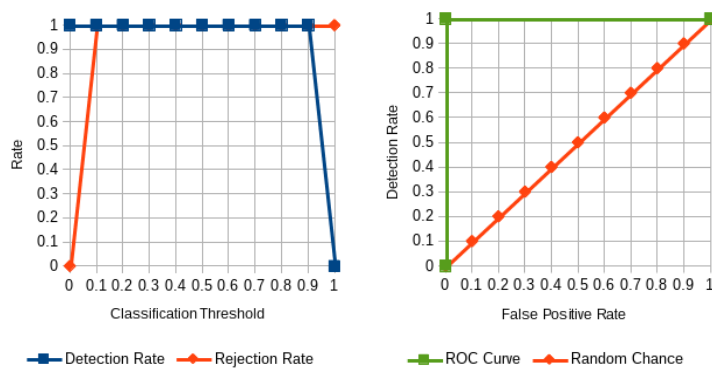


Figure B.29 Classification results of the Bayes' classifier for the pancake maker.

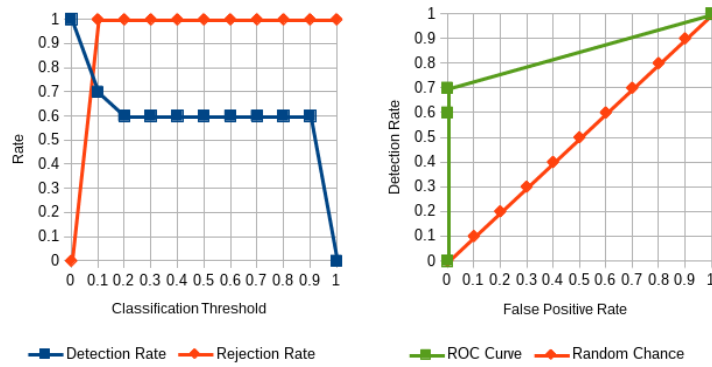


Figure B.30 Classification results of the Bayes' classifier for the pancake tube.

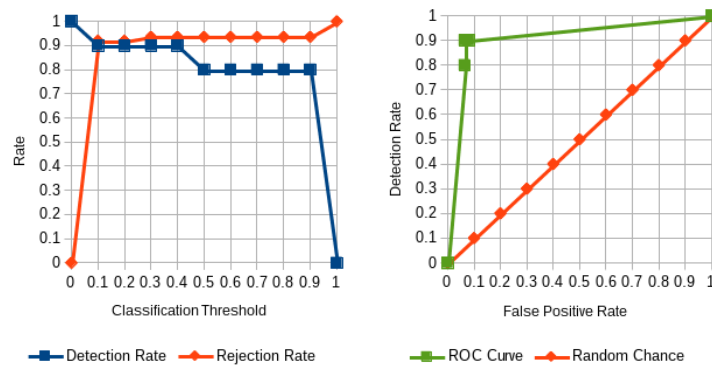


Figure B.31 Classification results of the Bayes' classifier for the spatula.

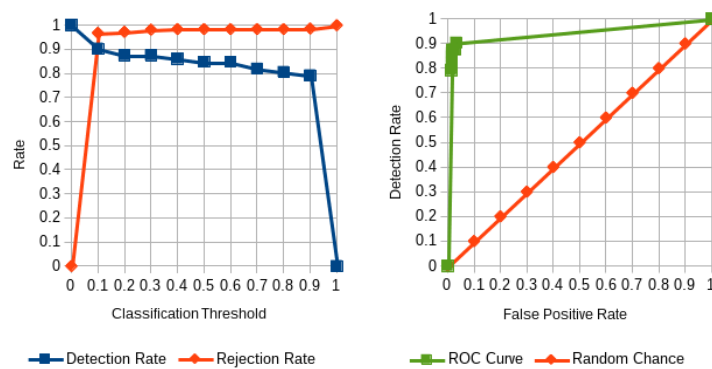


Figure B.32 Overall classification results of the Bayes' classifier.

B.5 Experiment B1

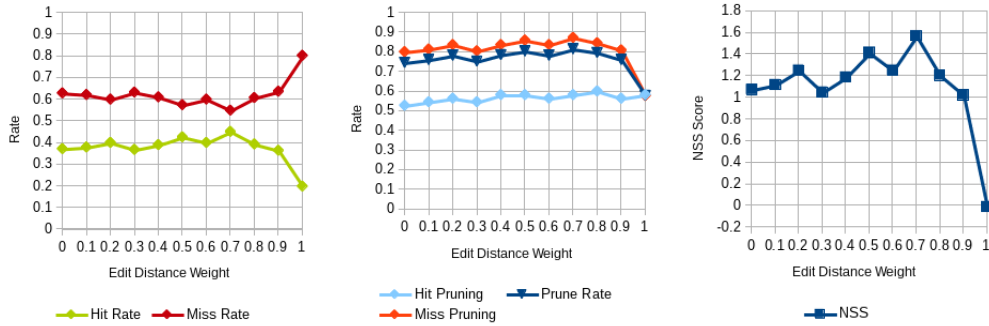


Figure B.33 Results for the edit weight parameter experiment for the corn flakes package in the default scene.

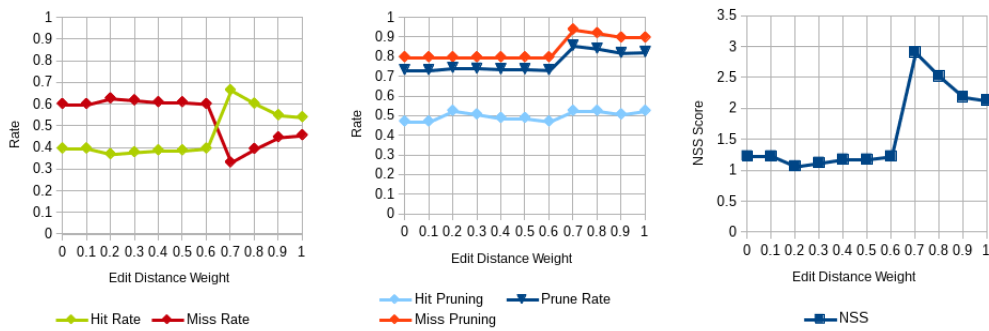


Figure B.34 Results for the edit weight parameter experiment for the corn flakes package in the default scene when the label of the package is known.

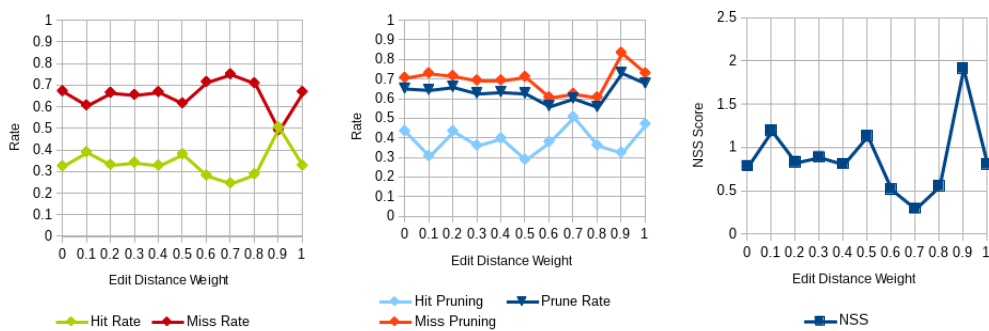


Figure B.35 Results for the edit weight parameter experiment for the spatula in the default scene when the label of the package is known.

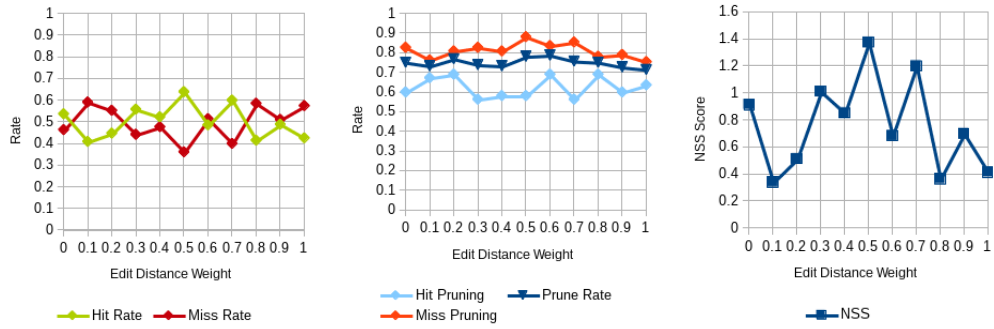


Figure B.36 Results for the edit weight parameter experiment for the corn flakes package in the occlusion scene.

B.6 Experiment B2

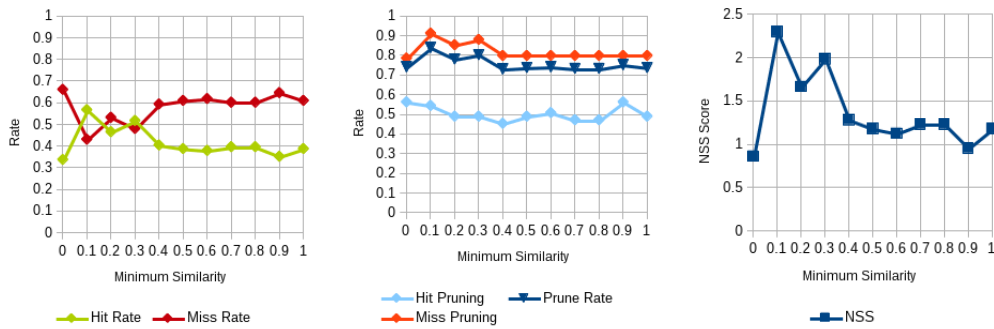


Figure B.37 The results of experiment B2 where the target is the cornflakes package and the edit distance weight is set to 0.4.

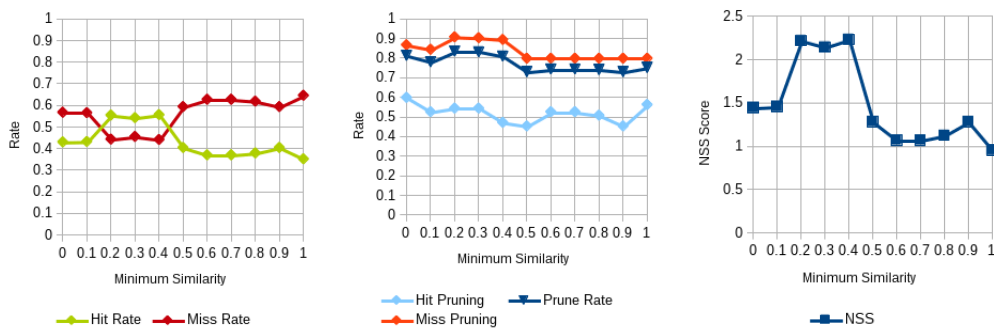


Figure B.38 The results of experiment B2 where the target is the cornflakes package and the edit distance weight is set to 0.5.

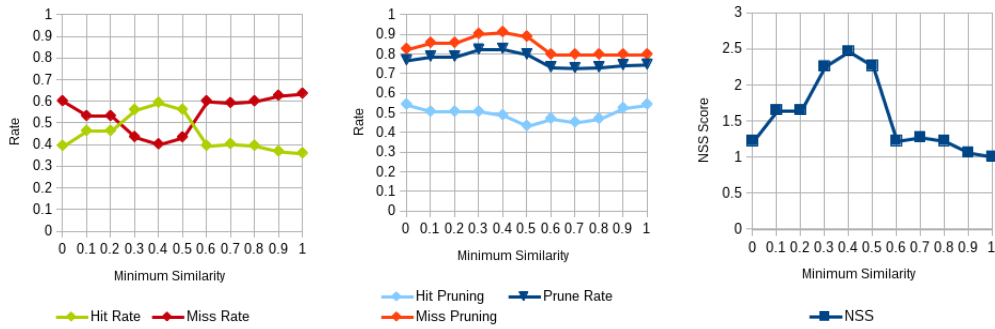


Figure B.39 The results of experiment B2 where the target is the cornflakes package and the edit distance weight is set to 0.6.

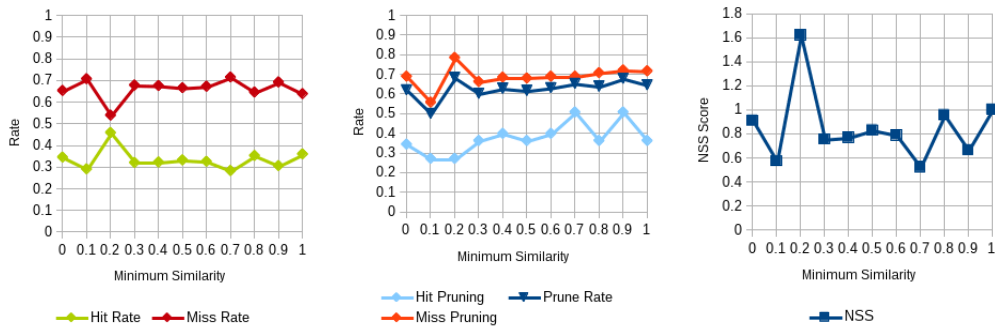


Figure B.40 The results of experiment B2 where the target is the spatula and the edit distance weight is set to 0.4.

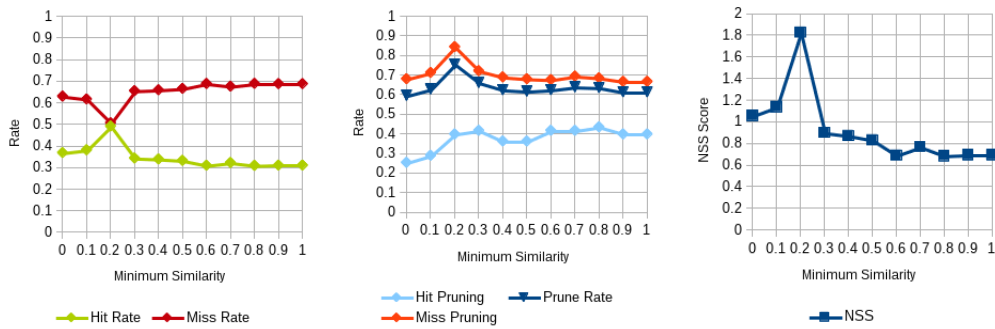


Figure B.41 The results of experiment B2 where the target is the spatula and the edit distance weight is set to 0.5.

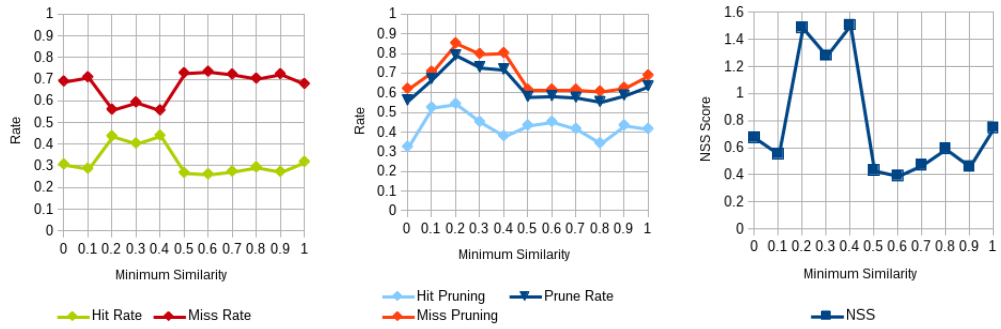


Figure B.42 The results of experiment B2 where the target is the spatula and the edit distance weight is set to 0.6.

B.7 Experiment B3

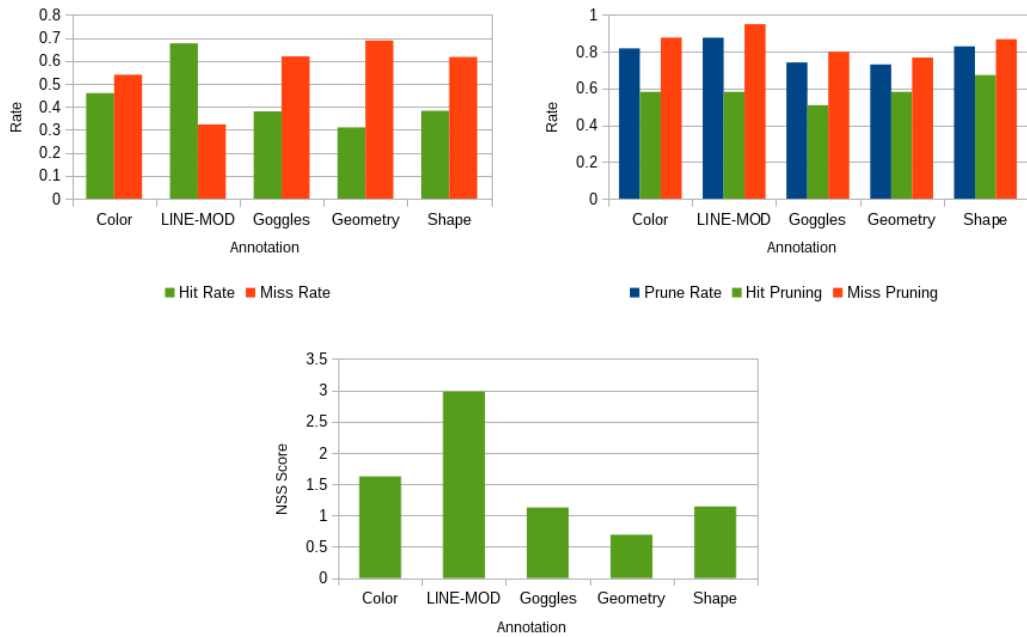


Figure B.43 Results of experiment B3 for the corn flakes package.

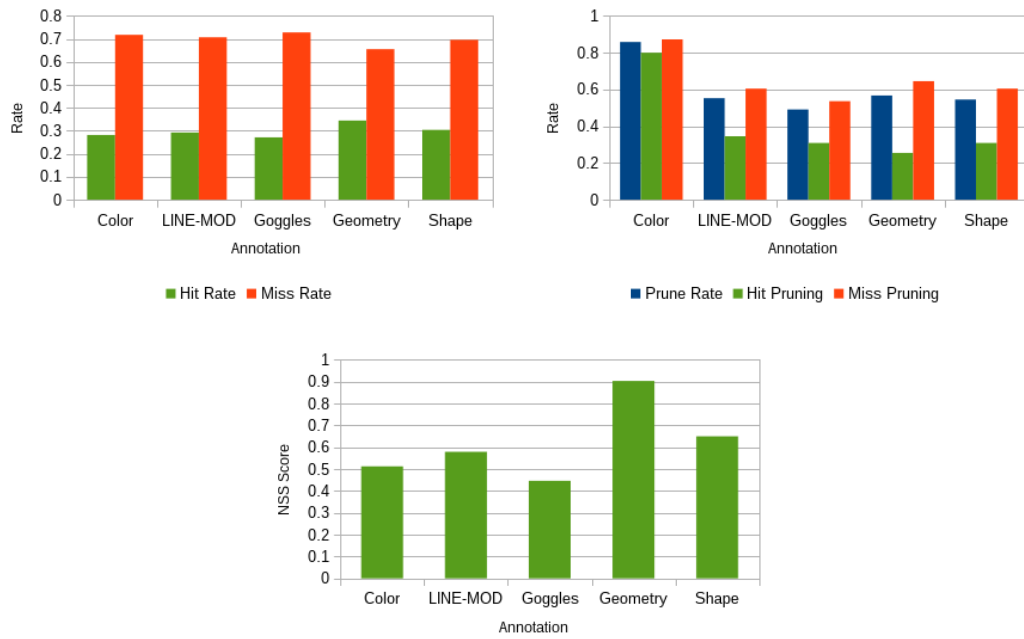


Figure B.44 Results of experiment B3 for the ice tea package.

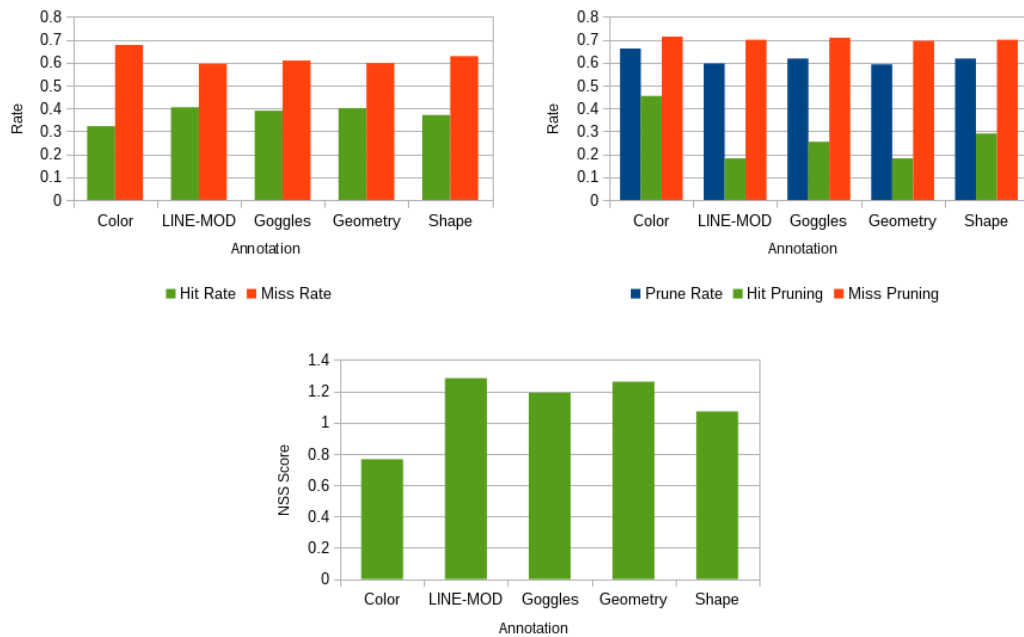


Figure B.45 Results of experiment B3 for the spatula.

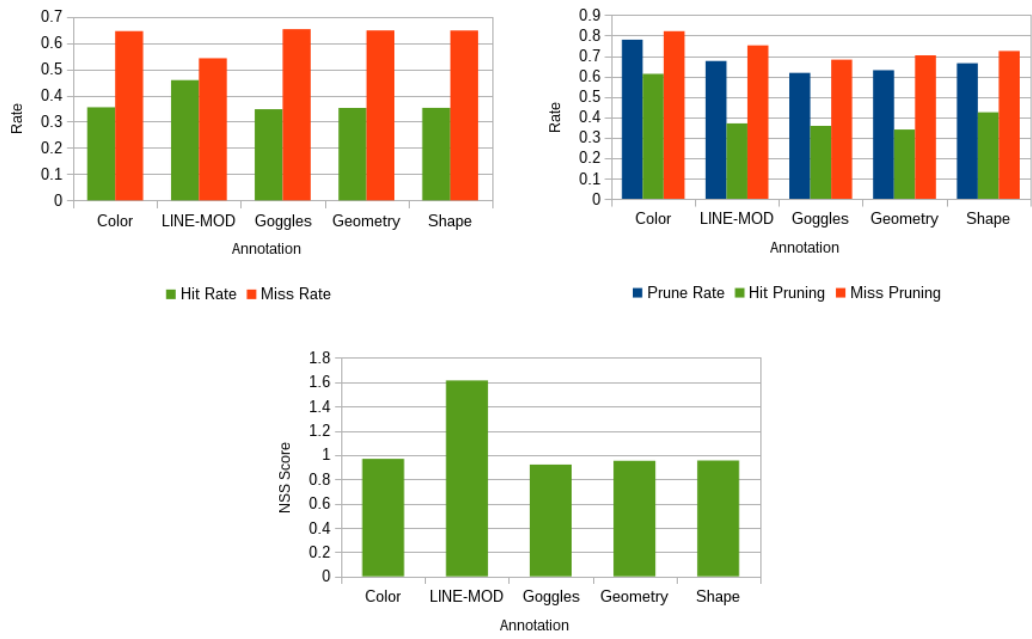


Figure B.46 Overall results of experiment B3.

B.8 Experiment C1

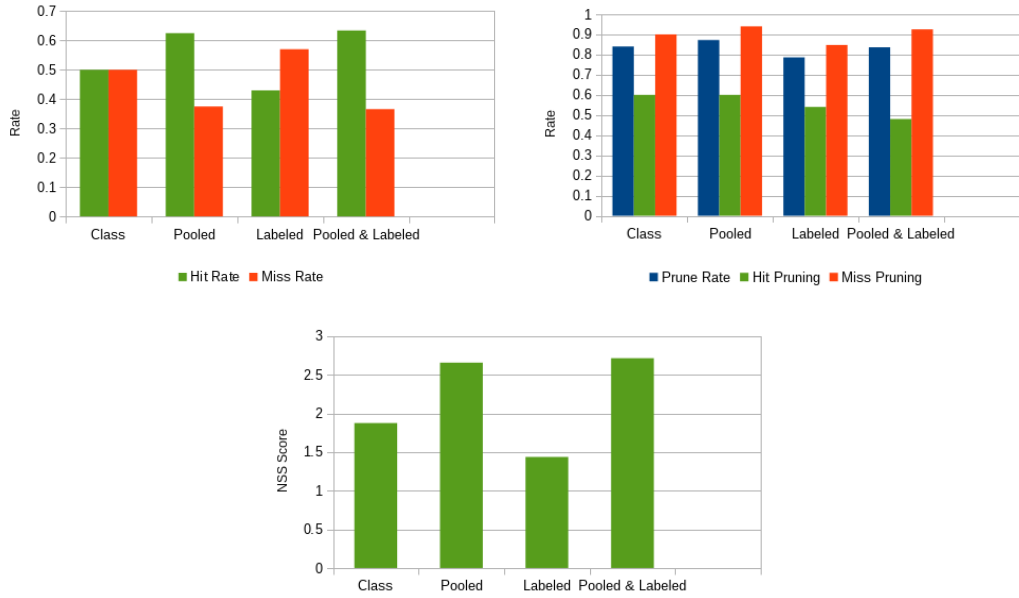


Figure B.47 Results of experiment C1 for the cornflakes package.

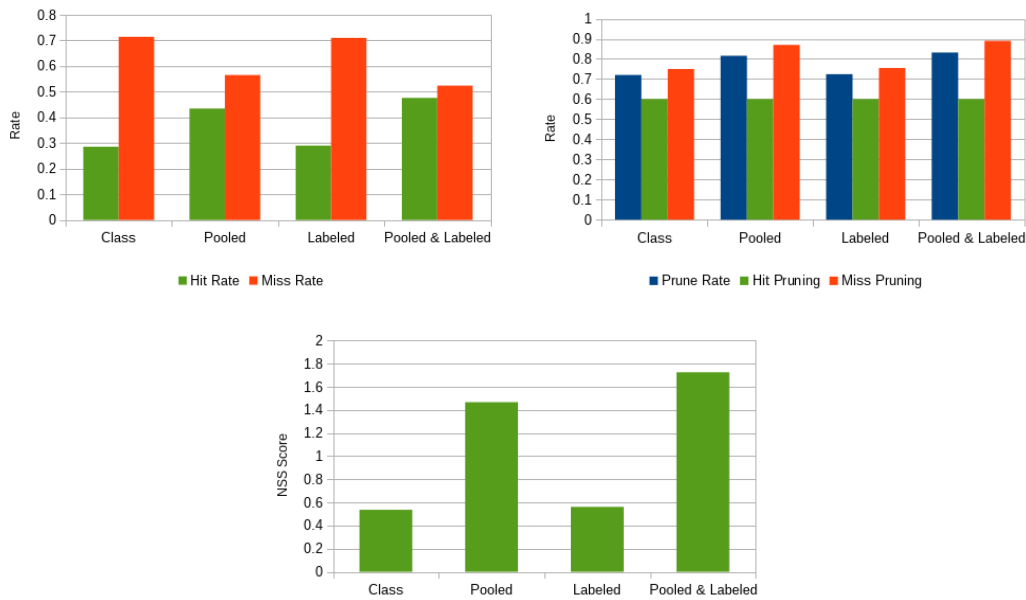


Figure B.48 Results of experiment C1 for the ice tea package.

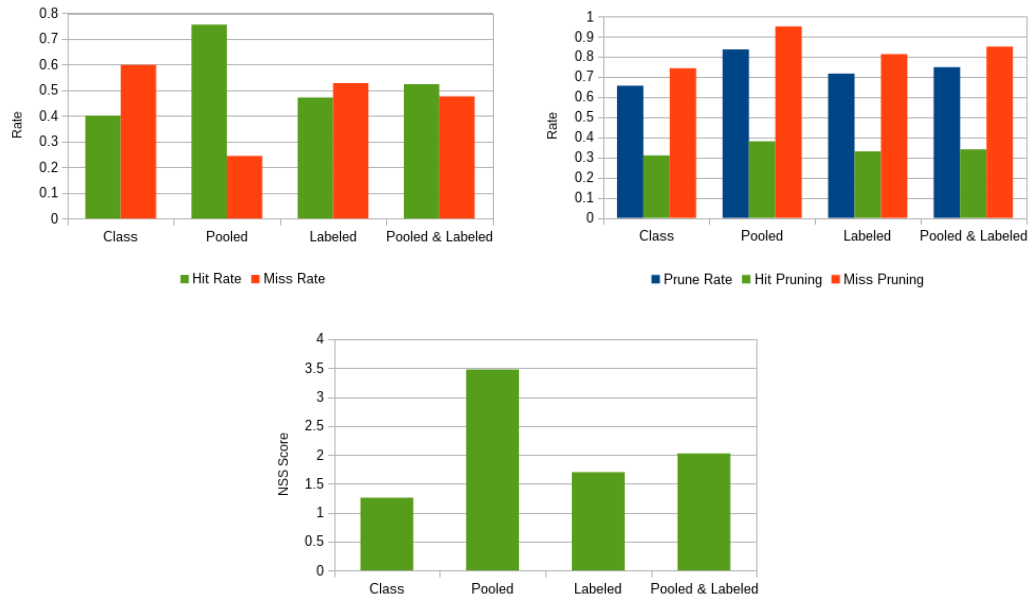


Figure B.49 Results of experiment C1 for the spatula.

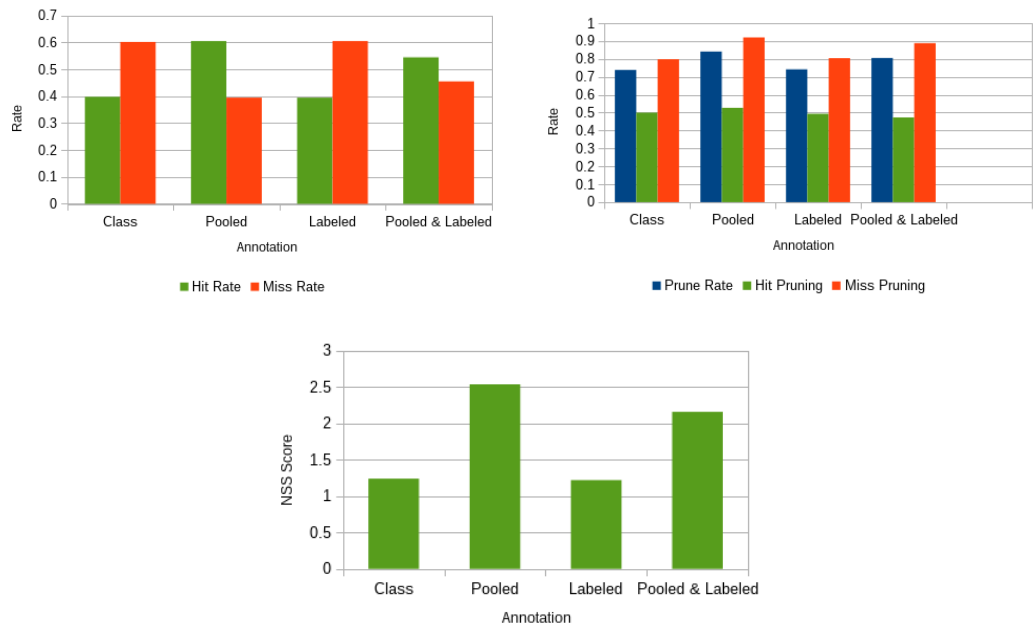


Figure B.50 Overall results of experiment C1.

Appendix C

CD Appendix

The attached CD contains following files:

- A digital version of this thesis
- The source code of the perception control framework
- The perception methods libraries compiled for Ubuntu 12.04 LTS ¹ for a 64 bit architecture
- The control strategies and rules which were used for the validation
- The training samples and the recordings of the reference scenarios in the form of ROS bags ²
- The validation results in the form of excel tables

¹<http://releases.ubuntu.com/12.04>

²<http://wiki.ros.org/Bags>