Knowledge-based scene exploration using computer vision and learned analysis strategies

U. Ahlrichs [1] , D. Paulus[2] , H. Niemann
Chair for Pattern Recognition (Computer Science 5)
University Erlangen–Nuremberg
Martensstr. 3, 91058 Erlangen, Germany
niemann@informatik.uni-erlangen.de

Preprint

AHLRICHS, ULRIKE, HEINRICH NIEMANN & DIETRICH PAULUS: *Knowledge-based scene exploration using computer vision and learned analysis strategies*. Int. Journal of Pattern Recognition and Artificial Intelligence, 18(4):627–664, 2004.

[1]Formerly with Chair for Pattern Recognition (Computer Science 5)

[2]Corresponding author with new address: Institut für Computervisualistik, Universitätsstr. 1 56070 Koblenz, Germany

Contents

U. Ahlrichs [3] , D. Paulus[4] , H. Niemann
Chair for Pattern Recognition (Computer Science 5)
University Erlangen–Nuremberg
Martensstr. 3, 91058 Erlangen, Germany
niemann@informatik.uni-erlangen.de

Knowledge-based scene exploration using computer vision and learned analysis strategies,

[3]Formerly with Chair for Pattern Recognition (Computer Science 5)

[4]Corresponding author with new address: Institut für Computervisualistik, Universitätsstr. 1 56070 Koblenz, Germany

Knowledge-based scene exploration using computer vision and learned analysis strategies

U. Ahlrichs [†], D. Paulus[‡], H. Niemann
Chair for Pattern Recognition (Computer Science 5)
University Erlangen–Nuremberg
Martensstr. 3, 91058 Erlangen, Germany
niemann@informatik.uni-erlangen.de

Abstract

In this contribution we demonstrate how the task of visual scene exploration can be solved by a knowledge-based vision system. During scene exploration, the system searches for a fixed number of a-priori known objects in a static scene. If not all objects are visible using the initial camera set-up, the camera parameters have to be adjusted and the camera has to be moved by the system. This problem is reduced to the choice of optimal camera actions. The information about the objects and the camera actions is uniformly represented in a semantic network. In addition, a control algorithm is provided that finds the optimal assignment from objects to parts of a scene based on a suitable analysis strategy. This strategy is acquired by the system itself using reinforcement learning methods. The paper focuses on aspects of knowledge representation concerning the integration of camera actions and on the integration of reinforcement learning methods in a semantic network formalism and applies them in a realistic setup. Experiments are shown for images of two office rooms.

1 Introduction

In future, autonomous mobile systems will presumably become more and more important in every-day life. Typical tasks of these systems include simple fetch-and-carry services as well as the guidance of persons through complex environments with moving obstacles [KI94]. Therefore, these systems need – among others – the capability to explore their environment. In this contribution, visual exploration is based on information from color images which are acquired using an off-the-shelve camera system. The pan-tilt camera can be moved on a linear sledge automatically under computer control. In this context, the goal of the scene exploration is to find a set of a-priori known objects within a static scene; the objects are depicted in Figure 1. Since some of them may not be visible with the initial camera set-up, the camera parameters have to

[†]Formerly with Chair for Pattern Recognition (Computer Science 5)

[‡]Corresponding author with new address: Institut für Computervisualistik, Universitätsstr. 1 56070 Koblenz, Germany

[0]This work was partially supported by the “Deutsche Forschungsgemeinschaft” under grant NI 191/12–1. Only the authors are responsible for the content.

Figure 1: The set of objects that have to be found contains an adhesive tape dispenser, a glue-stick, a stapler, a text marker, a hole punch, and a flower pot. These objects are depicted in the upper row. The lower row shows two different parts of a typical scene that has to be analyzed.

be adjusted. For example, in Figure 1 the image of the scene that is shown on the left contains the hole punch while this object is not visible in the second image on the right that visualizes another part of the same scene. The scene exploration covers the choice of optimal camera parameters as well as the analysis of the image data. To analyze the image data, a-priori knowledge about the application domain is essential. This knowledge is represented using a semantic network which combines the information on the structure of the scenes with the information about so-called *camera actions*. These actions select the optimal values for camera parameters such as position, zoom, viewing angle, etc., and perform camera movements accordingly. Our semantic network formalism allows the representation of both, the declarative knowledge of the scene structure as well as the procedural knowledge of image analysis algorithms and the camera movements, in an uniform way.

During scene exploration the information, which is stored in the semantic network, is exploited by a control algorithm. On the one hand, this algorithm has to search for the optimal match between the information from the sematic network and the image data. We apply the well-known A* graph search to solve this problem [Nil82]. However, there are also other optimization techniques suitable to find an optimal match as outlined in Section 3. On the other hand, the control algorithm has to decide which operation will be performed during the next analysis step. While the semantic network is designed manually by the user of a system, the sequence of analysis steps is learned by a reinforcement learning approach. The integration of the learning method into the control algorithm shows one possibility to combine semantic networks with learning methods. There are also approaches that acquire the semantic network automatically using for example a labeled sample set of objects [Win94]. However, this acquisition needs a bigger effort than the handcrafted construction of a semantic network and it also introduces new sources of errors. Therefore, we restrict ourselves to the learning of the sequence

of analysis steps. Since all other parts of the control algorithm are problem-independent, the adaptation of analysis strategies results in a control algorithm that can be easily transfered to a new application domain.

Recently, only a few comparable systems have been published. Systems that are close to ours from the knowledge representation point of view are described in [BFKS99, DCB+89, BH96]. These systems use semantic networks for the interpretation of assemblies which are constructed using wooden construction-kits, for the interpretation of street scenes or for planning views to achieve a more reliable object recognition. Apart from semantic networks a lot of other representation formalisms are available such as Bayesian networks. An overview on knowledge-based image understanding systems can be found in [CL97]. Bayesian networks have reached more attention during the last years since the representation of uncertain information has gained more interest. In [KKW98] a system for recognition of free-form objects is outlined which represents the information using Bayesian networks. These networks are also used to guide the control algorithm. In [Rim93], for example, a system decides which operation is performed during the next analysis step based on information from a multiple compound Bayesian network. Similar approaches can be found in [MHZR99, Rob97]. In [BDB99] the control strategy is learned using reinforcement learning. The goal was to select the best sequence of operations among a set of available preprocessing und segmentation methods. The task of the system is the detection of roof tops in arial images. The same system was applied to learn analysis strategies using our experimental set-up [DAP01].

Our contribution focuses on two points. First, the learning of analysis strategies is considered. Second, the integration of camera actions in a previously passive semantic network is described. After a short introduction into the semantic network formalism and the control algorithm in Section 2 and Section 3 the integration of reinforcement learning methods into the control algorithm is outlined in Section 4. Theory is applied to the exploration of office scenes in Section 5. This section also includes a detailed description of the integration of camera actions into the semantic network. Afterwards, the evaluation of the system by means of experiments is described in Section 6. The sequence of computations preformed in the experiments is related to the structure of the paper in Figure 2. The paper concludes with a summary and an outlook.

2 Knowledge Representation with Semantic Networks

Semantic networks have a long history in pattern recognition which dates back to the late sixties when Quillian introduced these networks to model the semantics of English words [Qui68]. The networks correspond to directed, labeled graphs, where the nodes of the graphs contain information about objects, events, or facts. The formalisms usually distinguish two different types of nodes, *concepts* and *instances*. While concepts contain an intentional description of items, instances refer to their extensional representation in the data. The nodes are connected to each other by different links. Most formalisms include, for example, the *part* and the *specialization* link where the latter is used to model inheritance from a general concept to a more special one. Later publications [Woo75, Bra77] stressed the importance of a restricted set of different link and node types since this allows for the use of an problem-independent control algorithm. Systems like KRYPTON [BGL85], NIKL [KBR86] SB-ONE [Pro90] or PSN [LM79] and ERNEST [Sag85, SN97] took these suggestions into account. Additionally, Minsky [Min75] proposed to structure the contents of the nodes in subconceptual units like attributes or relations.

The first vision systems that are based on semantic networks were published in [Min75,

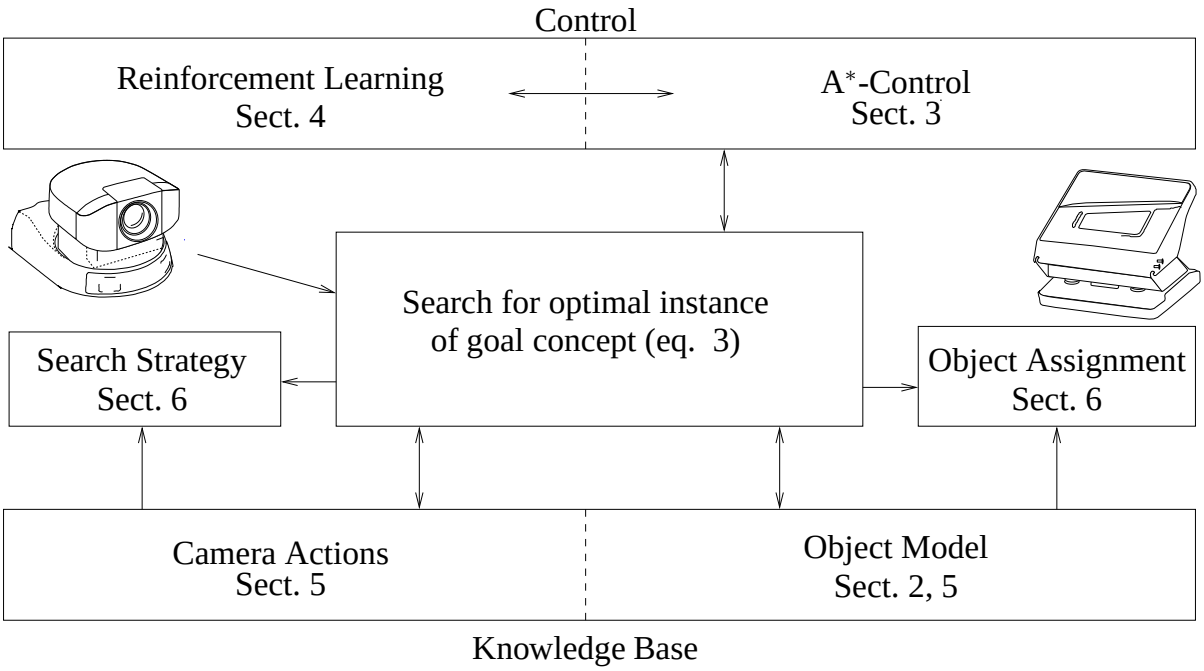

Figure 2: Structure of the Contribution

Fre77]. A lot of other systems, e.g. VISIONS [HR78], ALVEN [TMCZ80], ACRONYM [Bro81], SCHEMA [DCB+89], SIGMA [MH90], LLVE [MH90], CITE [DC97b], SOO-PIN [DCL96, DC97a], and systems based on ERNEST [Sag85, Sal97] followed.

In this paper, we use the semantic network formalism ERNEST [NSSK90] which contains some extensions that are especially valuable for applications from the field of pattern recognition. In Figure 3 a network $\mathcal{N}$ is shown which uses the formalism for the representation of information about houses. The ovals in Figure 3 correspond to concepts $C_i \in \mathcal{N}, i = 1, \ldots, n$, e.g. the concept "window" represents information about windows. All concepts have the same structure that is determined by the following definition:

$$
\begin{array}{rcll}
\mathcal{C} & = & (& \\
 & & N_r, & \text{name} \\
 & & P^*_{ci}, & \text{set of context-independent part links} \\
 & & P^*_{cd}, & \text{set of context-dependent part links} \\
 & & K^*, & \text{set of concrete links} \\
 & & V^*, & \text{set of specialization links} \\
 & & H, & \text{set of modalities} \\
 & & R^{sn*}, & \text{set of relations} \\
 & & A^{sn*}, & \text{set of attributes} \\
 & & \boldsymbol{G}, & \text{judgment} \\
 & & \boldsymbol{pr} & \text{priorities} \\
 & &) & .
\end{array}
\tag{1}
$$

A network is a set of concepts that are related to each other by links, i.e. $\mathcal{N} = \{\mathcal{C}_i\}, i = 1, \ldots, n$. Concepts provide the framework and the structure for the description of objects, events, actions, etc.; the slots are filled with individual values in so-called *instances*. Instantiation in our case happens during image analysis. The concepts $\mathcal{C}_i$ are linked to each other by different kinds of links which are subsumed into different sets in (1). The part link is used to represent the de-

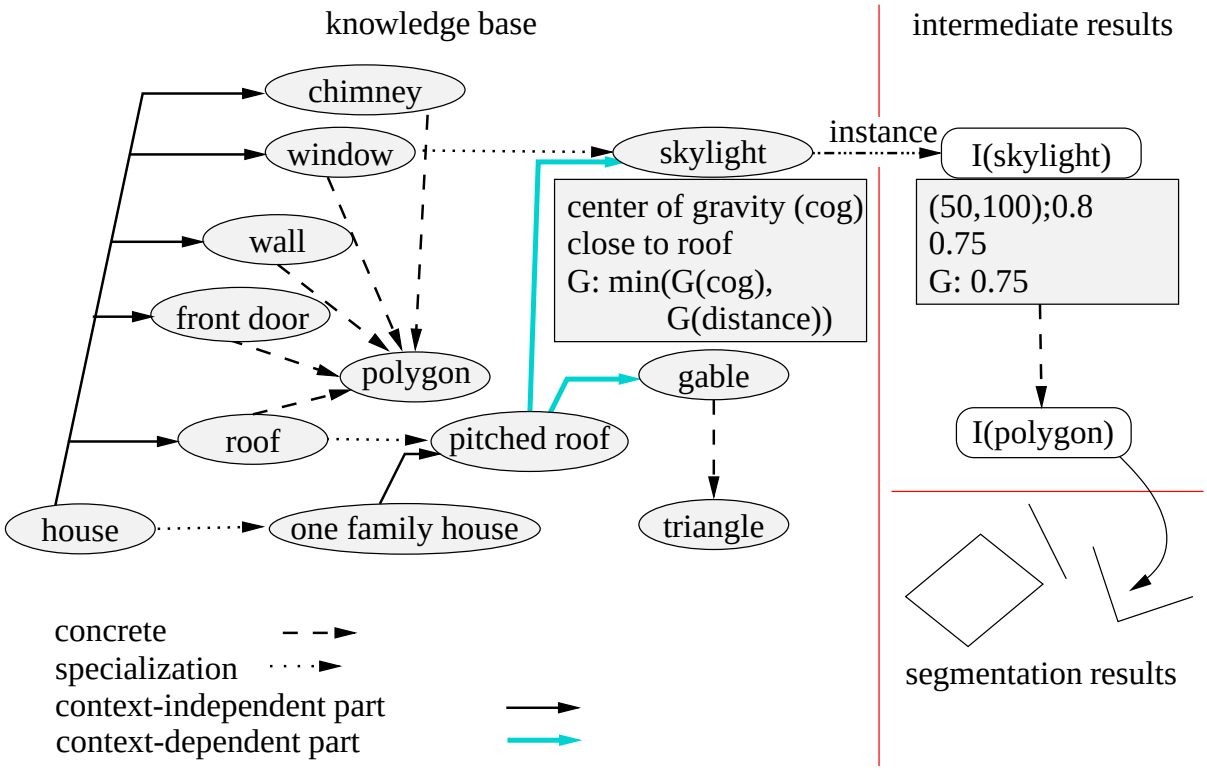

Figure 3: Example for knowledge representation with ERNEST

composition of a concept. The concept “house” in Figure 3, for example, is decomposed into the concepts “roof”, “front door”, “wall”, “window”, and “chimney”. ERNEST distinguishes two different part links, a *context-dependent* and a *context-independent* one. While all parts of the concept “house” are context-independent, the link between the concepts “pitched roof” and “skylight” is a context-dependent part link. For these links it is necessary to establish a context for the instantiation of the concept that represents the constituent. Therefore, a (partial) instance of the concept “pitched roof” has to be built before an instance for the concept “skylight” can be calculated. Along the spezialization link, information is inherited from the more general concept, e.g. the concept “one family house” inherits the part links from the concept “house”. The concrete link was added to the formalism to cope with the special structure of pattern recognition problems. This link combines concepts originating from different levels of abstraction or from different conceptual systems. Therefore, it allows for the representation of hierarchies. In Figure 3 a concrete link connects the concepts “gable” and “triangle”, where the first concept lives in the world of objects and the second in the world of images.

Additionally, all concepts are structured by sets of sub-conceptual entities like the set of attributes A^{sn*} or the set of relations R^{sn*}. The concept “skylight”, for example, contains the attribute “center of gravity” and the relation “close to roof”. The semantic network does not only represent declarative knowledge. Functions are added that calculate values for attributes and judgments for attributes and relations. Therefore, the structure of attributes $\mathcal{A}$ in ERNEST

is defined as

$$\begin{array}{rcll} \mathcal{A} & = & (& \\ & & N_r, & \text{name} \\ & & w, & \text{attribute value} \\ & & J_w, & \text{restrictions} \\ & & \boldsymbol{G} & \text{judgment} \\ & &) & . \end{array} \tag{2}$$

The judgment $\boldsymbol{G}$ provides a measurement of uncertainty which describes how well the a-priori knowledge of restrictions J_w fits to the information found in the data. Concepts also contain a judgment function $\boldsymbol{G}$ which calculates an assessment for an associated instance. This assessment is usually based on the judgments of attributes, relations, and links.

There are two other structures in (1) that are special to ERNEST. First, different occurences of one object are modelled by *modalities* $H_i, i = 1, \ldots, l$. These different occurences exist, for example, in computer vision where different views for one object are provided and for each view different parts of the object are visible. It would be possible to represent this information using the specialization link. However, this would lead to fairly complex representations. Second, so-called *priorities* $\boldsymbol{pr}$ are introduced. These vectors describe how urgently a concept should be instantiated during analysis.

Additionally, during analysis so-called *modified concepts* are generated. These nodes are distinguished from concepts and instances in ERNEST. They represent intermediate results and are calculated either based on the data-driven or model-driven propagation of restrictions. Further information concerning ERNEST can be found in [SN97, p. 295 – 311], [Sag85, Nie90].

The represented knowledge is exploited by a control algorithm as we have already mentioned in Section 1. This algorithm is outlined in the following section.

3 A Problem-Independent Control Algorithm

The task of the control algorithm is twofold. Its first subtask comprises the search for an optimal interpretation of the data. The solution to this subtask is described in the following section. The second subtask is the search for the optimal sequence of the concepts' calculation which is essential for an efficient use of the knowledge. We treat this task in Section 3.2.

3.1 Search for the optimal interpretation

During analysis the optimal interpretation with respect to the judgment function for the observed data is determined. For image analysis, observed data are images and the goal is a description of the data. Rather than using the pixels of the image directly, images are usually transformed to more abstract descriptions first. After filtering which transforms images into images, data-driven processing of images detects simple geometric features such as lines, points or areas in the images. This stage of computation is called *image segmentation*. The semantic network modelling of our application thus consists of concepts for the office objects such as the concept "adhesive tape", more general concepts such as "office scene" and concrete concepts such as "color region"; the latter is computed by image segmentation. In Section 5 we will discuss the structure of the network in detail.

The optimal interpretation is described by the content of instances for a special concept, which is called *goal concept* $I(C_g)$. In ERNEST every semantic network contains at least one

goal concept. These concepts represent sufficient knowledge about the application domain such that their instances hold all necessary information about the observed data. Goal concepts depend, of course, on the application domain and the intention of the system designer. Like every concept in the semantic network goal concepts use a judgment function to assess their associated instances. Since there are ambiguities in the data due to segmentation errors as well as in the semantic network due to modalities, usually more than one instance for a goal concept is calculated during analysis. Each of these instances corresponds to an interpretation of the data, where the judgment of the instance describes the match between a-priori knowledge and data. Therefore, the search for an optimal interpretation of the data can be reduced to the search for the optimally judged goal instance. This is described formally by

$$I^*(C_g) = \underset{\{I(C_g)\}}{\operatorname{argmax}} \boldsymbol{G}(I(C_g)|\,\mathcal{N}, \mathcal{B})\,, \tag{3}$$

where $\mathcal{B}$ denotes the results of image segmentation.

Different instances that are calculated for one goal concept are so-called *competing instances*. These instances represent different interpretations and are stored during analysis in different so-called *search tree nodes* $v_q, q \in \mathbb{N}$. Each of these nodes represents a state of analysis and contains the semantic network as well as all computed modified concepts and instances, i.e.

$$v_q = \mathcal{N} \cup \{I_{j_q}(C_i), M_{k_q}(C_l)|C_i, C_l \in \mathcal{N}, j_q, k_q \in \mathbb{N}\}\,, \tag{4}$$

where j_q and k_q denote the number of instances and modified concepts that are generated for a concept within v_q. Each competing instance is stored in a different search tree node. All nodes are members of a search tree, where the root of this tree corresponds to the search tree node that contains the uninterpreted data. During analysis the tree is expanded until a node is reached which contains a goal instance or until analysis stops because the interpretation of the data failed. All search tree nodes also contain a judgment, which is defined as

$$\boldsymbol{G}(v_q) = \begin{cases} \widehat{\boldsymbol{G}}(\widehat{\boldsymbol{G}}(I^{v_q}(C_1)), \ldots, \widehat{\boldsymbol{G}}(I^{v_q}(C_n))|C_g) & \text{if } C_g \text{ has not been instantiated yet and } v_q \text{ contains the instances } I^{v_q}(C_1), \ldots, I^{v_q}(C_n) \\ \boldsymbol{G}(I^{v_q}(C_g)) & \text{if } C_g \text{ is instantiated} \end{cases}, \tag{5}$$

where $\widehat{\boldsymbol{G}}(I^{v_q}(C_i))$ denotes the *estimation* of a judgment for the concept $C_i \in \mathcal{N}$. The estimation of judgments is necessary since instances are calculated during analysis and a search tree node may not contain instances for each concept in the semantic network at the considered point of time. Using (5) the problem of searching for an optimal goal instance can be reduced to find the optimal search tree node in the search tree. This is also equivalent to the search for an optimal interpretation as we explained in (3).

All paths in the search tree can be distinguished by their judgments. Here, the judgment of a path coincides with the judgment of the search tree node which is the leaf of the path. In order to find the optimal interpretation for the data, the optimal path has to be searched in the search tree. This task can be solved by a heuristic graph search algorithm like the A*-algorithm [Nil82]. The use of a search algorithm is essential since the efficiency of the analysis would enormously decrease if all possible paths in the search tree are expanded and explored. If the A*-algorithm is used, restrictions concerning the judgment functions have to be considered to guarantee the admissibility of the search. First, we need a monotonous function, and second, the estimations of all judgments in (5) have to be optimistic, i.e. they are not allowed to be lower

than the actual judgment. A judgment function for our application that meets these criteria is given in Section 5.2.

The expansion of the search tree follows a set of transformations which can be applied to the members in a search tree node. The application of a transformation T results in the generation of new search tree nodes along new paths in the search tree, i.e

$$T\{v_k\} \rightarrow \{v_j\}_{j=1,\ldots,m}\,. \tag{6}$$

Usually, more than one transformation is applicable to a search tree node. Therefore the control has to provide a sequence of transformations that leads to the interpretation of the data. The solution to this problem is outlined in the following section.

3.2 Determine the sequence of instantiations

Besides the calculation of the optimal interpretation of the data, the control also has to determine the sequence of operations that transform the root of the search tree node to the node that contains the interpretation of the data. Since the chosen sequence $T_{k_1},\ldots,T_{k_{m-1}}$ influences the efficiency of the search for the interpretation, the optimization problem arises, how to find the optimal sequence of transformations $T^*_{k_1},\ldots,T^*_{k_{m-1}}$. This problem is described by

$$\begin{aligned} T^*_{k_1},\ldots,T^*_{k_{m-1}} = \underset{T_{k_1},\ldots,T_{k_{m-1}}}{\operatorname{argmax}} \quad & G\{T_{k_1}\{v_1\} \rightarrow \{v_{j_1}\}_{j_1=1,\ldots,n_1}, \\ \forall v_l \in \{v_{j_1}\}_{j_1=1,\ldots,n_1} \quad & T_{k_2}\{v_l\} \rightarrow \{v_{j_2}\}_{j_2=1,\ldots,n_2} \\ & \vdots \\ \forall v_l \in \{v_{j_{m-1}}\}_{j_{m-1}=1,\ldots,n_{m-1}} \quad & T_{k_{m-1}}\{v_l\} \rightarrow \{v_{j_m}\}_{j_m=1,\ldots,n_m}\} \end{aligned} \tag{7}$$

where in general T_{k_j} is one of six so-called *inference rules* explained in [SN97]

$$T_{k_j} \in \{\text{rule}_1, \text{rule}_2, \ldots, \text{rule}_6\}\,, \tag{8}$$

and T_{k_j} does not have to be the same for all v_l in $\{v_{j_i}\}_{i=1,\ldots,m-1}$. Rule$_1$ to rule$_6$ describe how to calculate modified concepts and instances. The first two rules deal with the instantiation of (modified) concepts. A (modified) concept can only be instantiated if instances for all its parts and concretes are available. If a (modified) concept has a context-dependent part, first a partial instance is built. This instance is completed after the instantiation of the corresponding context-dependent part. The third rule handles the optional links, that are only instantiated if the control decides that further information is needed. The expansion of the network is done using the fourth and fifth rule where the fourth rule describes a data-driven expansion and the fifth rule states the conditions for a model-driven expansion of a concept. Using the sixth rule instances for concept can be estimated before the analysis has started. A detailed explanation of these rules is given in [SN97].

However, the transformations are not applied to the search tree nodes themselves but to the elements of the search tree nodes, i.e. the concepts, modified concepts and instances. Since transformations can operate on more than one concept or modified concept in a search tree node, the control also has to determine which concept or modified concept should be considered

during the next point in time. Therefore (7) can be rewritten by

$$\begin{aligned} & O_1^* T_{k_1}^* O_2^*, \ldots, O_{m-1}^* T_{k_{m-1}}^* O_m^* = \\ & \underset{O_1 T_{k_1} O_2, \ldots, O_{m-1} T_{k_{m-1}} O_m}{\operatorname{argmax}} G\{T_{k_1}\{O_1\} \rightarrow \{v_{j_1}\}_{j_1=1,\ldots,n_1}, \\ & \exists v_l \in \{v_{j_1}\}_{j_1=1,\ldots,n_1} \; \exists O_2 \in v_l T_{k_2}\{O_2\} \rightarrow \{v_{j_2}\}_{j_2=1,\ldots,n_2} \\ & \vdots \\ & \forall v_l \in \{v_{j_{m-1}}\}_{j_{m-1}=1,\ldots,n_1} \; \forall O_m \in v_l T_{k_{m-1}}\{O_m\} \rightarrow \{v_{j_m}\}_{j_m=1,\ldots,n_m}\} \end{aligned} \tag{9}$$

Thus, we have to search for an optimal sequence of network objects O_i and transformations T_{k_j} that are applied to these objects. Network objects correspond to concepts, instances, or modified concepts. As we have stated above, transformations need not be the same for search tree nodes on different paths in the search tree. The same holds for network objects O_i of different $v_l \in \{v_{j_i}\}_{j=1,\ldots,m}$.

ERNEST used to solve these two optimization problems seperately in a simple way. To determine the sequence of network objects, priorities are considered [SN97]. These priorities are calculated for each concept in the semantic network. They describe the position of a concept within the network regarding the goal concept or primitive concepts. A primitive concept is a concept that contains no link to parts or concretes. For example, the first and second priority give a measure for the length of the path from a concept to a primitive concept along concrete links. Priorities are application-independent and can be combined with a precedence value for each concept. This value is usually application-dependent and is determined by the designer of a system.

The sequence of transformations is a-priori fixed by the structure of the control algorithm. The algorithm prefers instantiations to the expansion of the semantic network. If neither rule 1 or rule 2 nor rule 4 or rule 5 can be applied the expansion of optional links is considered.

In this paper we suggest another approach to solve the optimization problem in (9) which takes into account the dependencies between network objects and applicable transformations. The structure of the solution is adopted from Markov decision processes. These processes can be used to describe which network object is reached after application of a transformation to the current object. Since a transformation is equivalent to one of the inference rules mentioned below we have to deal with uncertain results of a transformation. This uncertainty originates from the underlying procedural knowledge which is integrated into each concept and which is executed whenever a concept is instantiated. For example, the number of competing instances which are calculated for one concept is dependent on the analyzed data and can vary considerably although the same inference rules, i.e. the same procedures, are applied. This influence is reduced if a sequence of transformations is adapted to the application domain. For example, a model-driven propagation of restrictions is valuable for application domains with multiple competing interpretations.

The number of competing instances determines the number of successors for a search tree node. The higher the number of search tree nodes that are generated during an analysis the lower the efficiency of the analysis. Therefore, the sequence of transformations and network objects have to be chosen very carefully.

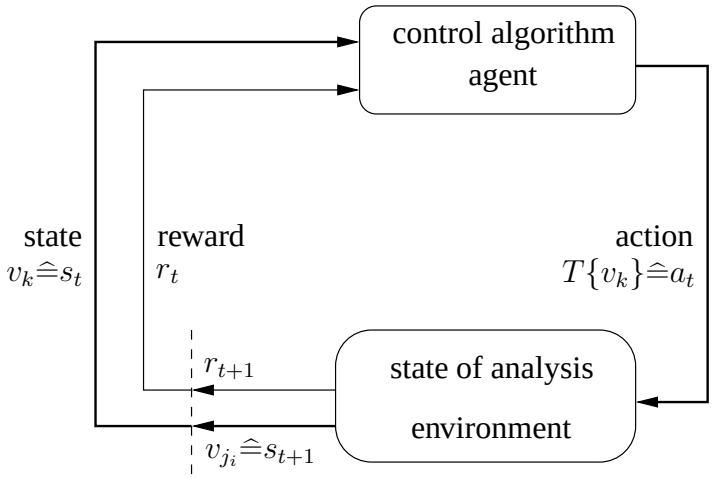

Figure 4: ERNEST-control as reinforcement learning problem (after [SB98])

4 Learning during analysis

The control algorithm which is used to guide the analysis includes problem-independent and problem-dependent parts. While problem-independent parts can be used for arbitrary application domains, problem-dependent parts have to be adapted. The adaptation is usually a very time consuming procedure if this has to be done manually by the user of a system. If a problem-dependent module can be replaced by a module which adapts itself to a new application domain, this strongly simplifies the transfer of a knowledge-based system to a new application domain.

In ERNEST the control algorithm contains only one problem-dependent part. This part deals with the selection of the optimal sequence of transformations and calculated network objects. Priorities which are problem-independent provide one means to guide the analysis as outlined in Section 3.2. In addition, a *precedence value* for concepts is determined by the user depending on the application domain. During analysis, the control algorithm first considers a concept's precedence value. Priorities have an effect on the instantiation sequence only if all concepts that can be processed have the same precedence. Almost all systems that used the ERNEST-control were based on user-defined precedence values to improve the efficiency that can be achieved with problem-independent priorities. In this section we describe an approach which learns the precedence values during analysis using Reinforcement Learning. This leads to a significant improvement of the efficiency as the experiments show in Section 6.

4.1 Reinforcement Learning with ERNEST

The integration of the learning methods into the ERNEST control scheme results from the control cycle of transformations and search tree nodes which was described in Section 3.2. The framework for reinforcement learning methods resembles this control scheme. In Figure 4 the ERNEST control is depicted in terms of the reinforcement learning framework. Starting with a search tree node v_k the control has to decide which transformation should be applied during the next analysis step. The transformation $T\{v_k\}$ leads to a set of new search tree nodes $\{v_j\}_{j=1,\ldots,m}$ as we outlined in (6). Every search tree node represents a state of analysis. Since the application of transformations are combined with the A*-graph search, only the optimal search tree node

$v_{j_i} \in \{v_j\}_j = 1, \ldots, m$ in the search tree is considered during the next analysis step. The iteration of this scheme yields a sequence of transformations as described in (7). If in Figure 4 the search tree nodes v_k and v_{j_i} are substituted by the processed network object we get the optimization problem shown in (9).

The same processing scheme holds for reinforcement learning methods. The introduction to these methods is adopted from [SB98]. Starting with a *state* $s_t \in S$ an agent selects an *action* a_t. The set of states S includes the different states of the environment. After this action was executed a new state of the environment s_{t+1} is reached. In order to assess the execution of an action in a state the environment gives a *reward* r_{t+1} to the agent. The decision for an action is made by the agent based on a so-called *policy*

$$\pi_t(s, a) = p(a_t = a | s_t = s) \tag{10}$$

which describes the probability for an action a to be selected in a state s. The agent's goal is to find a policy that maximizes the *return*

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,, \tag{11}$$

where the so-called *discount rate* γ, $0 \leq \gamma \leq 1$, weights the value of a reward received k time steps ahead in the future. To guarantee the convergence of (11) the discount rate can be chosen equal to 1 only if the agent deals with an episodic task. Episodic tasks break into subsequences which end with a special state, the so-called *terminal* state.

Therefore, the control cycle of reinforcement learning is similar to the ERNEST-control. The environment corresponds to the state of analysis that is represented by a search tree node, the agent corresponds to the control algorithm.

The calculation of the return is based on *Markov Decision Processes* which model the behaviour of the environment. Therefore, all states have to fulfill the Markov property. Additionally to states and actions, a Markov Decision Process is determined by *transition probabilities* and the *expected value of the next reward*. For Markov Decision Processes it is possible to calculate the expected return the agent receives if it starts in state s and follows policy π

$$Q^{\pi}(s, a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\} . \tag{12}$$

We refer to $Q^{\pi}(s, a)$ as *action-value function for policy* π. The values of this function describe how well the execution of an action a in a state s is, regarding the return, if policy π is followed. These values are called Q*-values* in the following.

If the action-value function is known the optimal policy can be determined by

$$\forall a \in A_t(s) : \pi(s, a) = \begin{cases} 1 & \text{if } \ a = \operatorname{argmax}_{a'} Q(s, a') \\ 0 & \text{otherwise} \end{cases} \,, \tag{13}$$

where $A_t(s)$ denotes the set of all executable actions in state s. If more than one action has an optimal Q-value, equal probabilities have to be assigned to these actions. This leads to an optimal policy, too.

If the model of the environment is completely known, i.e. if all transition probabilities and expected values of rewards are available, (12) can be solved by dynamic programming leading to an optimal policy. Usually a complete model of the environment is not available. Therefore sampling methods are applied to estimate the values of $Q^{\pi}(s, a)$. One of these learning methods is outlined in the following section.

4.2 Learning methods

In this paper, we apply *Monte Carlo Methods* to learn the optimal policy. They are described in the following [SB98]. These methods are well suited for episodic tasks like the one that we consider here. Additionally, they do not assume a completely known model of the environment. In fact, they acquire the action-value function based on sample sequences of states, actions, and rewards which arise through the agent's interaction with the environment. If the action-value function (12) is known the optimal policy can be easily determined by (13).

The learning is done by an iterative scheme that is called *general policy iteration*. It splits up into two steps:

1. *Evaluation of policy* π: This step generates an episode based on π_k. The new action-value function for all state action pairs (s, a) results from

$$Q^{\pi_k}(s, a) = E\{R_t | s_t = s, a_t = a\} . \tag{14}$$

   This expected value is approximated by averaging over the returns that have been received for a pair (s, a) in the preceeding episodes. In our application domain each state action pair can occur only once in an episode. Therefore, the average is calculated from the returns of different episodes. In other applications averaging also takes place within an episode if pairs are observed more than once. The update of the Q-value is done at the end of an episode.

2. *Improvement of policy* π: In this step the policy $\pi_{k+1}(s, a)$ is improved with respect to the action-value function. It holds

$$\forall s \in S : \pi_{k+1}(s) = \operatorname*{argmax}_a Q^{\pi_k}(s, a) , \tag{15}$$

   where $\pi_{k+1}(s)$ denotes the policy which was determined after $k + 1$ iterations.

Step 1 and step 2 are iterated until an optimal policy π^* has been determined. The optimality of a policy is defined by the judgment criterion to be optimized. The criterion which was used in our application is given in (21).

$$\pi_0 \xrightarrow{(1.)} Q^{\pi_0} \xrightarrow{(2.)} \pi_1 \xrightarrow{(1.)} Q^{\pi_1} \xrightarrow{(2.)} \pi_2 \xrightarrow{(1.)} \ldots \xrightarrow{(2.)} \pi^* \xrightarrow{(1.)} Q^{\pi^*} \tag{16}$$

Note, that in step 1 the agent is not allowed to follow a *greedy* strategy during the whole episode. In this case the agent would choose always the optimal action. However, this does not lead to the aquisition of any new information. There are many strategies to circumvent this problem. We chose an ϵ-greedy policy, where the probabilities to select an action a in state s are distributed according to

$$\forall a \in A_t(s) : \pi(s, a) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A_t(s)|} & \text{if } \; a = \operatorname{argmax}_{a'} Q^{\pi}(s, a') \\ \frac{\epsilon}{|A_t(s)|} & \text{otherwise} \quad . \end{cases} \tag{17}$$

The term $|A_t(s)|$ denotes the number of actions which are executable in state s. The higher the value of ϵ, the more the environment is explored, i.e. an action is selected which does not have the maximal Q-value. Besides Monte Carlo Methods also temporal difference learning or eligibility-traces could be applied. The Monte Carlo Methods are particularly suitable for the rewards that are given to the agent by the environment at the end of an episode.

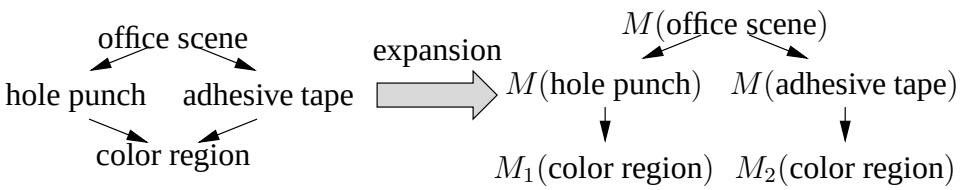

Figure 5: Expansion of a simple semantic network

4.3 Definition of states, actions, and rewards

Rewards, states, and actions vary for different application domains. In this section we define these three structures for the ERNEST-control, in particular for the semantic network that is used for knowledge representation.

States We choose the network objects themselves as states for the Markov Decision Process. In Figure 5 all the concepts "office scene", "hole punch", "adhesive tape", and "color region" as well as all modified concepts on the left side of the figure correspond to network objects. This choice is obvious since the reinforcement learning framework mirrors the ERNEST-control cycle as we outlined in Figure 4 and in (9). Note, that only network objects within *one* search tree node are distinguished while competing network objects are distributed on different search tree nodes. Therefore, we obtain a number of

$$|S| = 2(\sum_{j=1}^{n} n_j) + 1 \tag{18}$$

$$n_j \leq |C_j.P^*_{inv} \cup C_j.K^*_{inv}|\,, \tag{19}$$

different states for a semantic network of n concepts, where n_j denotes the number of different modified concepts for the concept C_j within a search tree node. The sets P^*_{inv} and K^*_{inv} refer to the sets of inverse part links and inverse concrete links, i.e. links that are opposite to the ones mentioned so far in the description of concepts. The extra state in (18) corresponds to the terminal state. This state coincides with the instance of the goal concept, since the analysis stops after this instance is calculated. Figure 5 shows how more than one modified concept is generated for the concept "color region". Since the concepts "hole punch" and "adhesive tape" are linked to the concept "color region" and these concepts represent information about different regions in a scene, two modified concepts and two corresponding instances have to be generated during analysis. The result of the expansion is depicted in the right part of Figure 5. This so-called *expanded network* contains five modified concepts that do not compete with each other. Therefore, eleven states have to be distinguished to learn the optimal analysis strategy for the network of Figure 5. The set of these eleven states contains the five modified concepts shown on the right side of Figure 5, five instances that have to be calculated to instantiate the network and the terminal state.

The definition of states is restricted by the prerequisite that all states have to fulfill the Markov property. For the state definition that we gave above this prerequisite is only met in combination with *dynamic action sets* that will be outlined in the following paragraph. This problem arises since the states have no information about the preceeding sequence of transformations which has led to the current state. For example, if the control has already calculated an instance for the concept "hole punch" of Figure 5, no other transformation can be applied to this

Action 7: *Jump to another network object*
IF the current network object cannot be expanded nor instantiated
THEN jump to another concept C_j or modified concept $M(C_k)$

Figure 6: Action a_7 to change the network object

network object. However, the information about the instantiation history is available within the search tree nodes. Therefore, at any point in time the agent gets only a restricted set of actions from which the agent can select the next action.

Dynamic action sets In Section 3.2 we explained how a sequence of transformations which is applied to the uninterpreted data leads to the data's interpretation. In this case transformations correspond to six inference rules that determine how instances are calculated, restrictions are propagated, and thus the semantic network is expanded. It is obvious to use these transformations as actions for the reinforcement learning. There is only one kind of action that is missing. The six inference rules do not select the network elements to which a transformation should be applied. Therefore, we introduce additional actions that describe transitions from one network object to another. We call them *jump actions*. The application of these actions does not result in new information during analysis, but they are needed to guide the agent through the expanded network. Jump actions are described by the rule depicted in Figure 6. They are only used if the currently considered network object cannot be expanded or instantiated. Action 1 to 6 correspond exactly to rule 1 to 6 (cmp. eq. 8), such that the set of possible actions is given by

$$A_t = \{a_1, \ldots, a_6\} \cup \{a_7^1, a_7^2, \ldots, a_7^m\}\,, \tag{20}$$

where m denotes the number of modified concepts of the expanded network. For example, m is five for the network depicted in Figure 5. Using the different actions the following sequence leads to an instantiation of the network shown in Figure 5:

$$\begin{aligned}
&\text{office scene} \xrightarrow{a_1} M(\text{office scene}) \xrightarrow{a_5} M(\text{office scene}) \xrightarrow{a_5} M(\text{office scene}) \xrightarrow{a_7^2} \\
&M(\text{hole punch}) \xrightarrow{a_5} M(\text{hole punch}) \xrightarrow{a_7^3} M(\text{adhesive tape}) \xrightarrow{a_5} M(\text{adhesive tape}) \xrightarrow{a_7^4} \\
&M_1(\text{color region}) \xrightarrow{a_2} I_1(\text{color region}) \xrightarrow{a_7^5} M_2(\text{color region}) \xrightarrow{a_2} I_2(\text{color region}) \xrightarrow{a_7^2} \\
&M(\text{hole punch}) \xrightarrow{a_2} I(\text{hole punch}) \xrightarrow{a_7^3} M(\text{adhesive tape}) \xrightarrow{a_2} I(\text{adhesive tape}) \xrightarrow{a_7^1} \\
&M(\text{office scene}) \xrightarrow{a_2} I(\text{office scene})
\end{aligned}$$

The different states for reinforcement learning are identified by different numbers. Here, $M(\text{office scene})$ gets state number 1, $M(\text{hole punch})$ state number 2, $M(\text{adhesive tape})$ state number 3, and $M_1(\text{color region})$ and $M_2(\text{color region})$ state numbers 4 and 5, respectively. Note, that action a_5 is applied twice to the modified concept $M(\text{office scene})$: This action leads to the model-driven expansion of a concept. With every application of this action just one link is expanded. Since the concept "office scene" in Figure 5 has two part links this action has to be selected twice.

Get current network object from search tree node					
IF	the network object is a concept				
THEN	add a_1 to $A_t(s)$				
ELSE	IF	the network object is expandable			
	THEN	for all links of the network object which have not been expanded yet: add a_5 to $A_t(s)$			
	ELSE	IF	the network element is instantiable		
		THEN	add a_2 to $A_t(s)$		
		ELSE	FOR all modified concepts of the expanded network (except the current network object)		
				IF	the network object with number j is a concept or expandable or instantiable
				THEN	add a_7^j to $A_t(s)$

Figure 7: Determination of an action set based on information from the search tree node

The instantiation history has be taken into account such that the agent is allowed to choose actions only from a limited set, when such a choice is required for a network object during analysis. These *dynamic action sets* are determined by the environment using the algorithm which is shown in Figure 7. Starting with the current network object it is examined if the network object is a concept. Then a modified concept can be generated using a_1. If this is not the case, the action a_5 is added to the action set for every link of the network object that has not been expanded yet. For network objects that are instantiable the action set is built by action a_2 which results in the instantiation of a modified concept. If none of the inference rules is applicable, the new action set comprises only jump actions. Usually not all network objects in the expanded network can be processed during the next analysis step. In order to increase the efficiency of the learning process, jump actions are just added to the action set if their corresponding network objects are instantiable or expandable.

Using this definition of states and actions an optimal sequence for (9) can be learned by Monte Carlo Methods. Remember that the goal is not just to learn the sequence of transformations that leads to the instantiated goal concept, but also the acquisition of an optimal sequence of processed network objects. The ERNEST-control determines this sequence directly from the precedence and priority values of the concepts. Refering to the reinforcement learning approach the precedence values correspond to the Q-values of the jump actions. These precedence values change with every analysis step, since the Q-values for the jump actions vary for different states. Therefore, the reinforcement learning yields to a more powerful control which adapts itself to the analysis with every analysis step. In order to learn the Q-values, rewards are necessary which give hints to the agent which action is optimal in a certain state. These rewards are outlined in the following section.

Rewards Reinforcement learning is based on rewards that give hints to an agent which quality an executed action has in the current state. The rewards constitute the return which the agent has to maximize as was shown in (11). We use application-independent rewards to prevent that the problem of defining application-dependent precedences for concepts is shifted to the problem of defining application-dependent rewards.

Two different approaches to asses the actions in consideration of the different states are tested in the following. As we have seen in Section 3.2, the efficiency of the analysis in highly influenced by the number of search tree nodes which are generated during the analysis of the data. Therefore, the first approach assesses every episode by the number of search tree nodes according to

$$r_T = \frac{1000 - |v_{ji}|}{1000}\,, \tag{21}$$

where T denotes the end of the episode and the number of calculated search tree nodes v_{ji} (cmp. eq. 4) is determined for the current episode. In this case the return for each state corresponds to the discounted reward of the terminal state. For each analysis we choose an upper bound of 1000 search tree nodes which are allowed to be calculated during an analysis. If the bound is exceeded the analysis stops and the agent receives a negative reward of -1. A user-defined strategy generates 219 search tree nodes averaged over ten different scenes such that this upper bound seemed to be sufficient for our application domain. In this approach the reward is combined with a user-defined initialization for the Q-values. This initialization is determined by the problem-independent priorities that are assigned to every concept in the semantic network.

The second approach uses the priorities as rewards for the jump actions. Whenever such an action is selected by the agent, this action is rewarded by the priority which is assigned to the goal state of the jump. Since concepts that are close to the goal concept have a higher priority than concepts that are close to primitive concepts, this approach prefers a model-driven analysis strategy. The actions a_1 to a_6 are assessed by zero. The reward for the jump action is combined with the reward for the generated search tree nodes which is calculated according to (21). Due to the discount rate, the influence of the number of search tree nodes is higher on state action pairs that are visited at the end of an episode.

4.4 Integration into the control algorithm

The integration of the learning-methods into the ERNEST-control is done by replacing the old selection of transformations and network objects by the agent's choice of actions. As we have outlined in Section 4.3, the policy according to which the agent selects its action is determined by the Q-values of the individual state action pairs. During the learning phase these Q-values are acquired. Figure 8 shows a structure diagram for the integration of the agent based control. At the beginning of analysis the Q-values are initialized. This step is optional and only necessary if the first reward function is used for learning. Then the best search tree node v_b is selected from the OPEN set of the A*control algorithm (cf. Section 3.1). This set contains all nodes from the search tree which have not been processed, yet. If the node v_b contains a sufficiently high judged instance of the goal concept, the end of the analysis is reached. For the Monte Carlo Methods which are used in this paper to learn the optimal policy an update of the policy is done only at the end of an episode. The update corresponds to the second step of the general policy iteration scheme in (15). Therefore, this update only follows after the optimal instance for a goal concept has been found. If, however, the end of analysis has not been reached yet, two cases must be distinguished as shown in Figure 9: no optimal goal concept for C_1 has been instantiated; therefore the search tree has to be expanded further and the modified concept $M_2(C_4)$ will be instantiated next by action a_2. The two cases will become clear after the new concept of an action history has been introduced.

In Figure 8 we consider a control which uses the same sequence of transformations for all paths in the search tree (cf. Section 3.2). Therefore, in the first case the environment corresponds

given : a set of goal concepts $C_{g_i},\ i = 1, \ldots, n$, OPEN $:= \emptyset$
/$*$ *optional initialization of Q-values* $*$/
calculate for each state s_i the Q-values of the jump actions $a_7^j, j \neq i$ using the problem independent priorities
choose a network object as start state and generate a search tree node for this element
add the new search tree node to OPEN
/$*$ *control as search in the search tree* $*$/
WHILE OPEN$\neq \emptyset$
- remove the best node v_b from OPEN
- IF: if the network objects in v_b fulfill an analysis goal
- THEN: STOP with SUCCESS /$*$ *interpretation in node* v_b $*$/
- update policy based on rewards
- set optimal node v_b as agent's environment
  - IF: complete action history has been processed
  - THEN: agent selects next action
  - ELSE: select next action from action history
  - apply action to the current network object
- UNTIL action leads to new search tree nodes
- add new search tree nodes to OPEN

STOP with ERROR; /$*$ *data not interpreted* $*$/

Figure 8: Integration of reinforcement learning into the ERNEST-control

to a search tree node which contains the longest sequence of state action pairs in the whole search tree. Then the next action is chosen by the agent. In the following, the sequence of actions which led to the current network object is called *action history*. Continuing with our example in Figure 9, the first case can be found in the upper path v_2, v_4, v_5. In the second case, the action history has not been completely processed. Thus, the next action is taken from the action history. If we expand v_3 in Figure 9, we choose a_7 followed by a_2 as we have done before in the upper part.

If after execution of the selected action new search tree nodes are generated, the control gets back to the A*-search after adding the new nodes to the OPEN-set. After the A*-search has selected the optimal nodes from the OPEN-set, a new control cycle is executed.

In order to get samples for Monte Carlo methods, we perform many complete instatiations of the semantic network, i.e., we do complete analyses. Each analysis corresponds to one episode of iterative scheme in (14). The Q is updated in (15) by the reward which uses the total number of search tree nodes generated during the episode.

If a semantic network contains concepts that model different occurrences of one term with modalities, we cannot use the same policy on all paths in the search tree. This would lead to serious problems if a concept contains a different number of links for different modalities. Therefore, the next action is always selected by the agent and the agent is stored for each path of the search tree. If a backtracking is performed by the A*-search, this leads back to the old agent. However, this approach is less efficient, since each agent holds several data structures and these structures have to be copied for each new path in the search tree. Since many iterations have to be performed to estimate the Q-values properly, we use this approach only for networks that

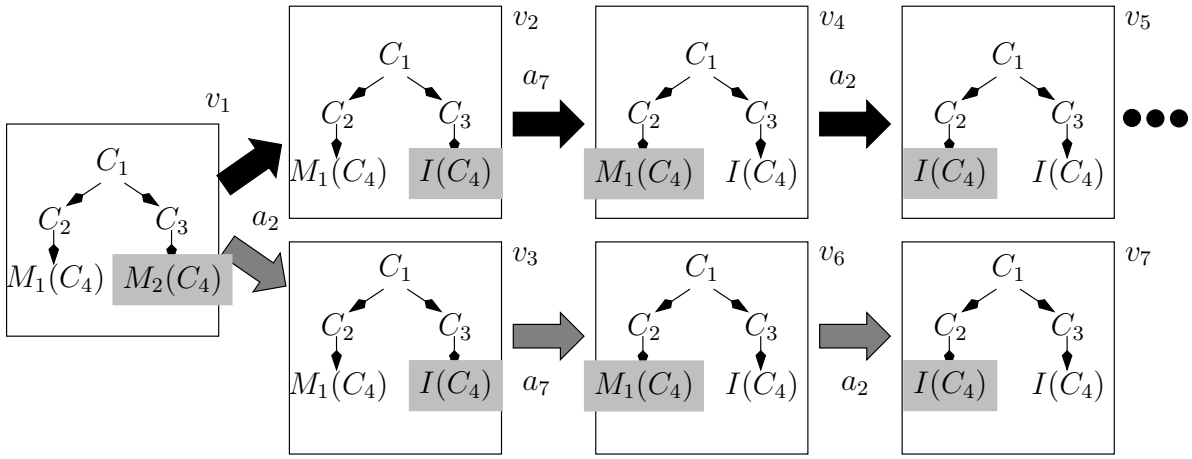

Figure 9: Agent and A^* control

include concepts with modalities.

5 Scene exploration based on semantic networks

The application domain chosen here is the exploration of arbitrary office scenes [APN00]. Since the main contribution of this paper is the conceptional work regarding the learning of analysis strategies and the integration of camera actions into a semantic network, the object-recognition task of the system is simplified: we consider six different office tools and a desk. The tools comprise a red hole punch, a red gluestick, a red adhesive tape dispenser, a blue stapler, a blue flowerpot, and a yellow text marker. These objects need not be visible in an image taken with the initial camera set-up. The office tools are modelled by 2-d regions for a typical aspect of the object in a stable position such that features like the region's height, and width as well as the region's color can be used for recognition. The 2-d object models can easily be substituted by more sophisticated ones, for example by a standard object recognizer like the one presented in [MN97, Low87]. Additionally, the knowledge base can be easily extended by concepts for other office tools due to the modularity of the concept-centered representation.

As we explained in Section 2, semantic networks in ERNEST represent declarative and procedural knowledge. While the declarative knowledge determines the structure of the knowledge base with its concepts, attributes, relations, and links, the procedural knowledge corresponds to image analysis methods or methods for camera movements. Both parts of the semantic network are outlined in the following for the considered application.

5.1 Declarative knowledge

The structure of the knowledge base for our application domain is shown in Figure 10. The gray ovals correspond to concepts that represent information about the scene. This set of concepts includes the concepts for the office tools and the table. These concepts form the parts of the concept "office scene". On the segmentation level, all office tools are modelled as color regions in 2-d while the table is substantiated by the concept "point set" which represents a set of 3-d points. The latter concept and the concept "color region" live on a lower level of abstraction

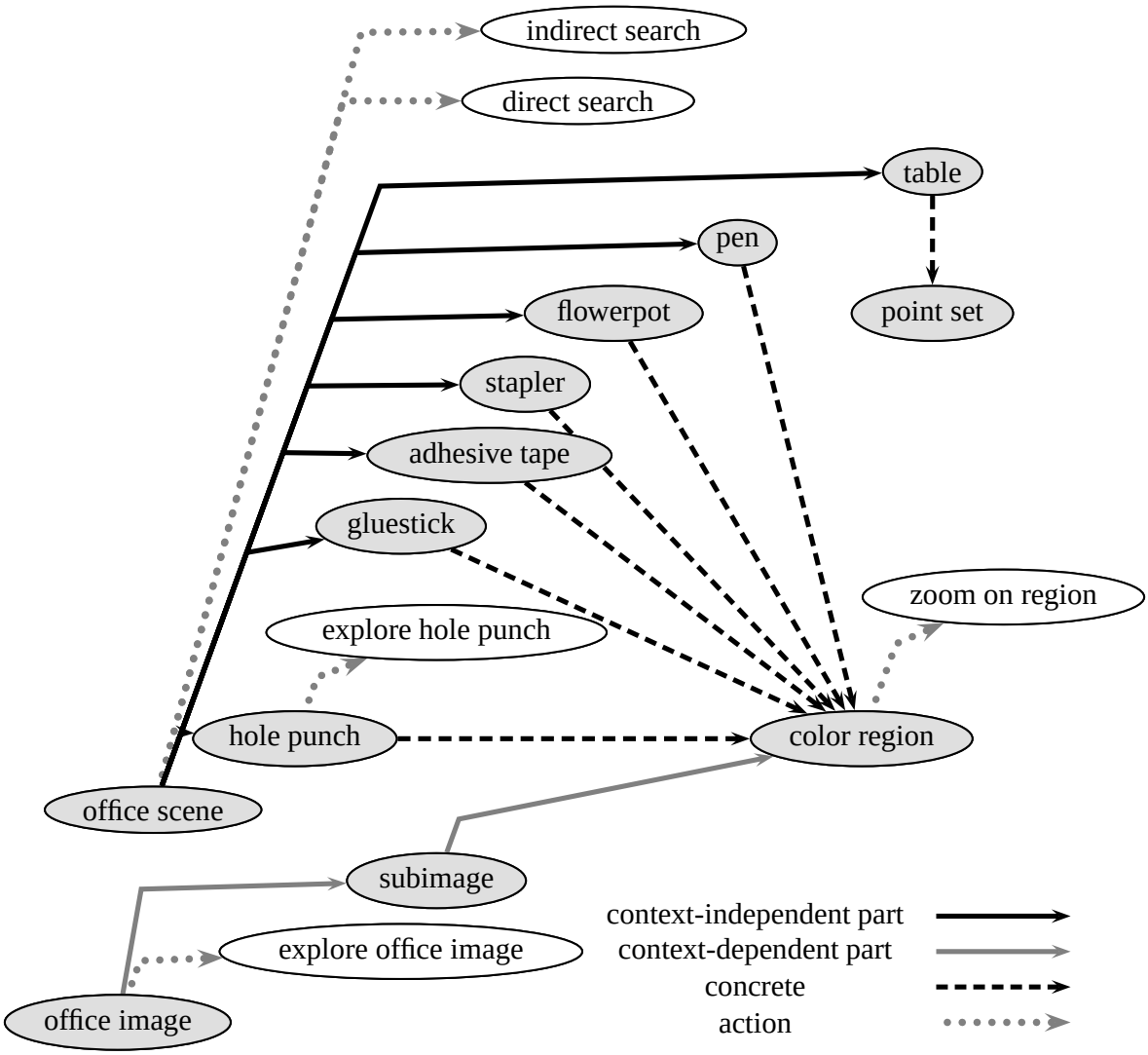

Figure 10: Knowledge base with integrated camera actions

than concepts that belong to the office tools. Therefore, they are integrated into the semantic network by a concrete link, e.g. the concept “hole punch” is connected to the concept “color region” using a concrete link.

Since one of our intentions was to integrate camera actions into the semantic network, we represent also information about images and image acquisition explicitly within the knowledge base. The concepts “office image” and “subimage” contain this knowledge. Instances of the concept “office image” store a so-called *overview image* which is taken with a small focal length. Using this image, hypotheses for the positions of the searched objects can be calculated using histogram backprojection (cf. Section 5.2). Since the concept “subimage” is represented as context-dependent part of the concept “color image”, the semantics of this link guarantees that a (partial) instance of the concept color image is calculated before the instance of the concept “subimage” is generated. Thus, an overview image is always available before the hypotheses generation takes place. The same principle is used for the connection of subimages and color regions. Since we first need a subimage to apply a region segmentation, the concept “color region” is connected to the concept “subimage” by a context-dependent part link.

An important aspect of our approach is that the knowledge base unifies the representation of objects and their relations and the representation of camera actions. These actions are also represented by concepts and integrated into the knowledge base. For this integration we introduced a new link into the ERNEST-formalism, which mirrors the semantics of the context-dependent part link. The idea is that a selection of a camera action is always done based on the state of analysis. For example, if all objects have already been found, no other part of the scene has to be examined. However, this state has to be available first to decide on the camera action. This dependency is modelled by the *action* link. Although the semantics of the link coincides with the semantics of the context-dependent part link, the new link is needed to make the knowledge representation more acessible and readable. For example, the camera action "zoom on region" is no part of the concept "color region", but an action that refers to the segmented color regions.

The camera actions can be found on different levels of abstraction within the semantic network. On the highest level of abstraction one can find camera actions which are equivalent to search procedures and which are used to find objects in a scene. The first example is the concept "direct search". Each computation of this concept calculates a new pan angle and a new tilt angle for the camera in such a way, that a new part of the scene becomes visible and a new overview image can be taken. The second example is the concept "indirect search". This concept represents an indirect search [Wix94], i.e. the search for an object using an intermediate object, e.g. in order to find a punch we first find a table and then search for the punch on it. The "indirect search" also leads to a change in the pan angle and the tilt angle, but additionally it adjusts the camera's focal length such that the resolution of the resulting image is increased.

On the lower level of abstraction in Figure 10, the camera action "zoom on region" can be found. The effect of this action is the fovealization of regions that are hypotheses for the objects. If the region that belongs to a hypothesis is too small to be reliably recognized, i.e. the height and width of the particular object cannot be determined reliably (cf. Section 5.2), we use the fovealization to get more detailed information about it. We refer to images captured after fovealization as *close-up views*. Additionally, the camera action "explore office image" is modelled on the lower level. The criterion for its selection is the number of hypotheses which the hypotheses generation yields. If no hypothesis is found, immediately a new part of the scene is considered by adjusting the pan and the tilt angle of the camera; a new overview image is then acquired.

The *computation* of a camera action concept leads to the selection of new camera parameters or the *performance* of a camera action. Thus, only one camera action concept can be computed at once. Additionally, the control has to decide if a camera action should be performed at all. In order to represent competing camera actions, i.e. actions which cannot be performed at the same time, we make use of modalities. Remember, that modalities have been introduced to represent different concurrent realizations of a concept, such as a chair with or without arm rests or with varying number of legs (cf. Section 2). Transfered to the representation of camera actions we get the following combinations: For example, the concept "office scene" constans actions represented as the concepts "direct search" and "indirect search", each of them is represented in one modality of "office scene". These modalities also contain all links to the concepts which represent the office tools and the table. A third modality of "office scene" does not include a camera action, but just the links to scene concepts. Therefore, there is one realization of "office scene" with "direct search" as an action, another realization with "indirect search", and a third without any camera action. The same holds for the concepts "color region" and "office image". Both concepts hold two modalities — one which contains the corresponding camera action, and one which does not include any camera action.

Figure 11: Typical office scene. The small images show the close-up views for the hypotheses.

During analysis, these ambiguities arising from modalities are resolved by the control algorithm and instances are calculated for each modality in a concept. These instances are judged differently where the judgment determines which action should be preferred or if a camera action should be performed. The judgments specify the *utility* of a camera actions (Section 5.2). Based on these judgments the camera action which is optimal with respect to the criterion defined by the judgment functions can be selected by the control algorithm.

5.2 Procedural Knowledge

The functions for the computation of the attributes and relations of a concept and the judgment of the corresponding instance form the *procedural* knowledge of the network.

Image analysis The image processsing part of the scene exploration system splits into several subtasks. First, hypotheses for the object location have to be determined in the overview images. This is done by histogram-backprojection [SB91] where histograms of red, blue, and yellow objects are learned before analysis. 50.000 experiments have been performed to select the optimal color space and normalization method to hypothesize the objects [PCN98]. Using the resulting hypotheses, subimages are extracted from the overview image where the hypotheses' center corresponds to the center of the subimages. The subimages are represented by the concept "subimage". The processing of the subimages depends on the resolution of the image. If the subimage was cut out of the overview images, hypotheses from the histogram backprojection are segmented as regions. However, if the region corresponding to a hypothesis is too small to be reliably verified, a fovealization of this image region is performed. This is done by instantiating the concept "zoom on region". Afterwards, a new image is captured and stored in the instance of "subimage". The close-up view is segmented by a split and merge approach [DJ93]. In both cases, the segmented regions are used to instantiate the concept "color region". In addition to the region representation, this concept contains attributes for the region's height and width as well as for the color of a region. After a normalization of the pose of a region within the image plane these features are determined and used for object recognition. To find the best matching region, judgments are needed that assess how good a segmented region fits to the represented expectations in the knowledge base. These judgments are outlined in the following.

Management of uncertainty A management of uncertainty is provided by the judgment functions for attributes, relations, and concepts within the semantic network. To rate the instances for concepts that represent information about office tools and the table, probabilities are used. This applies also for instances of the concept "color region" that are assessed according to restrictions from the office tool concepts.

The judgment of the instance $I(C_k)$ related to the concept C_k subsumes the judgments of the concepts' components, i.e. the judgments of attributes $I(C_k).A^{sn*}$, relations $I(C_k).R^{sn*}$, parts $I(C_k).P^* = I(C_k).P^*_{ci} \cup I(C_k).P^*_{cd}$ and concretes $I(C_k).K^*$:

$$\begin{aligned} & p(I(C_k)|I(C_k).A^{sn*}, I(C_k).R^{sn*}, I(C_k).P^*, I(C_k).K^*) \;\; = \\ & \frac{1}{\eta} \cdot p(I(C_k)) p(I(C_k).A^{sn*}|I(C_k)) p(I(C_k).R^{sn*}|I(C_k)) \cdot \\ & \qquad p(I(C_k).P^*|I(C_k)) p(I(C_k).K^*|I(C_k)) \end{aligned} \tag{22}$$

where $\eta = p(I(C_k).A^{sn*}, I(C_k).R^{sn*}, I(C_k).P^*, I(C_k).K^*)$ denotes the normalization factor that is irrevelevant for maximization. We assume that the individual distributions are pairwise independent and $p(I(C_k))$ is uniformly distributed. Therefore, we base the judgment of an instance on the judgments of the constituents of the corresponding concepts. Since these constituents vary for different concepts, the judgment functions also differ between concepts. However, in order to rate the individual attributes, we chose a uniform approach. Each attribute is judged according to a normal distribution with parameters that have been estimated off-line using a classified set of samples with 40 images for each search object. During analysis values for the attributes are calculated and judged according to the corresponding distribution. For example, we get for the judgment of the hole punch by

$$\begin{aligned} p(I(\text{hole punch})|I(\text{hole punch}).A^{sn*}) \;\; \sim \;\; & \exp\left(\frac{(x_{\text{width}} - \mu_{\text{width}})^2}{-2\sigma^2_{\text{width}}}\right) \cdot \\ & \exp\left(\frac{(x_{\text{height}} - \mu_{\text{height}})^2}{-2\sigma^2_{\text{height}}}\right) \cdot \\ & \exp\left(\frac{(\boldsymbol{x}_{\text{color}} - \boldsymbol{\mu}_{\text{color}})^2}{-2\sigma^2_{\text{color}}}\right), \end{aligned} \tag{23}$$

where $\mu_{\text{width}}, \mu_{\text{height}}$, and $\boldsymbol{\mu}_{\text{color}}$ correspond to restrictions for the attribute values. To assess instances of the concept "color region", these restrictions are propagated during analysis from the concepts that represent office tools. All instances of concepts that store information about office tools are rated according to (23).

The judgment of instances of the concept "office scene" is given by

$$p(I(\text{office scene})|I(\text{office scene}).P^*) = \prod_{P_j \in \Psi_P(C_k, H_i)} p(P_j|I(C_k)), \tag{24}$$

i.e. it is determined by all the judgments of all parts that are in the current modality H_i. The corresponding parts are selected by the function $\Psi_P(C_k, H_i)$. Instances of the concepts "office image" and "subimage" are assessed by the value one, since they contain uncertain information. Since we need to provide optimistic estimates to guarantee the admissibility of the A*graph search this is also the estimate for every concept that has not been instantiated, yet, in the considered search tree node.

The judgment of instances for concepts that contain information about the scene is used to derive the judgment of camera actions. As we have already explained in Section 5.1, camera

actions are selected according to the current state of analysis. The goal of the camera actions' performance is to provide more information about the scene and reduce the uncertainty of intermediate results. Therefore, the judgments have to reflect if new information is needed and which camera action yields the information with lowest cost. We suggest that *utilities* are a good means to describe these situations [Jen96].

Since we integrated different camera actions into the semantic network, also the judgment functions for the corresponding concepts differ substantially. Therefore, we consider each different type of action separately in the following.

Direct and indirect search For these two camera actions the utility measure relies on the evidence for the searched office tools and for the table. The judgments of the instances that are associated to these objects reflect if an object has already been found. For each instance that is associated with one of the six searched objects or the table we have a hypothesis with states *object found* and *object not found*. Depending on these states the optimal camera action is chosen. The utilities are calculated using a utility table which contains the utility of an action a provided that the hypothesis is in state h, where a belongs to the set of executable actions and h is a state of the random variable H. In general, just the distribution of H is known. Therefore, we can only compute the mean utility

$$EU(a|e) = \sum_{h \in H} U(a,h)p(h|e)\,. \tag{25}$$

The variable e denotes the evidence which arises from the intermediate results of analysis, U refers to the utility of an action a in state h. The control algorithm chooses the action which maximizes the mean utility.

To represent the hypothesis for the selection of the two different camera actions "indirect_search" and "direct_search", we define the vector

$$\begin{aligned} \boldsymbol{h}_{\text{officescene}} \;=\; & (\delta(I(C_A)), \delta(I(C_B)), \delta(I(C_H)), \delta(I(C_K)), \delta(I(C_L)), \\ & \delta(I(C_S)), \delta(I(C_T)))\,. \end{aligned} \tag{26}$$

as hypothesis H. Hence, the states of H are all configurations of this vector which describe if instances of the adhesive tape $I(C_A)$, flower pot $I(C_B)$, stapler $I(C_H)$, gluestick $I(C_K)$, hole punch $I(C_L)$, pen $I(C_S)$, and table $I(C_T)$ are available where δ is an indictor function telling whether the instance given as an argument exists. Within this vector, 1 denotes "object found", and 0 denotes "object not found". For example, if $\boldsymbol{h}_{\text{officescene}} = (1,1,1,1,1,1,1)$ all objects have been found. To estimate the probability of this hypothesis, we define

$$p(\delta(I(C_i))) = \begin{cases} p(I(C_i)|I(C_i).A^{sn*}, I(C_i).R^{sn*}, \\ \qquad I(C_i).P^*, I(C_i).K^*) & \text{if} \quad \delta(I(C_i)) = 1 \\ 1 - p(I(C_i)|I(C_i).A^{sn*}, I(C_i).R^{sn*}, \\ \qquad I(C_i).P^*, I(C_i).K^*) & \text{if} \quad \delta(I(C_i)) = 0 \end{cases} \tag{27}$$

Therefore, the evidences which are known from the current search tree directly influence the value of $p(\delta(I(C_i)))$. Since we assume mutual independency between the different $p(\delta(I(C_i)))$ the overall value is determined by multiplication of $p(\delta(I(C_i)))$.

We used just 0 and 1 as utilities. Every time when a table has been found the indirect search gets the utility 1, while the utility of the direct search becomes 0. If the hypothesis assumes that no table has been detected in the data, the utilities are vice versa. Finally, the judgment of the

table is decisive if an indirect or a direct search is performed. This would change if the utilities are chosen differently. There is only one case in which no camera action is performed, i.e. if all search objects have been found.

Zoom on region For the action "zoom_on_region" we use a hand-crafted utility function based on the region's size and the current zoom setting. This function takes into account that the utility of a fovealization increases if the size of the hypothesized regions is small while it decreases if the region is relatively large. Therefore, we define for the size x of a region

$$g(x) = \exp\left(\frac{-x}{\alpha}\right) . \tag{28}$$

where α denotes a normalization factor. This factor was determined heuristically and describes a measurement for the number of pixels inside a bounding box that encloses a hypothesized region. The utility for no fovealization is $1 - \exp(\frac{-x}{\alpha})$. If the hypothesis does not correspond to a valid region the entries of the utility table are defined as follows: The utility for a fovealization is zero, while the utility for no camera action is 0.1. The latter was also determined heuristically and has the effect that a fovealization is preferred over the execution of no camera action.

In addition to the utility measure we need the probability for the hypothesis to determine the expected utility of the fovealization. We define

$$\boldsymbol{h}_{\text{colorregion}} = (\delta(I(C_{\text{colorregion}}))) , \tag{29}$$

where the probability for this hypothesis is given according to (27). Using the utilities given below the expected utility is calculated according to (25).

Explore office image In contrast to the camera actions considered in the paragraphs above the camera action "explore office image" is assessed without probabilities. The decision for this action is solely based on the number of hypotheses found. The camera action gets a judgment of 1 if no hypotheses were detected. A judgment of 0 is provided if hypotheses are available.

6 Experiments

The theoretical aspects of the proposed approach concerning knowledge representation and learning of analysis strategies were outlined in the preceeding sections. In this section we show the performance of the system in different experimental set-ups. First, we have a look at the detection rates and the recognition rates for the objects. For example, these are dependent on the used color space for histogram backprojection. Second, we describe experiments to demonstrate the influence of the learned analysis strategies on the efficiency of the system.

6.1 Experiments using a user-defined strategy

The experiments outlined in the following are all based on an ERNEST-control algorithm that uses a user-defined analysis strategy. This strategy

$$\begin{aligned} &\text{office image} \xrightarrow{a_7} \text{office scene} \xrightarrow{a_7} M(\text{flower pot}) \xrightarrow{a_7} M_1(\text{color region}) \xrightarrow{a_7} \\ &M_1(\text{subimage}) \xrightarrow{a_7} M_1(\text{color region}) \xrightarrow{a_7} M(\text{flower pot}) \xrightarrow{a_7} \\ &M(\text{hole punch}) \xrightarrow{a_7} \dots \end{aligned} \tag{30}$$

Figure 12: Overview images for the two different office scenes; on the left office 1 is depicted, on the right office 2 is shown.

models a model-driven expansion of the network which is followed by an instantiation of the expanded network. The action a_7 represents the jump action introduced in Figure 6. All other concepts that represent information about office tools are instantiated in the way given in (30). After instantiating the concept “hole punch” the calculation of instances of the concepts “glue-stick”, “stapler”, “pen”, and “adhesive tape” follows.

All experiments were performed in two different office environments that are shown in Figure 12. In toteal we considered 70 different scenes in each office where 35 scenes belong to office 1 and 35 to office 2. In each scene the positions of the objects are changed. In 25 of the 35 scenes all objects were included in the first overview image; therefore, no scene exploration using a direct or indirect search had to be done for those experiments.

Using these 50 scenes we investigated the performance of the hypotheses generation. In Figure 13 one result for histogram backprojection is shown. The original image is given in Figure 12 (left). We used the intensity normalized rg color space to represent the colors of the red and blue objects. The histogram of the yellow object is learned in the RGB color space. The interest map shown in Figure 13 depicts the fusion of the three backprojection results. Black areas correspond to a high match of the colors from the histogram and the image area. For each overview image we select the twelve best hypotheses that are verified in the following using the a-priori knowledge about the objects. The selection is based on the mean gray-value of the region which corresponds to a hypothesis in the backprojected image. In Figure 13 (right) the best twelve hypotheses for Figure 12 (left) are marked by crosses.

Quantitative results are given in Table 1 where the number of correctly generated hypotheses for each object in office 1 and office 2 is shown. We obtained these results by manually labeling all overview images for the two offices. The labels contain information about the 2D position of the objects that the system has to search for. These 2D positions for the objects were compared to the center of gravity of the generated hypotheses. If the center of gravity of a hypothesis coincides with a labeled 2D position of an object we call the hypothesis a correct hypothesis for this object. The hypotheses do not contain any information about the class of the corresponding object, this information is provided by the labeling process.

Since we captured 25 overview images in each office, we should find 25 hypotheses for each object in each office. The hypothesis generation yields the best results for the adhesive tape dispenser. In 96 % of all 50 scenes a hypothesis was correctly generated for this object.

Figure 13: Result of histogram backprojection

	hole punch red	adhesive tape red	gluestick red	stapler blue	pen yellow	flower pot blue	others
office 1	22	24	18	22	24	23	6.2
office 2	23	24	13	17	22	24	5.4
total	90 %	**96 %**	62 %	78 %	92 %	94 %	-

Table 1: Number of hypotheses generated in office 1 and office 2. In each office 25 experiments were performed such that the maximal number of generated hypotheses would be 25 for each object.

Problems occurred for the gluestick. Since this object is very small the mean gray-value was often lower than the value of the twelve best hypotheses. Overally we detected 85.3 % of all possible hypotheses within the 50 scenes. This is an important result since objects that are not found during the hypotheses generation are lost for later verification steps.

Depending on the size of the region that belongs to a hypothesis in the backprojected image, a fovealization is performed as was outlined in Section 5.1. The resulting close-up views are segmented using a split and merge approach. Results for this segmentation are depicted in Figure 14. The approach shows good visual results. However, a few segmentation errors are visible, e.g. the gluestick is split into three regions and the flower pot is merged with a little piece of background. On average 14 regions are generated for each close-up view.

If no fovealization is performed, the regions in the backprojected image are directly given to the verification step. In the following, we compare three different systems that handle the fovealization differently. The first system (system 1) uses no fovealization at all, but classifies the regions from the backprojected images. System 2 and system 3 both fovealize hypotheses. However, system 3 uses just regions which are segments of close-up views while system 2 decides dynamically on a region's fovealization. If a fovealization is performed, the close-up view is segmented and the computed regions are subject to the verification step, else the regions of the backprojected image are verified.

In Table 2 the recognition rates of system 1 for the six searched objects are shown. The recognition module classifies the generated hypotheses in one of the six object classses. For the evaluation of the performance of the system, the label information was used again. It appears

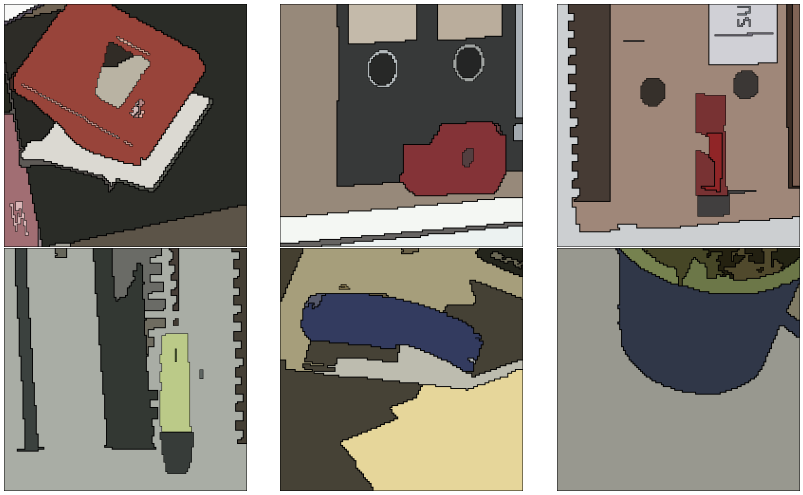

Figure 14: Results for the region segmentation using split and merge

	hole punch	adhesive tape	gluestick	stapler	pen	flower pot
office 1	18	15	10	15	15	12
office 2	18	18	6	4	19	13
total	72 %	66 %	32 %	38 %	68 %	50 %

Table 2: Recognition rates for the six searched office tools

that the hole punch was recognized correctly in 18 scenes of office 1 and office 2, respectively. For all other objects the number of recognized objects decreases. Major problems occurred for the gluestick and the stapler in office 2. Like the hypotheses generation the recognition of the gluestick is poor due to its small size. The recognition of the stapler is complicated by color shifts from blue to black in the overview images. Therefore, the region that forms the hypothesis for a stapler is usually smaller than expected and the stapler is very often mixed with parts of the background.

On average, we achieved a recognition rate of 54.3 %. The recognition rates for the single objects are given in Table 2. Like the total recognition rate these rates do not take into account if a hypothesis has been generated for an object during histogram backprojection. However, if no hypothesis has been generated for an object in an overview image, a correct recognition of this object is not possible since the recognition process is based on the hypotheses generation. If we also consider the generated hypotheses for the objects the total recognition rate increases to 64.7 %. With respect to the highly structured background in the 50 overview images this is a very good result.

Evaluating system 2 on the same 50 scenes leads to a slightly improved recognition rate. The mayor drawback of this approach is that objects that are recognized correctly based on the overview image are lost during the verification step using the close up views. On the one hand, system 2 found 23 objects that system 1 misclassified, but on the other hand it lost 18 objects.

The loss of the objects has different reasons. Errors in the depth estimation lead to wrong feature values or wrong calculations of the focal length for the close-up views. Incorrect regions, that are always present in image segemtation, cause feature values that are not compatible with

the expected feature values for the objects.

The influence of segmentation errors and errors in the depth estimation becomes also apparent by the evaluation of third system's performance. This system uses the regions of close-up views for the verification of the objects. The semantic network for system 3 does not contain the information about the image acquisition, but it has the concept "color region" as primitive concept. Instances of this concept are generated for every segmented region and the best matching region is selected according to the regions' judgment. Using system 3 just 34 % of all objects are assigned to the correct region in the close-up views.

6.2 Experiments based on learned analysis strategies

Besides the complexity of the application domain which is, for example, reflected in the number of generated hypotheses for the searched objects, the analysis strategy has an important impact on the efficiency of a knowledge based system. Dependening on this strategy the number of generated search tree nodes can vary between 194 for a model-driven strategy and 1000 for a data-driven strategy. This evaluation was done using a part of the semantic network depicted in Figure 10. The part included all scene concepts (gray ovals) without the concepts "table" and "point set". The model-driven strategy coincides with the user-defined strategy we used in all experiments of the preceeding section. The two strategies were evaluated using ten different overview images which contained all searched objects. No camera action was performed by this system.

All learned policies are evaluated refering to the search tree nodes that are generated by the user-defined strategy. In the following, we compare the two different kind of rewards which were introduced in Section 4.3. First, we have a look at the policy which are learned by the agent based on initialized Q-values and a reward that takes just the number of search tree nodes into account (cf. (21)). As we outlined in Section 4.3, the initialization is done based on the problem-independent priorities. In Figure 15 results are shown where eleven different policies $\pi_1, \ldots, \pi_{10}$, and π_{all} were evaluated on the above mentioned ten different scenes. Before evaluation, the agent learned the different strategies where image j was used to learn policy π_j. The learning state ended after 8000 iterations. The policy π_j was evaluated on all ten different scenes. For each scene a mean value for the generated search tree nodes and a minimal value was calculated by averaging the generated search tree nodes when every network object of the expanded network was once used as start state. The minimal value corresponds to the minimal number of search tree nodes where this minimum is calculated using all network objects of the expanded network as start states. The mean and minimal value for all ten scenes were averaged. These averaged values are depicted in Figure 15. Additionally, the agent learned a policy using all ten different overview images. Here, in each iteration the analysed overview image changed. Figure 15 reveals that for every policy π_j the average of minimal search tree nodes is lower than 194. On average between 75 and 78 search tree nodes are calculated. The average of the mean number of search tree nodes lies for two policies below the number of search tree nodes generated by the user-defined strategy. This shows that the generalization capabilities for a policy is strongly dependent on the scene which was used for learning. The policy that was acquired using all ten scenes generates 224 search tree nodes on average. Therefore, the generalization capabilities of the policies increase if different scenes are used for learning as it was expected.

Besides the generalization capability the convergence is decisive if a learning method is evaluated. The convergence for the learning of policy $\pi_1, \ldots, \pi_5$ is shown in Figure 16 for the mean number of search tree nodes. For the first 4000 iterations the mean number of search

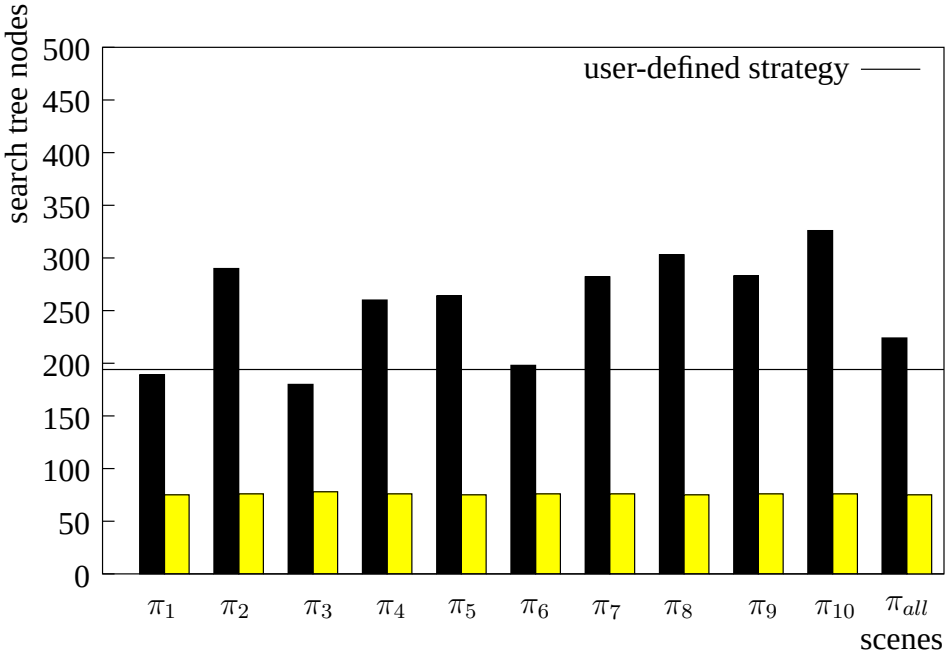

Figure 15: Number of search tree nodes generated on average for ten different policies $\pi_1, \ldots, \pi_{10}$; the black bars depict the average for the mean number of search tree nodes, the white bars refer to the minimal number of search tree nodes.

tree nodes are depicted for every hundred iterations, then every fivehundred iterations the mean number is considered. The mean number of search tree nodes decreases for all policies if the number of iterations increases.

Using the second approach to reward the actions of the agent during the learning phase, we achieved the results shown in Figure 17. The approach does not use any initialization, but it rewards jump actions based on priorities and a whole sequence of states and actions using the number of search tree nodes. The experiments revealed that the generalization capabilities are worse in comparison to the other reward function. As one can see from Figure 17, the average of the mean number of search tree nodes is larger than 194 for all learned policies. Even the average of the minimal number lies below 194 for policy π_2 or π_5. However, if all ten overview images are used, we get an average of 294 search tree nodes. This supports the expectation that the generalization capabilities increase if more data is available for learning.

Although, the results are worse than the results of the first approach this approach has more the nature of typical reinforcement learning frameworks. For example, we have to take into account that no initialization is performed. As basic experiments revealed, one can achieve fairly good results on some scenes if just the initialized Q-values are used to determine the optimal policy. However, this is not the case in general. Additionally, the second approach stopped after 2000 iterations. This is caused by the exploration factor we used during the learning phase. For the first approach we chose $\epsilon = 0.05$ while in the second we started with $\epsilon = 0.7$. This factor was reduced during the iterations such that is was 0.01 after 2000 iterations.

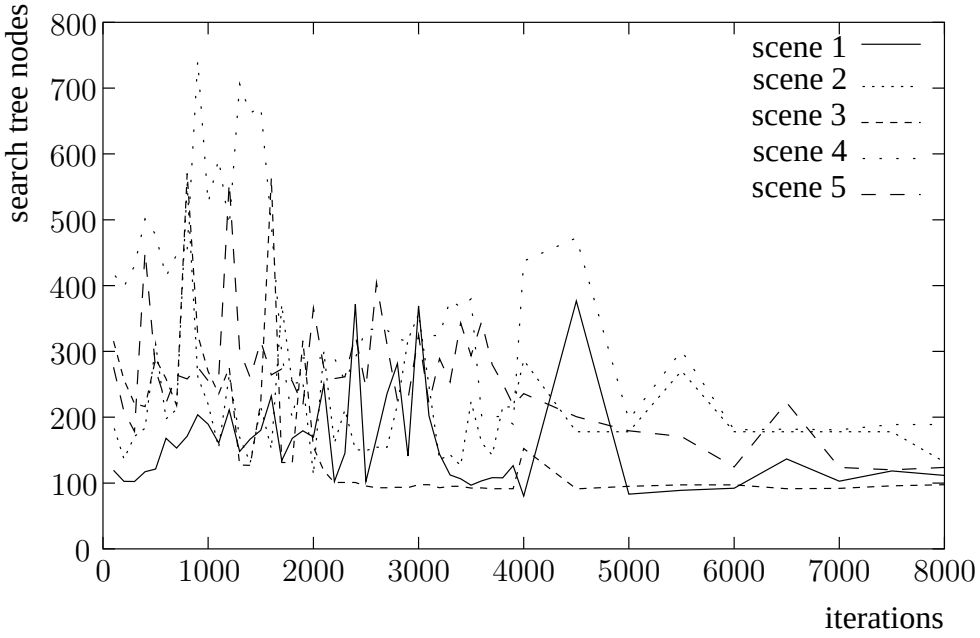

Figure 16: Mean number of search tree nodes evaluated on the ten scenes after a different number of iterations

7 Conclusion and Outlook

In this paper, we described a knowledge based system that is used for scene exploration. The main focus was the learning of analysis strategies on the one hand and the integration of camera actions into a semantic network on the other hand. The experiments showed the effectiveness and efficiency of the approach. In 50 office scenes which show the searched object in a highly structured background we recognized 54.3% of the objects correctly.

The evaluation of the reinforcement learning approach to acquire the analysis strategies showed that even a simple optimization criterion like the number of search tree nodes results that are comparable to a handcrafted strategie. Therefore, the time consuming procedure to define such a strategie can be replaced by the learning approach.

References

[APN00] U. Ahlrichs, D. Paulus, and H. Niemann. Integrating aspects of active vision into a knowledge-based system. In *Proceedings of the* 15^{th} *International Conference on Pattern Recognition (ICPR)*, pages IV:579–582, Barcelona, Spain, 2000. IEEE Computer Society Press.

[BDB99] J. Bins B. Draper and K. Baek. Adore: Adaptive object recognition. In H. Christensen, editor, *Computer Vision Systems*, pages 522–537, Heidelberg, Jan. 1999. Springer.

[BFKS99] C. Bauckhage, J. Fritsch, F. Kummert, and G. Sagerer. Towards a Vision System for Supervising Assembly Processes. In *Proceedings of the Symposium on Intelligent Robotic Systems (SIRS'99)*, pages 89–98, Coimbra, Portugal, 1999.

[BGL85] R.J. Brachman, V. Gilbert, and H. Levesque. An essential hybrid reasoning system: Knowledge and symbol level accounts of krypton. pages 532–539, Los Angeles, Calif., 1985.

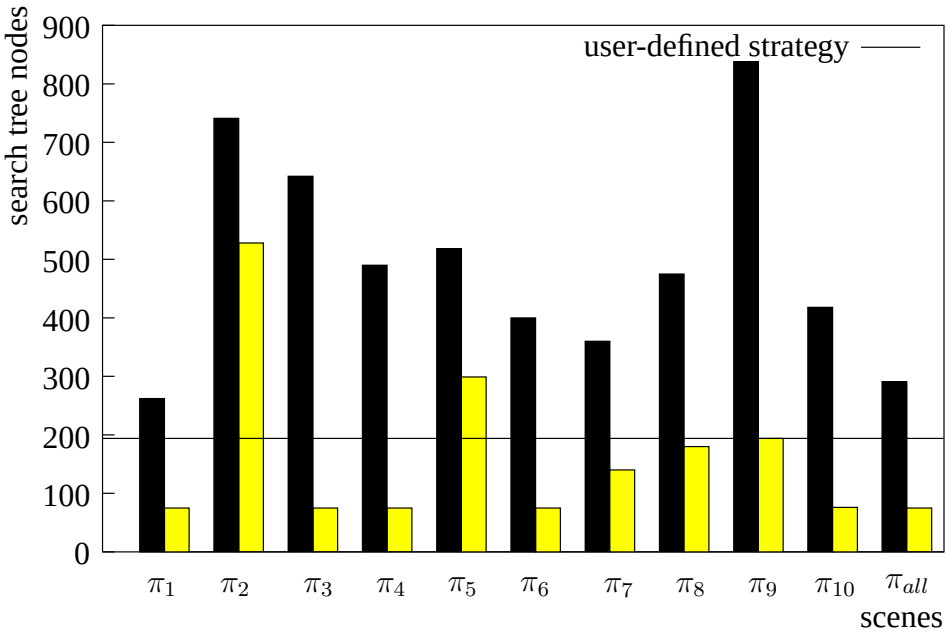

Figure 17: Number of search tree nodes generated on average for ten different policies $\pi_1, \ldots, \pi_{10}$; the black bars depict the average for the mean number of search tree nodes, the white bars refer to the minimal number of search tree nodes.

[BH96] U. Büker and G. Hartmann. Knowledge based view control of a neural 3-D object recognition system. In *International Conference on Pattern Recognition*, volume IV, pages 24–29, Los Alamitos, Kalifornien, 1996. IEEE Computer Society Press.

[Bra77] R. Brachman. What's in a concept: Structural foundations for semantic networks. Technical report, Bolt, Beranek and Newman, Cambridge, Mass., 1977.

[Bro81] R. Brooks. Symbolic reasoning about 3-d models and 2-d images. *Artificial Intelligence*, 17:285–348, 1981.

[CL97] D. Crevier and R. Lepage. Knowledge-based image understanding systems: A survey. *Computer Vision and Image Understanding*, 67(2):161–185, August 1997.

[DAP01] B. Draper, U. Ahlrichs, and D. Paulus. Adapting object recognition across domains: A demonstration. In *International Workshop on Computer Vision Systems*, 2001. to appear.

[DC97a] S. Dance and T. Caelli. SOO-PIN: Picture Interpretation Networks. In T. Caelli and W.F. Bishof, editors, *Machine Learning and Image Interpretation*, Advances in Computer Vision and Machine Intelligence, pages 225–254. Plenum Publishing Cooperation, New York, 1997.

[DC97b] C. Dillon and T. Caelli. Cite — Scene Understanding and Object Recognition. In T. Caelli and W.F. Bishof, editors, *Machine Learning and Image Interpretation*, Advances in Computer Vision and Machine Intelligence, pages 119–187. Plenum Publishing Cooperation, New York, 1997.

[DCB+89] B.A. Draper, R.T. Collins, J. Brolio, A.R. Hanson, and E.M. Riseman. The Schema System. *International Journal of Computer Vision*, 2:209–250, 1989.

[DCL96] S. Dance, T. Caelli, and Z. Liu. A Concurrent, Hierachical Approach to Symbolic Dynamic Scene Interpretation. *Pattern Recognition*, 29(11):1891–1903, November 1996.

[DJ93] M. Dubuisson and A. K. Jain. Object contour extracting using color and motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 471–476. IEEE Computer Society, Ney York City, 1993.

[Fre77] E. Freuder. A Computer System for Visual Recognition using Active Knowledge. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 671–677, MIT, Cambridge, Massachusetts, USA, 1977.

[HR78] A. Hanson and E. Riseman. Visions: A computer system for interpreting scenes. In A. Hanson and E. Riseman, editors, *Computer Vision Systems*, pages 303–333. Academic Press, Inc., New York, 1978.

[Jen96] F. V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, London, 1996.

[KBR86] T. Kaczmarek, R. Bates, and G. Robins. Recent developments in nikl. In *Proc. of the Nat. Conf. on Artificial Intelligence*, pages 978–985, Philadelphia, Pa., 1986.

[KI94] K. Kawamura and M. Iskarous. Trends in Service Robots for the Disabled and the Elderly. In *Intelligent Robots and Systems*, pages 1647–1654, Piscataway, NJ, 1994. IEEE Computer Society.

[KKW98] B. Krebs, B. Korn, and F.M. Wahl. A task driven 3d object recognition system using bayesian networks. In *International Conference on Computer Vision*, pages 527–532, Los Alamitos, Kalifornien, 1998.

[LM79] H. Levesque and J. Mylopoulos. A procedural semantics for semantic networks. In N. Findler, editor, *Associative Networks*, pages 93–121. Academic Press, Inc., New York, 1979.

[Low87] D.G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[MH90] T. Matsuyama and V. Hwang. *SIGMA: A Knowledge-Based Aerial Image Understanding System*. Plenum Press, New York, 1990.

[MHZR99] M. Marengoni, A. Hanson, S. Zilberstein, and E. Riseman. Control in a 3D Reconstruction System using Selective Perception. In *International Conference on Computer Vision*, volume II, pages 1229–1236, Los Alamitos, Kalifornien, 1999. IEEE Computer Society Press.

[Min75] M. Minsky. A framework for representing knowledge. In P.H. Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, New York, 1975.

[MN97] H. Murase and S. K. Nayar. Detection of 3D objects in cluttered scenes using hierarchical eigenspace. *Pattern Recognition Letters*, 18(5):375–384, 1997.

[Nie90] H. Niemann. *Pattern Analysis and Understanding*, volume 4 of *Springer Series in Information Sciences*. Springer, Heidelberg, 1990.

[Nil82] N.J. Nilsson. *Principles of Artificial Intelligence*. Springer Verlag, Berlin Heidelberg, 1982.

[NSSK90] H. Niemann, G. Sagerer, S. Schröder, and F. Kummert. ERNEST: A Semantic Network System for Pattern Understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 12(9):883–905, 1990.

[PCN98] D. Paulus, L. Csink, and H. Niemann. Color cluster rotation. In *Proceedings of the International Conference on Image Processing (ICIP)*, Chicago, October 1998. IEEE Computer Society Press.

[Pro90] H. Profitlich. Sb-one: Ein wissensrepräsentationssystem basierend auf kl-one. Technical Report Memo 43, Sonderforschungsbereich 314: Künstliche Intelligenz – Wissensbasierte Systeme, Universität des Saarlandes, Saarbrücken, 1990.

[Qui68] M. R. Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing*, pages 227–270. MIT Press, Cambridge, Mass., 1968.

[Rim93] R. Rimey. Control of Selective Perception using Bayes Nets and Decision Theory. Technical report, Department of Computer Science, College of Arts and Science, University of Rochester, Rochester, New York, 1993.

[Rob97] V. Roberto. Vision as Uncertain Knowledge. In V. Cantoni, S. Levialdi, and V. Roberto, editors, *Artificial Vision: Image Desciption, Recognition and Communication*, pages 135–160, New York, 1997. Academic Press.

[Sag85] G. Sagerer. *Darstellung und Nutzung von Expertenwissen für ein Bildanalysesystem*. Springer, Berlin, 1985.

[Sal97] R. Salzbrunn. *Wissensbasierte Erkennung und Lokalisierung von Objekten*. Shaker Verlag, Aachen, 1997.

[SB91] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, November 1991.

[SB98] R.S. Sutton and A.G. Barto. *Reinforcement Learning*. A Bradford Book, Cambridge, London, 1998.

[SN97] G. Sagerer and H. Niemann. *Semantic Networks for Understanding Scenes*. Advances in Computer Vision and Machine Intelligence. Plenum Press, New York and London, 1997.

[TMCZ80] J.K. Tsotsos, J. Mylopoulos, H.D. Covvey, and S.W. Zucker. A Framework for Visual Motion Understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2(6):563–573, 1980.

[Win94] A. Winzen. *Automatische Erzeugung dreidimensionaler Modelle für Bildanalysesysteme*, volume 89 of *Dissertationen zur künstlichen Intelligenz*. infix, St. Augustin, 1994.

[Wix94] L. Wixson. Gaze Selection for Visual Search. Technical report, Department of Computer Science, College of Arts and Science, University of Rochester, Rochester, New York, 1994.

[Woo75] W. Woods. What's in a link? foundations for semantic networks. In D. Bobrow and A. Collins, editors, *Representation and Understanding*, pages 35–82. Academic Press, New York, 1975.