Approach to Active Knowledge Based Scene Exploration

U. Ahlrichs, J. Fischer, D. Paulus, H. Niemann
Lehrstuhl für Mustererkennung (Informatik 5)
Universität Erlangen-Nürnberg
Martensstraße 3, 91058 Erlangen, Germany

This paper was submitted to es'99
December 1999, Cambridge, England

Contents

J. Fischer, U. Ahlrichs, D. Paulus, H. Niemann: *Approach to Active Knowledge Based Scene Exploration*,

Approach to Active Knowledge Based Scene Exploration

U. Ahlrichs, J. Fischer, D. Paulus, H. Niemann
Lehrstuhl für Mustererkennung (Informatik 5)
Universität Erlangen-Nürnberg
Martensstraße 3, 91058 Erlangen, Germany

Abstract

In almost every knowledge based image analysis system the knowledge base serves as model for the application domain and is used during the interpretation of the images. However, in visual based scene exploration a system needs the ability to perform camera movements, because objects may not be visible with the initial camera setting. Nevertheless, most of the known systems lack a representation of such movements which are necessary to acquire missing information. In this paper we will show how to integrate these (camera) movements into a conventional knowledge base using a semantic network formalism. To use the knowledge during the interpretation, an iterative control algorithm is applied which has any-time capabilities. This control algorithm was extended in order to deal with the actions which have been integrated into the knowledge base.

The suitability of the approach is verified by experiments in which an office scene is explored with an active camera. These experiments revealed, that in 93 % of all cases the objects were found correctly.

1 Introduction

In the near future autonomous mobile systems which use visual information will become more and more important for daily life. Examples for systems which carry out simple routine tasks like the transportation of mail, books or dishes can be found in [1]. In order to fulfill such tasks the system has to be able to interpret an image and perform actions to acquire information which is not available in the current image, but is needed for the task to be accomplished. If, for example, the robot wants to localize an object and this object is not visible in the initial camera set-up, the camera parameters have to be changed and the object localization starts with the new image. Another example is the modification of focus, if the image is blurred, or the modification of zoom, if small objects cannot be reliably recognized. The strategy which suggests these adaptations of camera parameters is called *active vision* [2]. The goal is to change the camera parameters such that they are suitable for following

processing steps, e.g. the image interpretation. The criterion of suitability has to be defined according to the task of the system.

In order to explore the environment autonomous systems use knowledge about the objects which are important for the task to be performed, and about the scene. In this paper a new approach is presented which combines the traditional representation of knowledge about objects and the scene with camera actions as suggested by the strategy of active vision. A camera action corresponds to a selection of new camera parameters or the move of the camera. The knowledge about scene information and camera actions is *uniformly* represented in *one* knowledge base using a semantic network formalism.

In classical image analysis, of course, many systems like VISIONS [3], SPAM [4] or SIGMA [5] are known which use information represented in a knowlegde base. But all these systems lack a representation of camera actions. Related work on the selection and performance of actions using Bayesian networks can be found in [6, 7]. Examples of approaches for uniform representation of actions and knowledge about the task domain are the situation calculus or decision networks, see [8, 9] for an overview or [10] for an example using Bayesian networks. We found the semantic network representation formalism particularly suitable for the description of objects [11], of medical decisions [12], and of a dialogue system [13], and prefer this formalism to a uniform description using e.g. Bayesian networks which give a less obvious object representation [10].

In order to make use of the represented knowledge about the scene, a control algorithm has to be provided. We proposed an iterative control algorithm which is particularly suitable for the application domain because of its any-time capabilities [14]. This algorithm was originally developed for pattern interpretation tasks. Here we demonstrate how we extended this control algorithm in order to handle the camera actions represented in the very same framework and allow an active scene exploration.

After an outline of the problem we deal with in section 2 we describe the knowledge representation formalism in section 3. In section 4 we explain the representation of actions. We outline the iterative control used in section 5 and its extension to actions in section 6. Finally, we demonstrate the feasibility and efficiency of our approach by means of experiments with a system for office scene exploration (section 7).

2 Problem Description

The problem which has to be solved by our system is the exploration of arbitrary office scenes with static objects, i.e. motion of objects is not modelled. The goal is to find pre–defined objects in these scenes where the location of the objects is not restricted. Since the main contribution of the approach is on the conceptional work to integrate the camera actions into the knowledge base, only a simplified task for object recognition is choosen in the experiments. Three red objects, a punch, a gluestick and an adhesive tape are used as shown below. These objects need not be visible in the initial set-up which makes it necessary to perform camera movements to search for the objects. Since these movements are time-consuming, efficient strategies are needed to choose the next camera setting or to determine if a camera movement should be performed at all. The decision where to look next relies heavily on the objects which have been found

already in the scene. This motivated the design of a knowledge base which contains both, the knowledge which is needed for identifying the objects, and the information about search strategies.

3 The Knowledge Representation Scheme

For the representation of task–specific knowledge we propose our semantic network formalism [15, 16] which provides the following types of network *nodes*:

- *Concepts* representing abstract terms, events, tasks, actions, etc.;
- *Instances* representing actual realizations of concepts in the sensor data;
- *Modified Concepts* representing concepts that are modified by restrictions arising from intermediate results (i.e., the interpretation of part of the sensor data leads to restrictions on the interpretation of the remaining sensor data).

There are also the following network *links*:

- *Part Links* connecting concepts to the parts they consist of;
- *Concrete Links* connecting concepts on different *levels of abstraction*;
- *Specialization Links* establishing inheritance relations from general concepts to more specific ones.

Thus, our *knowledge model* is a network of concepts linked to each other by the various types of links.

Since there may be many different possibilities for the realizations of a concept, *modalities* have been introduced[1] with the implication that each individual modality may define the concept. Examples will follow in section 4. Furthermore, parts of a concept may be defined as being *obligatory* or *optional*.

For the definition of properties or features, a concept has a set of *attributes*. There may also be *relations* between the attributes of a concept. Each attribute references a function which computes its value and a judgment function which computes a measure of confidence for the attribute value. During the computation of an instance (i.e. the instantiation of a concept) values for all attributes, relations and the corresponding judgment values are calculated; therefore a concept can only be instantiated if instances for all obligatory parts and concretes of the concept have been computed before in previous analysis steps.

Due to errors in the image segmentation (arising from noise and processing errors) and ambiguities in the knowledge base (arising, for example, from the various modalities), competing instances may be computed for a single concept. In order to measure the degree of confidence of an instance, a *judgment* function is needed (and included into the definition of a concept). In most cases this judgment function combines the judgment of the attributes, relations, and the instances of the parts and concretes of a concept.

The goal of the analysis is represented by one or more concepts, the *goal concepts*. Subsequently, an interpretation of the whole sensor data is represented

[1] Another possibility for the representation of different concept realizations is to define a concept for each realization; this, however, inhibits a compact knowledge representation.

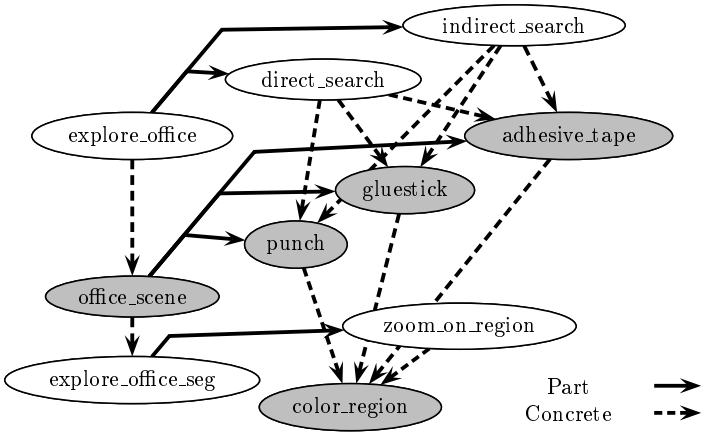

Fig. 1: Semantic network which combines the representation of camera actions (white ovals) with the representation of the scene (gray ovals). In addition, the network contains generalizations for some concepts, which are left out to keep up clarity.

by an instance of a goal concept. Now, in order to find the 'best' interpretation, the computation of an *optimal instance* of a goal concept with respect to the judgment function of the corresponding concept is required. In our approach the interpretation problem is viewed as a combinatorial optimization problem and solved as such (section 5).

4 Embedding Camera Actions into the Knowledge Base

4.1 Declarative Knowledge

The structure of the knowledge base of our application domain is shown in Fig. 1. As we have motivated in the introduction, the knowledge base unifies the representation of objects and their relations and the representation of camera actions on different levels of abstraction. The gray ovals represent information which would be found in almost any conventional knowledge base on this matter. This set of concepts contains, for example, the objects of the application domain, e.g. the concepts "punch", "gluestick" or "adhesive_tape" and their concrete representation which modelled by the concept "color_region". The concepts representing the objects are parts of the concept "office_scene"[2].

In addition to the representation of scene concepts, concepts for camera actions are integrated into the knowledge base. 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 instantiation of this concept computes a new pan angle

[2]In the following the "_seg" part of the concept names stands for segmentation.

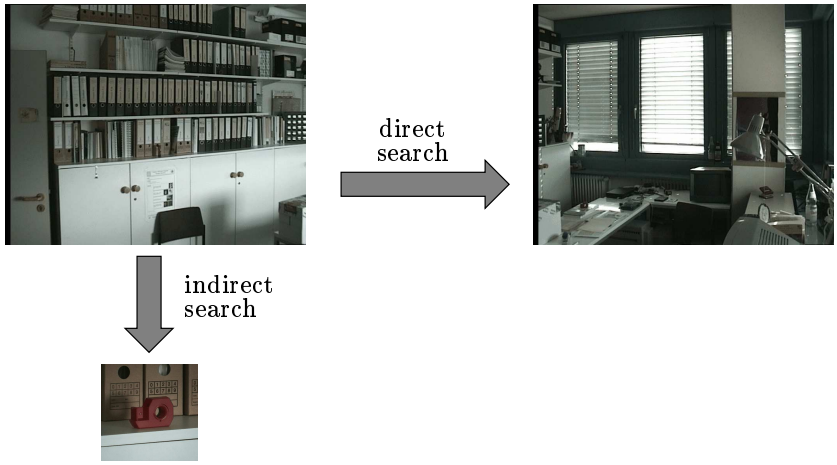

Fig. 2: Difference between indirect and direct search.

and a new zoom for the camera in such a way, that overview images, which are images captured with a small focal length, are obtained. If we look at all overview images as one image, we get a scan of the whole scene. The second example is the concept "indirect_search". This concept represents an indirect search [17], that is the search for an object using an intermediate object which is the punch in our application domain. Usually, large objects like tables or book shelves are used as intermediate objects. These objects have in common that they are relatively large and therefore can be found in an image captured with a small focal length. This is advantageous because less images are necessary to scan a whole scene. An example for the effect of direct and indirect search is depicted in Fig. 2. Using direct search a second overview images is taken, using indirect search with the cupboard as an intermediate object a close-up view of an area above the cupboard is generated. In our application the indirect search is performed only if we have already found an adhesive tape and a punch and we are searching for the gluestick. Recall that the punch is choosen as intermediate object in our application.

On the intermediate level of abstraction in Fig. 1, the camera action "zoom_on_region" can be found. The effect of this action is the fovealization of regions which are hypotheses for the objects the system is searching for. During the fovealization the pan angle and the zoom of the camera are adjusted in such a way that the region can be found in the middle of an image afterwards and is larger than before fovealization. The fovealization bases on the observation that the regions which are found in the overview images are too small for a good verification. Images taken after fovealization are called close-up views.

On the lowest level of abstraction camera actions are modelled which are performed data-driven and are independent of the application domain. The focus of a camera, for example, has to be adjusted in order to get sharp images. Furthermore, the zoom setting has to be chosen in such a way that not only

one homogeneous region is in the image.

Recall that the instantiation of a camera action concept leads to the selection of new camera parameters or the performance of a camera action. So, only one of the camera action concepts can be instantiated in a single step. In order to represent competing camera actions, i.e. actions which can not be performed at the same time, we make use of modalities (section 3). For example, the concept "explore_office" has as parts the concepts "direct_search" and "indirect_search", each of them is represented in one modality of "explore_office". The concept "region_seg_image" is another example for a concept which contains two modalities, one for "explore_office_image" and one for "office_image". In section 5 we explain how we deal with the ambiguities arising from the modalities.

4.2 Procedural Knowledge

In addition to declarative knowledge, each concept contains procedural knowledge which consists of the functions for value and judgment value computation of attributes, relations, and instances (section 3).

In the concepts of our scene representation, for example, functions for attribute value computation are defined which compute the color of a region (in the concept "color_region") or the height and the width of an object (in the concepts "punch", "gluestick" and "adhesive_tape"). In addition, each of these concepts contains attributes for the focal length and for the distance of the represented object refering to the close-up views of the objects.

A management of uncertainty is provided by the control based on the judgment functions (section 5). In order to rate the different camera actions, a utility-based approach is applied [18]. Probabilities are used to estimate the instances of the scene concepts "punch", "gluestick" and "adhesive_tape" because the utility measure relies on the evidence if an object has been found during analysis. The probabilities are calculated using a priori trained normal distributions for the individual attributes, the height, and the width of the objects. During training we calculate the mean and variance of these attributes for each object using 40 images.

5 Parallel Iterative Control

Our control algorithm [13, 14] treats the search for an optimal interpretation and for optimal camera actions as a *combinatorial* optimization problem and solves it by means of *iterative* optimization methods, e.g. simulated annealing, stochastic relaxation, and genetic algorithms. One advantage of this algorithm is its *any–time* capability. After each iteration step a (sub–)optimal solution is available and this solution can be improved by performing more iterations. If we have got enough time for analysis, the optimal solution is always found. Another advantage is that the algorithm allows an efficient exploitation of *parallelism* by automatically compiling the concept–centered semantic network into a fine–grained task–graph, the so–called *attribute network*. This network represents the dependencies of all attributes, relations, and judgments of concepts to be considered for the computation of goal instances. In Fig. 3 a schema of the parallel iterative control is shown. In following we explain how the control

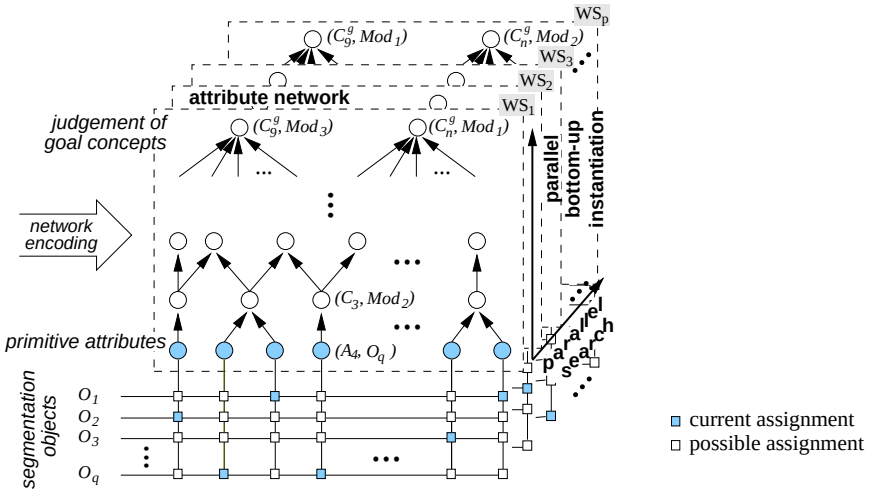

Fig. 3: Scheme of the parallel iterative control algorithm.

works principally. The attribute network is automatically generated in two steps from the semantic network:

- **Expansion** First the semantic network is expanded top-down such that all concepts which are necessary for the instantiation of the goal concepts exist. Concepts which have more than one predecessor in the network like the concept "color_region" in Fig. 1 are duplicated.
- **Refinement** The expanded network is refined by the determination of dependencies between sub–conceptual entities (attributes, relations, judgments, etc.) of all concepts in the expanded network. These sub–conceptual entities are represented by the nodes of the attribute network and the dependencies between them by the links.

Both steps are executed automatically in advance to search and depend only on the syntax of the semantic network representation. Nodes without predecessors (*primitive attributes* in Fig. 3) represent attributes that provide an interface to the initial segmentation (for example color regions), and nodes without successors (*judgment of goal concepts* in Fig. 3) represent the judgments (i.e. confidence measures) of goal concepts. Now, for the computation of instances of the goal concepts, all nodes of the attribute network are processed in a single bottom–up step. Therefore the flow of information is fixed from the primitive attribute node to the judgment nodes of the goal concepts. This bottom–up instantiation corresponds to a single iteration of the iterative optimization.

Parallelism can be exploited on the network and on the control level. Each node of the network for example may be mapped to a multiprocessor system for parallel processing (depicted in Fig. 3 as *parallel bottom–up instantiation*). In addition, several competing instances of the goal concepts may be computed

in parallel on several workstations. This is shown in Fig. 3 as *parallel search* on p workstations WS_1, ..., WS_p.

The search space the control algorithm has to deal with is determined by the competing segmentation results which arise, for example, from an erroneous segmentation, and by the different modalities of the concepts in the knowledge base (section 3). Recall that in our application domain the segmentation results are the color regions and the modalities determine which camera action is performed, e.g. the indirect or the direct search. In addition, we define the *current state of analysis* of our combinatorial optimization as a vector $\boldsymbol{r}$ which contains the assignment of segmentation results to primitive attributes and the choice of a modality for each (ambiguous) concept:

$$\begin{aligned} \boldsymbol{r} &= \left[(A_i, O_j^{(i)}); (C_k, Mod_l^{(k)})\right]^T \\ &\quad \text{with } i = 1, \ldots, m \; ; \; k = 1, \ldots, n \; . \end{aligned}$$

where m denotes the number of primitive attributes, j the index of the segmentation results, k the number of concepts in the semantic network, and l the index of the modality for each corresponding concept. In each iteration step judgments for the goal concepts are computed for the actual state of analysis. After an iteration step one randomly chosen entry of the analysis vector is changed. For example another segmentation result is bound to a primitive attribute node. Furthermore, a *performance function* for the state of analysis vector is introduced. The task of the control algorithm is now to optimize the performance function, i.e. to find *that* state of analysis which leads to the best interpretation, i.e. to an optimal instance of a goal concept.

Fig. 3 shows, for example, that in the current state of analysis for which the attribute network is computed on the first workstation (WS_1), the segmentation object O_q is assigned to the primitive attribute node A_4, modality 2 is assigned to the concept C_3, and modality 3 and modality 1 are assigned to the concepts C_9 and C_n, respectively, which are goal concepts (thus stated as C_9^g and C_n^g). Furthermore, it is shown that different instances (recall that the computation of instances depends only on the current state of analysis) are computed on the several workstations: the current state of analysis for which instances are computed on workstation p (WS_p) differs from that on WS_1 at least by the assignment of different modalities to the goal concepts C_9^g and C_n^g.

In our current application we have only one goal concept, which is "explore_office", and the performance function corresponds to the judgment function of this concept.

6 Expanding the Control to Actions

The goal in our application is to instantiate the concept "explore_office". Therefore we *alternately* have to interprete the image data and perform camera actions. This alternating computation cannot directly be expressed by either the syntax of the semantic network nor by the attribute network. This means that we have to extend the *control algorithm.*

If we compute the whole attribute network for our goal concept in one bottom–up step as described in section 5, we select the camera actions "direct_search" or "indirect_search" possibly without an optimal interpretation of

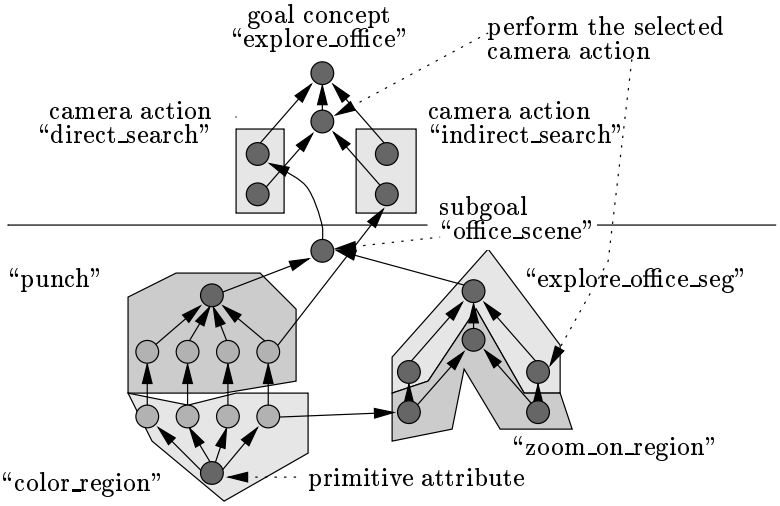

Fig. 4: Excerpt of the attribute network for the knowledge base depicted in Fig. 1 showing a subset of the network for the concepts “explore_office” and “office_scene”. Only the most important arrows are shown.

the concepts “punch”, “gluestick” and “adhesive_tape”. This makes no sense, because we first need to find an optimal interpretation for the view under the actual camera setting, before deciding which camera action has to be performed next. In addition, if we instantiate the concept “new_zoom” and then go on with the bottom–up computation of the attribute network we get an instance of “explore_office” based on image data which was taken with the old zoom.

In order to solve these problems, the control for the bottom–up instantiation of the attribute network is extended as it was already successfully demonstrated for an application with a speech dialog system [13]. The instantiation is divided into several data driven instantiations of subnets of the attribute network. The division is initiated by specifying subgoals prior to the computation of the attribute network which has to be done by the user once before the system is run. From these subgoals, the subnets can be automatically derived by the network encoder. This induces a partial order of the subgoals in the attribute network. Initial subgoals are chosen from the lowest level in the network. Analysis starts by the instantiation of the initial subgoal. This means that the bottom-up computation of the corresponding subnet is iterated until an optimum is found. Afterwards, the control chooses the next subgoal to be instantiated, and so on. This process continues until an optimal instance of the “explore_office” is found.

To give an example: If the user chooses, for example, the concepts "region_seg_image" and "office_scene" as subgoals, the control starts finding the best instance of "region_seg_image". Afterwards, it searches for the best instance of "office_scene". Once this instance has been found, the subnet which belongs to the goal concept "explore_office" is instantiated. This is done until the goal concept, that is "explore_office", is reached or a camera action is performed. If the judgment of the instance of the "explore_office" is below an application dependent threshold, the control starts again with the subgoal "region_seg_image". In Fig. 4 the subnets belonging to "explore_office" and to "office_scene" are depicted. The excerpt of the attribute network shows a part of the expanded knowledge base described in Fig. 1. Thus, we have three modified concepts of "color_region" which are bound to the concepts "punch", "gluestick" and "adhesive_tape". The highlighted areas contain the attributes which belong to the corresponding concepts. In addition, camera actions are performed by computing the attribute pan of concept "explore_office" and concept "explore_office_seg".

Fig. 4 depicts two subnets, one above the horizontal line, one below. The upper subnet belongs to the concept "explore_office" and shows the attributes required for the instantiation of that concept. Attributes originating from one concept are boxed and highlighted. The lower subnet explains the relations of attributes used to instantiate the subgoal "office_scene". Three modified concepts of "color_region" are bound to the concepts "punch", "gluestick" and "adhesive_tape"; only one ("punch") is depicted in the figure for simplicity. Depending on the state of analysis, a zoom camera action is performed if the color region assigned to the primitive attribute is too small.

7 Experimental Results

So far the lower part of the knowledge base in Fig. 1 is provided as one module. This part contains the concepts "office_image", "new_zoom", "new_focus", "explore_office_image", "explore_office_seg", and "zoom_on_region". In this module hypotheses for the red objects are computed by a histogram backprojection [19] which is applied to an overview image taken with the minimal focal length of the camera (cf. Fig. 5). In order to verify these hypotheses they are fovealized by moving the camera and varying the camera's focal length. This is exactly the task of the lower part of the knowledge base shown in Fig. 1. The hypotheses correspond to color regions which are the input of the primitive concept "color_region" of the semantic network. Thus, the primitive concept serves as interface to the module performing the task of the lower part of the knowledge base.

The suitability of the approach has been tested in 20 experiments while performing explorations in two different offices. In each experiment the objects were positioned at different places. Ten experiments took place in office_1 and the other ten in office_2. In the experiments, seven red objects are used, where three of them are modelled in the knowledge base. These three objects which are interesting for the interpretation step were hypothesized in 54 cases of 60 possible ones by the data driven hypotheses generation module using histogram backprojection. On average six close-up views were generated, that is, six object hypotheses were found in each overview image. The search space for the

Fig. 5: Overview images of two different office scenes (office_1 on the left, office_2 on the right). The close-up views below the overview images show fovealized object hypotheses.

	number of iterations N_i				
	10	30	50	100	150
office_1	57 %	82 %	93 %	93 %	93 %
office_2	13 %	46 %	54 %	70 %	79 %

Table 1: Percentage of correct recognized objects.

iterative control algorithm was reduced by restrictions concerning the color of the objects. These restrictions were propagated once from the higher concepts to the primitive concepts at the beginning of analysis. However, the task gets not trivial due to the propagation of the restrictions because on average between 43 and 66 color regions fulfill the restrictions. Therefore the system needs the ability to detect the objects in this set of hypotheses which is done by making use of the knowledge represented in the knowledge base. In Table 1 the results are shown for the two different offices. The recognition rates give the ratio between the number of correctly recognized objects and the total number of verified objects, performing N_i iterations. One can see that the recognition rate increases with the number of iterations up to a maximum of 93 % for office_1 and 79 % for office_2. Currently we use only 2D features which are view-point dependent. Because the objects' pose is more restricted in office_1 the recognition rate is higher than for office_2.

The increase with the number of iterations shows particularly well the anytime capability of the algorithm. The results revealed furthermore that 50 iterations for office_1 and 150 iterations for office_2 are sufficient to achieve an optimal result for a specific camera setting. The number of necessary iterations depends upon the number of hypotheses which are generated by the data driven module. For office_2 eight hypotheses were found on average, whereas for office_1 only five hypotheses were found. Therefore, more iterations had to be performed for office_2.

As optimization method the stochastic relaxation was used. The process-

ing cycle for one camera setting for interpretation (i.e., from the data driven hypotheses generation up to the computation of an optimal instance of "explore_office") lasts around five minutes. The major time need arises by moving the camera axes, waiting until the goal position is reached, for the median filtering in the histogram backprojection, and the segmentation of the color regions. One iteration, that is one bottom-up instantiation of the attribute network for the scene part of the knowledge base, takes on average only 0.015 s, that is, 2.25 s for 150 iterations.

8 Conclusion and Future Work

In this paper, we proposed an integrated formalism for representing and using knowledge about an application domain combined with the various camera actions the system needs to perform in order to explore a scene. The application domain is the exploration of office scenes. The current task of the system is the localization of three pre-defined red objects. In order to use the knowledge and actions represented we employed a parallel iterative control algorithm with any–time capabilities. Initial experiments have proved the feasibility of the approach.

Future work will concentrate on completing the implementation of the approach for the application presented. This includes a systematic determination of the subgoals by a goalconcept estimation and a reduction of processing time.

References

[1] K. Kawamura and M. Iskarous. Trends in Service Robots for the Disabled and the Elderly. In *Intelligent Robots and Systems*, pages 1647–1654, München, 1994.

[2] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 2(3):333–356, 1988.

[3] 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.

[4] D. McKeown, W. Harvey, and J. McDermott. Rule-based interpretation of aerial imagery. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7(5):570–585, 1985.

[5] T. Matsuyama and V. Hwang. *SIGMA. A Knowledge-Based Aerial Image Understanding System*, volume 12 of *Advances in Computer Vision and Machine Intelligence*. Plenum Press, New York and London, 1990.

[6] 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.

[7] T. Levitt, T. Binford, G. Ettinger, and P. Gelband. Probability based control for computer vision. In *Proc. of DARPA Image Understanding Workshop*, pages 355–369, 1989.

[8] S. J. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach.* Prentice-Hall, Englewood Cliffs, NJ, 1995.

[9] J. Pearl. *Probabilistic Inference in Intelligent Systems. Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[10] 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, Bombay, India, 1998.

[11] A. Winzen and H. Niemann. Automatic model–generation for image analysis. In W. Straßer and F.M. Wahl, editors, *Graphics and Robotics*, pages 207–219, Berlin Heidelberg, 1995. Springer.

[12] H. Niemann and G. Sagerer. A model as a part of a system for interpretation of gated blood pool studies of the human heart. In *Proc. 6th ICPR*, pages 16–18, München, 1982.

[13] Fischer, J. and Niemann, H. and Noeth, E. A Real–Time and Any–Time Approach for a Dialog System. In *Proc. International Workshop Speech and Computer (SPECOM'98)*, pages 85–90, St.–Petersburg, 1998.

[14] H. Niemann, V. Fischer, D. Paulus, and J. Fischer. Knowledge based image understanding by iterative optimization. In G. Görz and St. Hölldobler, editors, *KI–96: Advances in Artificial Intelligence*, volume 1137 (Lecture Notes in Artificial Intelligence), pages 287–301. Springer, Berlin, 1996.

[15] H. Niemann, G. Sagerer, S. Schröder, and F. Kummert. ERNEST: A semantic network system for pattern understanding. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(9):883–905, 1990.

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

[17] 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.

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

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