Active Knowledge–Based Scene Analysis

D. Paulus, U. Ahlrichs, B. Heigl, J. Denzler, J. Hornegger, H. Niemann

Lehrstuhl für Mustererkennung (LME, Informatik 5)
Martensstr. 3, Universität Erlangen–Nürnberg, 91058 Erlangen
http://www5.informatik.uni-erlangen.de
Ph: +49 (9131) 85–7894 Fax: +49 (9131) 303811
email: paulus@informatik.uni-erlangen.de

Accepted at

FIRST INTERNATIONAL CONFERENCE ON COMPUTER VISION SYSTEMS

JANUARY 13-15, 1999
Las Palmas, Gran Canaria SPAIN
(20 pages (limit: 20 pages))

Abstract: We present a modular architecture for image understanding and active computer vision which consists of three major components: Sensor and actor interfaces required for data–driven active vision are encapsulated to hide machine–dependent parts; image segmentation is implemented in object–oriented programming as a hierarchy of image operator classes, guaranteeing simple and uniform interfaces; knowledge about the environment is represented either as a semantic network or as statistical object models or as a combination of both; the semantic network formalism is used to represent actions which are needed in explorative vision.

We apply these modules to create two application systems. The emphasis here is object localization and recognition in an office room: an active purposive camera control is applied to recover depth information and to focus on interesting objects; color segmentation is used to compute object features which are relatively insensitive to small aspect changes. Object hypotheses are verified by an A*–based search using the knowledge base.

Keywords: Computer vision system, object–oriented design, object recognition, scene analysis, mobile systems, knowledge–based analysis

Contents

Active Knowledge–Based Scene Analysis

D. Paulus, U. Ahlrichs, B. Heigl, J. Denzler, J. Hornegger, H. Niemann

Lehrstuhl für Mustererkennung (LME, Informatik 5)
Martensstr. 3, Universität Erlangen–Nürnberg, 91058 Erlangen
http://www5.informatik.uni-erlangen.de
Ph: +49 (9131) 85–7894 Fax: +49 (9131) 303811
email: paulus@informatik.uni-erlangen.de

Abstract We present a modular architecture for image understanding and active computer vision which consists of three major components: Sensor and actor interfaces required for data–driven active vision are encapsulated to hide machine–dependent parts; image segmentation is implemented in object–oriented programming as a hierarchy of image operator classes, guaranteeing simple and uniform interfaces; knowledge about the environment is represented either as a semantic network or as statistical object models or as a combination of both; the semantic network formalism is used to represent actions which are needed in explorative vision.
We apply these modules to create two application systems. The emphasis here is object localization and recognition in an office room: an active purposive camera control is applied to recover depth information and to focus on interesting objects; color segmentation is used to compute object features which are relatively insensitive to small aspect changes. Object hypotheses are verified by an A*–based search using the knowledge base.

1 Introduction

Autonomous mobile systems with visual capabilities are a great challenge for computer vision systems since they require skills for the solution of complex image understanding problems, such as driving a car [36] or exploring a scene. The system presented in this contribution provides mechanisms for knowledge–based image understanding and active computer vision. It combines and links various modules for low–level image processing, image segmentation, and high–level image analysis. We combine data–driven and knowledge–based techniques in such a way that a goal–directed exploration guided by the explicitly represented knowledge is possible. The major goal here is to explore a scene with an active camera device. This can also be used in autonomous mobile systems which navigate and act based on visual information. Such systems need an explicit representation of actions and search strategies. A literature review in [5] on the topic of

This work was funded partially by the *Deutsche Forschungsgemeinschaft* (DFG) under grants number SFB 182 and SFB 603. Only the authors are responsible for the contents.

knowledge–based image analysis gives a comprehensive discussion of the state of the art. Image analysis systems have also been reported for example in [21, 22]. In [21] as well as here, semantic networks are used as a formalism for knowledge representation. We now use this formalism for the unified representation of objects, scenes, actions, and strategies in order to provide flexible and exchangable strategies for active vision and scene exploration.

A software system for image understanding usually has a considerable size. The major problem in software design of general imaging systems is that on the one hand highly run–time efficient code and low–level access to hardware is required, and that on the other hand a general and platform–independent implementation is desired which provides all data types and functions also for at least intermediate–level processing, such as results of segmentation. Today's software engineering is closely coupled with the ideas of object–orientation which can help simplifying code reuse; if applied properly, it unifies interfaces and simplifies documentation by the hierarchical structure of classes. Genericity provides an alternative solution to software engineering problems. Both concepts are available in C++. Object–oriented programming has been proposed for image processing and computer vision by several authors, in particular for the image understanding environment [15]; this approach is mainly used to represent data. We also use object–oriented programming for operators, devices, and actions.

In Sect. 2 we outline the general structure of our system and the object–oriented implementation. In Sect. 3 we describe some of the modules which are provided by the system; special emphasis is layed on the knowledge representation for computer vision and on the extension of the semantic network formalism to represent strategies and actions. We apply these modules in Sect. 4 to two problems in computer vision.

The goal of our example application in Sect. 4.1 is to explore an office room. Objects are hypothesized in the image based on their color. Their 3–D position is estimated from a coarse 3–D map computed from trajectories of colored points which are tracked during during a translational motion of the active camera. The objects are chosen in such a way that they cannot be distinguished by their color solely. Close–up views are captured and segmented into color regions. Features of these regions are subject to matching with the knowledge base. If objects are not found in the scene, the camera is moved based on action descriptions found in the knowledge base.

In Sect. 4.2 we describe a recent research project in the area of visual guided autonomous mobile systems. Many of the algorithms described in Sect. 3 and Sect. 4.1 are involved. First results for visual self–localization based on color histograms in natural office scenes are presented.

We conclude with a summary and future directions in Sect. 5.

2 System Architectures

The need for a flexible, knowledege–based computer vision system with real–time capabilities lead to "**An image analysis system**" (ANIMALS, [1, 26, 25]) which

is implemented in C++. It provides modules for the whole range of algorithms from low–level sensor control up to knowledge–based analysis and actions. For a unified documentation and description of the numerous modules, the notation as introduced in [23] has been used.

2.1 Data Flow for Knowledge–Based Analysis

The general problem of image analysis is to find the *optimal* description of the input image data which is appropriate to the current problem. Sometimes this means that the most precise description has to be found, in other cases a less exact result which can be computed faster will be sufficient. For active exploration, the goal is to fulfill the task which is described in the knowledge base.

These problems can be divided into several sub–problems. After an initial preprocessing stage, images are usually segmented into meaningful parts. Various segmentation algorithms create so called segmentation objects [25] which can be matched to models in a knowledge base containing expectations of the possible scenes in the problem domain. This is achieved best if the formalism for the models is similar to the structure of segmentation results, as it is the case for semantic networks and segmentation objects [23].

An overview of the main components of our image analysis system is shown in Figure 1; data is captured and digitized from a camera and transformed to a description which may cause changes in camera parameters or tuning of segmentation parameters. Models which are collected in the knowledge base are created from segmentation results (in Sect. 3.3) or at least have similar structure (in Sect. 3.6). These models are used for the analysis. Image processing tasks are shown in oval boxes; data is depicted as rectangles.

The dotted lines in Figure 1 indicate that a control problem has to be solved in active vision or active exploration resulting in a closed loop of sensing and acting. Information is passed back to the lower levels of processing and to the input devices; this way, parameters of procedures can be changed systematically, or the values of the camera and lens can be modified. Changes to the param-

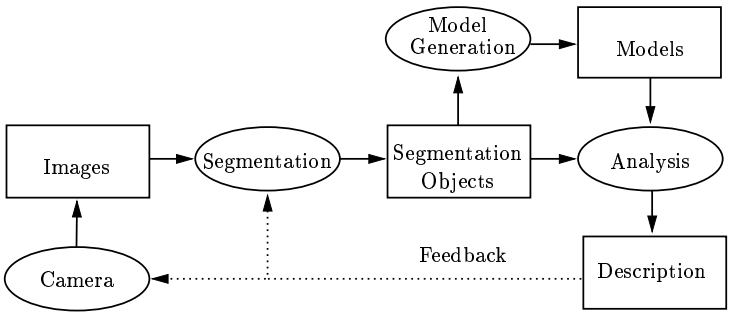


Figure 1. Data flow in an image analysis system after [25]

eters of the image input process, selection of the appropriate algorithms, and parameter control for both are summarized under the term *action*.

2.2 Examples of Other Software Environments

From the variety of software systems for image processing and analysis, we choose two well known examples.

A group of leading experts in image analysis joint their efforts for a common image understanding environment [15]. The system was planned as a basis for image processing, computer vision, and knowledge based image analysis. The system covers all areas of imaging with many applications; due to the many contributors, a variety of ideas has to be united into a hierarchy of classes. The design goals are: object–oriented programming with graphical interaction, extensibility, program exchange and a common performance evaluation platform for image processing. Real–time processing was explicitly excluded from the original goals [15, p. 160]. In the present version, no classes are provided for devices such as cameras. This environment is applied for example in [18] to the analysis of satellite images.

The other widely used system is Khoros [28] which provides a nice graphical user interface for image processing systems. Data structures beyond images and matrices are not available to the user. Therefore, knowledge–based processing is not integrated in the system. The interface to the large function library is compatible to C and does not provide object–oriented features.

Real–time processing as well as symbolic data structures are crucial for active exploration. Both systems have thus to be modified or extended to be used for our purpose.

2.3 Object–Oriented Design for Image Analysis

The algorithms and data structures of our system are implemented in a **H**ierarchy of **P**icture **P**rocessing **O**bject**S** (HIPPOS, written as $\Upsilon\pi\pi o\varsigma$ [26, 25]), an object–oriented class library designed for image analysis which is based on the commonly used NIHCL C++ class library. In [26], the data interfaces were defined as classes for the *representation* of segmentation results. The *segmentation object* [26] provides a uniform interface between low–level and high–level processing. In [16], this system is extended to a hierarchical structure of *image processing and analysis classes* and *objects* (cmp. [4]). Objects are the actual algorithms with specific parameter sets which are also objects (OperatorOpt in Figure 2, [16]). Classes as implementation of algorithms are particularly useful, when operations require internal tables which increase their efficiency since tables can then be easily allocated and handled. The basic structure of the class hierarchy for line–based image segmentation is shown in Figure 2. On a coarse level, operators for line–based segmentation can be divided into edge detection, line following, gap closing, and corner and vertex detection. For each processing step, which is implemented here as a class, there exists a large variety of choices in the literature. When the whole sequence of operations is subjected

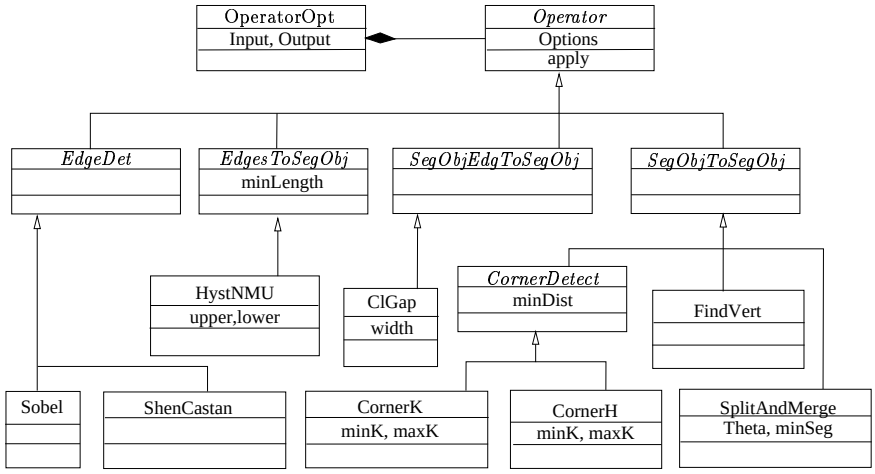


Figure 2. Part of a class hierarchy for image operators from [16] in UML notation

to optimization, either manually or automatically, it is crucial to have similar interfaces to exchangeable single processing steps, such as several corner detection algorithms. This is greatly simplified by object–oriented programming and polymorphic interfaces as shown here. Only the type of input and output data has to remain fixed for such a system. This is guaranteed by the abstract classes directly derived from the class `Operator`; for example the *EdgesToSegObj* defines the interface for the conversion of edge images to segmenation objects.

Several segmentation algorithms are implemented in our system which operate on gray–level, range, or color images using the interfaces provided by the segmentation object. To give an example, the application in Sect. 4.1 works on color images and applies a split–and–merge algorithm extended to color images. The result is represented as a segmentation object containing chain code objects for the contours of the regions. The major advantages of operator–classes for segmentation and image processing are threefold:

- Algorithms can be programmed in an abstract level referencing only the general class of operations to be performed; extensions of the system by a new derived special operator will not require changes in the abstract algorithm.
- Such an extension cannot change the interface which is fixed by the definition in the abstract base class. This guarantees reusable, uniform, and thus easy–to–use interfaces.[1] In Figure 2, this is shown for the edge detectors (Sobel and ShenCastan) and for two different corner detectors (from [16]).
- Dynamic information about the operator which is actually used, is available. For example, a program may just reference a filter object; during run time it will be decided which concrete filter should be used.

[1] Of course, this is not *always* possible to achieve.

In Sect. 2.4 we argue that these advantages produce no additional run–time overhead. Similar hierarchies as the one in Figure 2 exist for filters and for region–based segmentation.

2.4 Software Experiments

Our system is compiled and tested for various hardware platforms, including HP (HPUX 9.07 and HPUX 10.20, 32 bit and 64 bit Versions), PC (Linux), SGI (IRIX 6.20), Sun (Solaris), etc.

The NIHCL class library served as a tool for most general programming problems, such as dictionaries, lists, and external data representation. It is available for all platforms listed above. STL was not available when the system was initiated but has now been used as well.

Operator classes are invoked by virtual function calls. This overhead in comparison to direct function calls is negligible, as long as the operation to be performed is non–trivial (as it is the case in the examples given). Pixel access — of course — may not be programmed by virtual function calls. This would slow down processing too much. Instead, the concept of genericity is used here (provided by templates in C++). Safe and efficient pixel access without any run–time overhead is provided as described in [25]. The application in Sect. 4.2 shows that real–time processing is possible using this approach.

3 Modules of ANIMALS

Various modules are provided in ANIMALS which were implemented for several applications. Since all segmentation algorithms use the common interface by segmentation objects, the compatiblity is high. This flexibility first requires additional effort for portable software. On the long run it reduces the effort of software maintenance.

3.1 Sensors and Actors

Due to the variability of the used hardware, several frame grabbers, cameras, lenses, stereo devices, etc. are connected to the system. In order not to burden the programs by a large number of switches, all these devices are encapsulated as classes. Whereas the internal implementation may be sophisticated and thereby provides the required performance, the user interfaces for these classes are simple in C++. To give an example, the stepper motor objects can all be assigned the desired position by the assignment operator in C++; thus, the vergence axis of the stereo head has the same software interface as the pan axis of a surveillance camera.

Calibration of zoom lenses is a complex problem since the motors used in consumer cameras will not always lead to accurate postitions. For the object localization system in Sect. 4.1 we need the enlargement factor related to a given position of the zoom stepper motor. The calibration according to [39] is

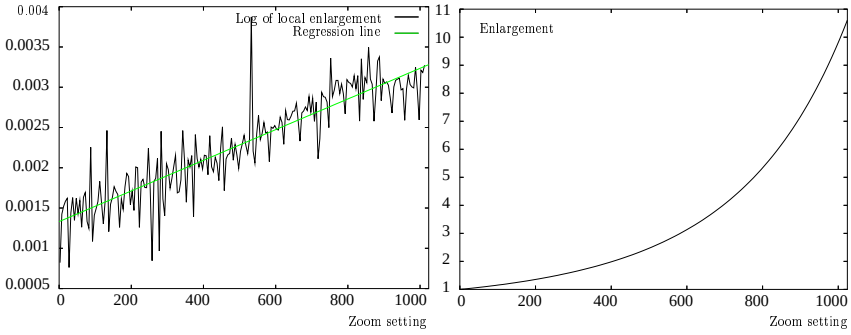


Figure 3. Logarithm of local enlargement factors and estimated enlargement factors

hard to do fully automatically, since a calibration pattern has to be chosen that is visible and that provides stable segmentation data for all possible focus and zoom settings.

In a simple approach we investigated the enlargement for two slightly different zoom settings for arbitrary lines in an image which are defined by two corresponding points, each. The corresponding points are found by tracking color points (Sect. 3.5) during zooming. To compute the factor from the smallest focal length to a given position, all intermediate results have to be multiplied. Naturally, this approach is not very stable since local errors accumulate. We took the logarithm of the factors and approximated these values by a regression line (Figure 3 (left)). The enlargement factors are computed by the exponential function (Figure 3 (right)). This provided reasonable accuarcy in the experiments in Sect. 4.1.

The results of this algorithm are recorded together with the camera object. Persistent storage of this object records all available calibration information.

3.2 Color

Most algorithms in ANIMALS operate on color images as well as gray–level images. Color provides simple cues for object localization since it is not very sensitive to aspect changes. Color histograms are used in [35] to form hypotheses for the position of an object in the two–dimensional projection in an image. A color image $[\boldsymbol{f}_{ij}]_{1\le i\le M, 1\le j\le N}$ is searched for an object which is characterized by its histogram $\boldsymbol{T} = [T_l]_{l=1\ldots L}$ in some quantization L. In addition, the approximate size of the object in the image is needed for the algorithm; this size is represented by a mask. $\boldsymbol{D}_r$ covering the object. The function h maps a color vector $\boldsymbol{f}$ to an index $l = 1 \ldots L$ in the histogram and thus permits to use arbitrary quantizations. The principle of the algorithm is shown in Figure 4. The histogram $\boldsymbol{H}$ of the image is used to produce an output image $\boldsymbol{B}$ of the size of the input image; internally, an intermediate image $\boldsymbol{A}$ of the same dimension is used.

Given: image histogram $\boldsymbol{T} = [T_l]_{l=1\ldots L}$ of an object, Wanted: object position (i_t, j_t)
Compute color histogram $\boldsymbol{H} = [H_l]_{l=1\ldots L}$ of given image
FOR Each bin $l \in \{1, \ldots, L\}$
$R_l = \min\{\frac{T_l}{H_l}, 1\}$ (compute ratio histogram $\boldsymbol{R} = [R_l]_{l=1\ldots L}$)
FOR All positions (i, j) in the image
$A_{i,j} := R_{h(\boldsymbol{f}_{i,j})}$, where $\boldsymbol{f}_{i,j}$ denotes the color vector at position (i, j)
$\boldsymbol{B} := \boldsymbol{D}_r \star \boldsymbol{A}$, where $\star$ denotes convolution
$(i_t, j_t) := \text{argmax}_{i,j}(B_{i,j})$

Figure 4. Localization of objects by histogram backprojection according to [35].

Color histograms for different color spaces are again represented as classes. Backprojection is a method of a common base class for these histograms. This means that the algorithm in Figure 4 can mostly be programmed as it is shown in the mathematical notation, which does not mention any particular color space.

3.3 Statistical Object Recognition

In a Bayesian framework for 3–D object recognition using 2–D images [1, 19], statistical model generation, classification, and localization is based on projected feature vectors $\boldsymbol{O}$. We assume that the image $[f_{i,j}]_{1\leq i\leq M, 1\leq j\leq N}$ is transformed into a segmentation object of two–dimensional feature vectors $\boldsymbol{O} = \{\boldsymbol{o}_k \in \mathbb{R}^2 | 1 \leq k \leq m\}$ consisting of points (e.g. corners or vertices) or lines which can be detected by several combinations of segmentation operators from the class hierarchy shown in Figure 2, e.g. by the operator EdgeToSegObj. Results are shown in Figure 5. Model densities of 3–D objects appearing in images embody three principal components: the uncertainty of segmented feature vectors, the dependency of features on the object's pose, and the correspondence between image and model features. Due to the projection of the 3–D scene to the 2–D image plane, the range information and the assignment between image and model features is lost. The statistical description of an object belonging to class Ω_κ is defined by the density $p(\boldsymbol{O}|\boldsymbol{B}_\kappa, \boldsymbol{R}, \boldsymbol{t})$, and discrete priors $p(\Omega_\kappa)$, $1 \leq \kappa \leq K$, if only single objects appear, or $p(\Omega_{\kappa_1}, \Omega_{\kappa_2}, \ldots, \Omega_{\kappa_q})$ for multiple

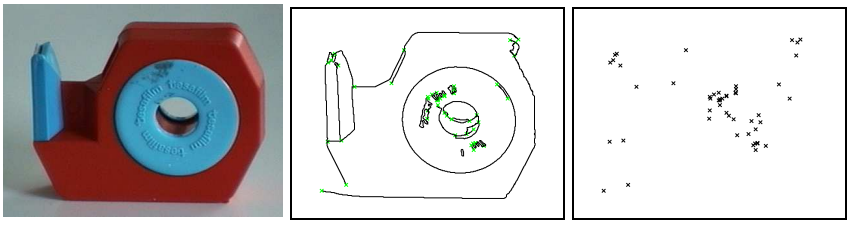


Figure 5. Object (left) segmented to lines and vertices (center) and points (right)

object scenes; the priors for the occurrences of an object are estimated by their relative frequencies in the training samples. The parameter $\boldsymbol{R}$ denotes rotation and projection from the model space to the image plane; $\boldsymbol{t}$ denotes translation. The parameter set $\boldsymbol{B}_\kappa$ contains the model–specific parameters which model the statistical behavior of features as well as the assignment. In the special case of segmented point features, $\boldsymbol{B}_\kappa$ statistically models the accuracy and stability of the object points.

Let us now assume that the parametric density of the model feature $\boldsymbol{c}_{\kappa,l_k}$ corresponding to $\boldsymbol{o}_k$ is given by $p(\boldsymbol{c}_{\kappa,l_k}|\boldsymbol{a}_{\kappa,l_k})$, where $\boldsymbol{a}_{\kappa,l_k}$ are the paramters for the density function of the feature $\boldsymbol{c}_{\kappa,l_k}$. A standard density transform results in the density $p(\boldsymbol{o}_k|\boldsymbol{a}_{\kappa,l_k},\boldsymbol{R},\boldsymbol{t})$ which characterizes the statistical behavior of the feature $\boldsymbol{o}_k$ in the image plane dependent on the object's pose parameters.

Using the robot or a camera mounted on some actor we record a set of images where the parameters $\boldsymbol{R}$ and $\boldsymbol{t}$ are known from calibration. For a segmented point it can be tested statistically, that a Gaussian distribution adequately models the features. The unknown parameters of the model density can be estimated by a maxium likelihood estimator. For object recognition and pose estimation, the parameters $\boldsymbol{R}$ and $\boldsymbol{t}$ are unknown as well and have to be estimated. Optimization classes (Sect. 3.8) were developed for this application [19].

Experimental evaluations compared standard methods for pose estimation with the introduced statistical approach. The statistical pose estimation algorithm requires 80 sec. using global optimization; to compare, the alignment method [37] needs 70 sec. in average on an HP 735. On a smaller sample of 49 images, the correct pose is computed for 45 images with the statistical approach; the alignment method failed for 11 images. In a test based on 1600 randomly chosen views of objects, the recognition rates for the statistical approach were in the range of 95% for 2–D recognition.

In another approach [27] we recognize objects directly by appearance–based statistical models of the image data. The approach can handle gray–level, color, and depth images as well as combinations of them.

3.4 Point Tracking

The basic idea in [34] is to select those points in a sequence of gray–level images, which exhibit features for stable tracking. Thereby the two questions which were formerly posed independently, are now combined to one problem: the question which points to select and how to track. The method minimizes a residue defined for a given window by approximating the image function with the spatio–temporal gradient in the Taylor expansion. By applying this method iteratively, the displacement is determined in sub–pixel accuracy.

We extended this differential method defined for real–valued image functions to vector–valued ones by substituting the gradient by the Jacobian matrix. Another extension to the original method is to restrict the search space of correspondenced only to an orientation assuming pure translational camera movement. In this case the Jacobin matrix is substituted by the directional derivative for the orientation vector of the epipolar line; this line links each pixel to the epipole.

For both extensions the criterion for tracking has been adapted. The use of color values shows, that more points can be selected and the tracking is more stable. In the case of restricted search space, the criterion results in a large gradient in the search direction. Therefore a huge number of points can be selected. We found that better and more robust tracking is possible in RGB than in an perceptually motivated color space or in gray–level image sequences; the number of points lost during tracking in RGB is more then 20% smaller than for gray–level images [1].

3.5 3–D Reconstruction

The module in Sect. 3.4 can be used to recover 3–D information. A camera mounted on a linear sledge is moved horizontally to record a sequence of images. Since points are tracked, no such correspondence problem as in stereo vision has to be solved explicitly. Instead, the regression line through the trajectory of a point determines its disparity. The regression error can be used as a measure for the certainty of the disparity value. The reliability of the range value is proportional to the disparity multiplied by the reliability of the disparity. The algorithm accepts an abstract point tracking operator (object), i.e., the same algorithm can be used for gray–level as well as color images.

3.6 Knowledge Base

In Sect. 3.3 we represented single objects by model densities. Alternatively, structural knowledge about scenes, object, as well as actions and strategies can be represented in a semantic network. Semantic networks have been applied successfully e.g. in [21, 22, 24] to pattern analysis problems. They provide an intuitive formalism for the representation of objects and facts which are required for computer vision. Our goal is to continue the work on knowledge representation concerning the semantic network formalism ERNEST [1, 24] by the integration of camera actions into the knowledge base. We also re–use the existing control algorithms for the semantic network. Alternative solutions for the explicit representation of actions are e.g. and–or trees [14] or Bayesian networks [32].

Using the notation and structure defined in [23, 33], a semantic network is a directed acyclic graph consisting of nodes and labeled edges. The nodes are concepts composed of attributes. They are used for the representation of objects and camera actions. The edges denote specialization which implies inheritance of attributes, part–of relations, and the concrete link which links entities of a different level of abstraction. Since image processing is never exact, all entities in the semantic network have an attached judgment which measures the degree of certainty. The judgment values are used to guide an A* graph search algorithm. The expansion of nodes in the search tree during analysis follows six inference rules [33], i.e. the control works independently of judgment functions and of the particular semantic network used in an application. Alternative possibilities for control strategies such as Markov decision processes are discussed in [12]; we also provide a second control algorithm called "parallel iterative control" [13].

Computer vision can be seen as an optimization problem which searches for the best match of segmentation data to the objects in the knowledge base and chooses optimal camera actions. A state search by A* thus is appropriate to solve this problem.

Figure 6 shows a typical semantic network. The lower part of the network contains objects which are used in the application in Sect. 4.1. The gray–shaded ovals represent different camera actions where each competing action (direct_search for a search without intermediate object [14], punch_besides_gluestick for a search for a punch using the intermediate object glue stick, ...) is linked to the concept explOfficeScene by competing part links; the control resolves these alternatives which are collected in so–called sets of modality (see [24] for details), i.e. for each alternative a new node in the search tree is generated during analysis. Concrete links relate the camera actions such as e.g. direct_search to the knowledge on objects. An instantiation of the concept direct_search calculates a new value for the attribute "pan position". The same holds for punch_besides_gluestick. Both instances have associated judgments, which are now used by the control in combination with the judgments of the scene knowledge to select the next search tree node for expansion. A subsequent instantiation of explOfficeScene using the higher judged camera action yields a camera movement to the position stored in the pan attribute. After this movement, new instances for the concepts representing the scene knowledge (colorRegion, punch ...) are generated. If the judgments of these concepts get worse, the search tree node which contains the instance of the other camera movement becomes the node with the highest judgment in the search tree. Therefore this node is selected by the control for expansion, which causes a second instantiation of the concept explOfficeScene. During this instantiation the other camera movement is performed.

The implementation, naturally, provides classes for all major items in the semantic network. A formal language definition is used to generate a parser and code generator which creates C++ code from the knowledge–base definition. Two control strategies can be tested on only one (unchanged) knowledge base definition. The classes involved here are an abstraction of the analysis procedure which is seen as a sequence of analysis states.

3.7 Active Contours and Active Rays

A simple and thereby powerful method for fast object detection and tracking is based on active contours or snakes [20]. A number of control points has to be chosen which define a parametric curve around an object. From an initial estimate, an energy minimization process contracts the contour to the object. To track moving objects, this step is repeated over time, in combination with some prediction mechanism to introduce some coherence of the contour movement in time [7].

In [1, 7] we proposed the notion of active rays which can also be used for the description of an object's contour; this reduces the 2–D contour extraction problem of active contours to a 1–D one. In this case, rays are cast in certain directions from an initial reference point inside the contour. The image is sampled

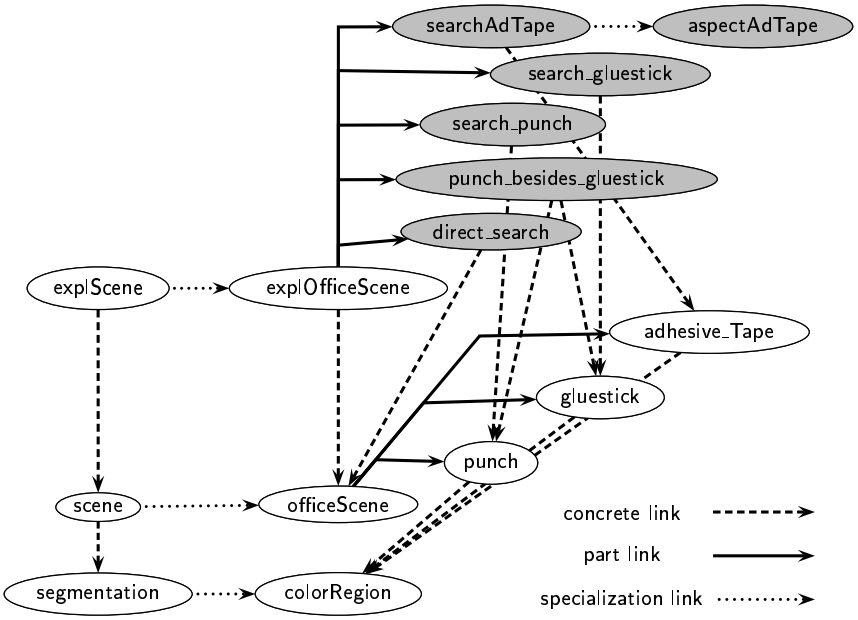


Figure 6. Combined representation of actions and objects in a knowledge base

along each ray. The contour points are now located in 1–D signals, i.e. the sampled image date on each ray. Similar to active contours, an internal and external energy is defined. The contour is extraced by minimizing the sum of these energies. In contrast to active contours, the energy minimization now takes place only along 1–D rays to attract points on each ray to the estimated object contour. The coherence in space of the contour is forced by the internal energy. Both ideas, active contours and active rays, are naturally represented as classes and share many lines of code.

Compared to active contours, there are several advantages of active rays for real–time applications: first, due to the representation of the contour points, parameterized by an angle defining the direction of each ray, neither any crossing can occur in the contour nor the contour points can move along the contour of the objects. Thus, prediction steps can be robustly applied. Second, an any–time algorithm can be defined which allows for an iterative refinement of the contour depending on the time which is available for each image. This is an important aspect for real–time applications. Third, textural features can be efficiently defined on 1–D signals, to locate changes in the gray–value statistics, identifying borders between two textured areas. This is a computational expensive task for active contours due to the independent search in the 2–D image plane [31]. Finally, an efficient combination with a 3–D prediction step is possible by using a similar radial representation of 3–D objects [8]

A complete object tracking system, called COBOLT (contour based localization and tracking, [7]) has been implemented in the ANIMALS framework

to evaluate the new approach in the area of real–time pedestrian tracking in natural scenes [9]. A pan–tilt camera is looking on a place in front of our institute. A motion detecion module detects moving objects and computes the initial reference point for active rays. The tracking module computes for each image the center of gravity of the moving pedestrian by using active rays and changes the settings of the axes as described in Sect. 3.1 to keep the moving object in the center of the image. In five hours of experiments under different weather conditions, in 70% of the time tracking was successful. On an SGI Onyx with two R10000 processors, 25 images per second could be evaluated in the complete system which summarizes the image grabbing, tracking and camera movement. For contour extraction alone using 360 contour points, approximately 7 msec. are needed per image.

3.8 Optimization

The solutions of the optimization problems outlined in Sect. 2.1, Sect. 3.3, and Sect. 3.7 require that several strategies for optimization are evaluated. The results of the individual algorithms, the sequence of the operations, and the setting of the camera parameters, each is included in the optimization process.

Probabilistic optimization routines which allow practically efficient solutions for object recognition are discussed in [19]. Again, a class hierarchy for optimization strategies similar to the operator hierarchy above simplifies the experiments.

The basic idea of the implementation is to program the algorithms independently from the function which is to be optimized. An abstract base for all optimization strategies has an internal variable which is the function to be optimized; the class provides a method for minimization or maximization to all derived classes.

All optimization algorithms can be divided into global and local procedures; additional information may be present such as e.g. the gradient of the function which yields another class in the hierarchy. Procedures which use the function directly are e.g. the combinatorial optimization, the simplex algorithm, or the continuous variant of the simulated annealing algorithm. The gradient vector can be used for the optimization of first order. Examples are the iterative algorithms implemented in [19], the algorithm of Fletcher und Powell, and the well known Newton–Raphson iteration. The interface to the optimization classes simply accepts vector objects of real numbers and does not impose any other constraints on the parameters other than an interval in the case of global optimization. The computation times measured in [19] for the optimization classes were around one minute for 10000 calls to the function to be optimized.

4 Examples and Applications

4.1 Active Object Recognition

The goal of our major example in the context of this article is to show that a selection of the modules presented in Sect. 3 can be combined to build a system

that locates objects in an office room. In [38] colored objects are located without prior knowledge; the image is partitioned and searched in a resolution hierarchy. The system here actively moves the camera and changes the parameters of the lens to create high resolution detail views. Camera actions as well as top–level knowledge are explicitely represented in a semantic network; the control algorithm of the semantic network used to guide the search is independent of the camera control. Knowledge on objects is represented in [11] by a hierarchical graph which could as well be formulated as a semantic network; the viewpoint control described there is part of the control algorithm for evaluating the graph. In contrast, [30] uses Bayesian networks to represent scenes and objects. Evidence gathered during analysis determines the next action to be performed which can be either a camera action or a new segmentation.

Real–world objects (e.g. a tape roller Figure 5, a glue stick and a punch Figure 7 right) are used in our experiments. The objects occupy only few pixels in the wide–angle views captured initially (Figure 7 left); they are presented to the system first isolated from the background; their color histogram for the current lighting is recorded. The approximate size of the object is stored in the semantic network manually in advance. This simple object model is sufficient to discriminate the objects in this example. The formalism used as well as the control structure allows for arbitrary other object models, such as for example aspects as applied in [11]; such a model will be simple for the object Figure 5 (left) but will be complex in the case of objects such as the punch in Figure 7 (right). Hypotheses for object locations in the scene are generated based on color using the algorithm outlined in Sect. 3.2. Results are shown below in Figure 9 for the object from Figure 5. An evalutation of six color normalization algorithms in [6] on 20 objects including those in Figure 5 and Figure 7 (right) revealed that the choice of the best color normalization algorithm with respect to object localization depends on the particular object. In most cases, UV histograms performed better than RGB.

The pan–tilt unit mounted on the linear sledge is moved to estimate 3–D information using the ideas outlined in Sect. 3.5; we compute a set of scattered 3–D points. The focal length is known from calibration (Sect. 3.1). A result is shown in Figure 8; neither the accuracy nor the number of points is sufficient to estimate 3–D surfaces of the objects.

The subsequent goal now is to fovealize each object hypothesis and to generate close–up views. This is required since for the object's size in the overview image no stable features based on segmented regions can be detected. Figure 7 shows an overview of the scene captured with minimal focal length. Figure 7 shows three hypotheses in the close–up view. First, the pan–tilt unit is rotated such that the optical axis of the zoom lens points to the hypothesized object position estimated from the backprojection. From the 3–D information and the approximate size stored in the knowledge–base we can now estimate the zoom position which is appropriate for a close–up view of the object. In [40, p. 45] three methods are listed to do fovealization technically. The method above using a pan–tilt device on a linear sledge adds a new fourth method to this list.

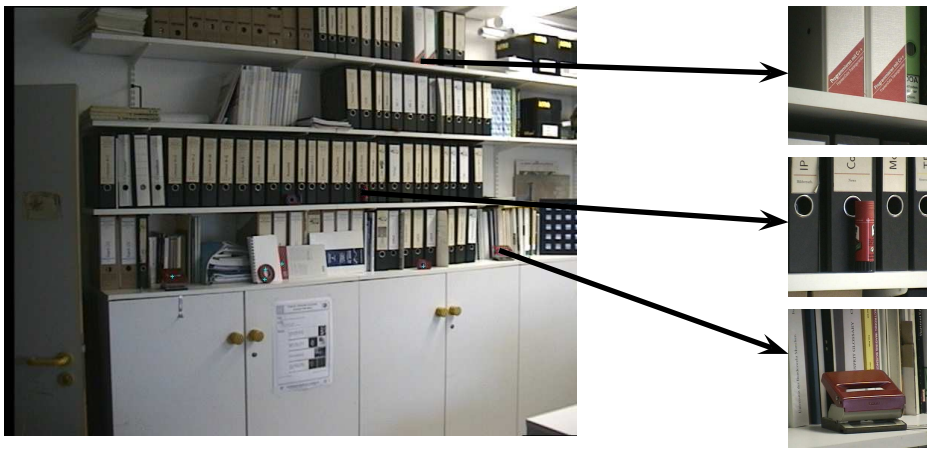


Figure 7. Scene overview and hypotheses for objects

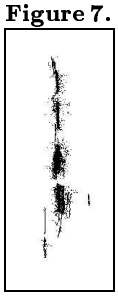
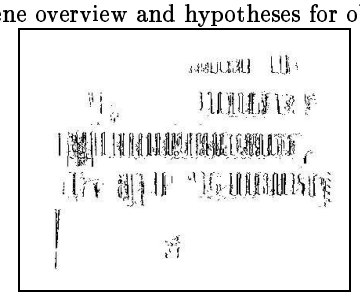


Figure 8. Two projections of 3–D points for Figure 7



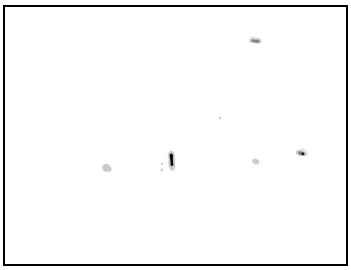


Figure 9. Backprojection of red object

The segmentation of color regions on the detail view of the scene now uses the color segmentation mentioned in Sect. 2.3 and passes these regions to the knowledge–based analysis module; they are collected in a segmentation object attributed by the focal length, distance to the camera, and reliability of the depth value. Results are shown in Figure 10.

The goal of the module for object verification is to find an optimal match of segmented regions in the color close–up views and the gray ovals in Figure 6. The search is directed by the judgment of the concept officeScene which is the goal concept for the subgraph of gray ovals in Figure 6. For the computation we use the attributes, relations, and their judgments in the parts and concrete concepts. In the concept colorRegion we use the Euclidian distance of the mean color vector to a prototype of the color red which is determined by the mean of a histogram, previously. The color space is the intensity–normalized RGB–space.

The similarity of regions for the concepts punch, adhesive_Tape, and gluestick is computed by the attributes height and width of the objects and thus — in the current implementation — depends on the aspect. An extension is envisaged in

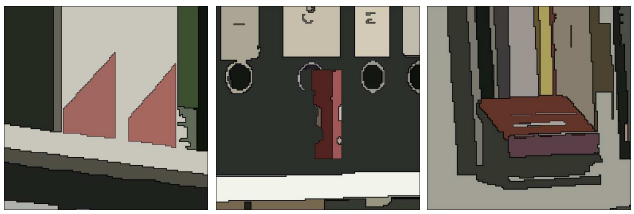


Figure 10. Segmentation of object hypotheses; shown in Figure 7 (right).

combination with the project in Sect. 3.3. The actual scale is determined by the 3–D information computed in Sect. 3.5.

For the search during matching of the regions we use an A* control [23] for the graph search which computes an instantiation path through the network. The concepts are then instantiated bottom–up in the order determined by the path. This means, that no restrictions are propagated during the analysis. The judgment of the nodes in the search tree is equal to the judgment of the goal concept; all judgments of non–instantiated parts are set to one as long as the goal concept is not reached, in order to guarantee an optimistic estimate. The computation of the judgment is deferred to computations of judgments for concepts which in turn use their attributes. The similiarity measure for color regions influencing the judgment is part of the task specific knowledge and can thus be exchanged easily.

In 17 experiments for evaluation of this application we used the punch, glue stick and two tape rollers of identical shape but different size; thus object identification based only on color was not possible. The data driven hypotheses located 77 of 97 objects correctly. To restrict the size of the search tree to 300–600 nodes, we presently use a heuristic judgment function which weights regions of the object's color higher than regions of other colors. The rate of correct verifications of the hypotheses is around 70%. This figure includes the frequent confusions of stapler with the large tape. If we leave this object out, the recognition rate is around 80%. The total time for processing is around 2 min. for one scene on an SGI O2 (R10000, 195 MHz). Large parts of this time are spent for the color segmentation, backprojection, and waiting for the camera to reach a stable position. The convolution in Figure 4 was replaced by a 21×21 median filter to obtain good local maxima which needs several seconds per image to be computed.

4.2 Autonomous Mobile Systems

Autonomous mobile systems ideally can be used to integrate different aspects of computer vision into one common application. In the project DIROKOL[2] a service robot will supply employees in hospitals by so–called fetch and carry services. The system is equipped with a couple of classical robotic sensors (sonar,

[2] The project is funded by the Bavarian Research Foundation (Bayerische Forschungsstiftung).

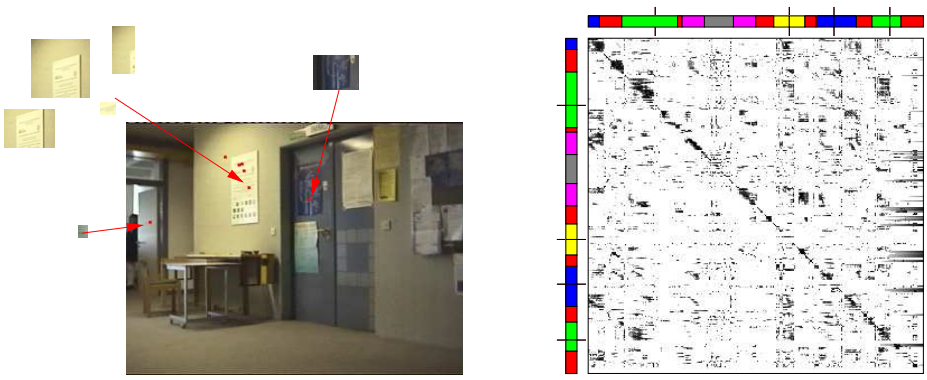


Figure 11. Left: Sample landmark defintion. Right: Confusion matrix of the positions during a movement through our lab showing the quality of self–localization. Dark areas indicate high correspondence between two positions. Bars indicate turning points.

infrared, etc.), a four finger hand [3] mounted on a robot arm, and with a stereo camera system for visual navigation, object recognition and active knowledge–based scene exploration. Similar to the application described in Sect. 4.1 semantic networks are used. The complexity of dynamic scenes in autonomous mobile systems applications makes it necessary, to apply probabilistic knowledge representation schemes (Markov models, dynamic Bayesian networks) as well. For these approaches it is more natural, to acquire knowledge during a training step automatically. The goal for the future is, to combine the classical approach of semantic networks with probabilistic representation schemes to have both, a hard–wired explicit knowledge base and an adjustable, trainable part, especially for the active components in an active vision system.

Actual work on probabilistic methods in this context has been performed on automatic natural landmark definition and localization with application to visual self–localization based on stochastic sampling [10]. The landmark definition is based on color histograms (Sect. 3.2) which has been extended to contain additional information on the distribution of the pixels position of a certain color bin as in [17]. The extented color histogram is implemented by re–using code of the classical histogram by deriving a new class with additional information about the distribution in the image plane of pixel falling into a certain histogram bin.

The experiments have shown, that on average 4.2 landmarks (variance: 11.19) have been automatically defined for each position (see Figure 11, left). With these landmarks self–localization by a stochastic sampling have been evaluated. The computation time for landmarks definition and self–localization is 230 msec. and 14 msec. for 100 samples, respectively. In Figure 11, right, results are shown by using a confusion matrix of the positions in our laboratory. As expected, the diagonal of the matrix has many entries, indicating a good self–localization ability. Also, similar positions seen during the movement are recognized (dark entries

on certain side diagonal elements). It is worth noting, that these results are only based on color histograms without any 3D knowledge or other active sensors. Actually neither dependencies between landmarks seen at a certain position nor context in time is taken into account. This is done in our actual work.

5 Conclusion and Future Work

Most knowledge–based systems in computer vision represent only objects or scenes in the knowledge base. Here, an approach to represent actions and strategies in the same formalism was presented. This will be most important in active vision and for autonomous systems.

We described our system architecture for active image understanding which is implemented in an object–oriented fashion. Classes and objects encapsulate data, devices, and operators as well as the knowledge base. The applications presented make use of a subset of the modules and prove that the approach is feasible for knowledge–based analysis as well as real–time processing. We argued that this programming paradigm simplifies solving image analysis tasks. Object–oriented programming is preferred to genericity for hierarchies of operator classes; assuming that the task of an operator is not trivial, the overhead imposed by this implementation scheme is negligible. Genericity is used for regular data structures such as for pixel access in image matrices.

In our application for active object recognition and scene exploration, we used a knowledge base represented as a semantic network of camera actions and object models. In our application for autonomous mobile systems, color was used for landmark detection as a first stage of knowledge–based navigation and operation. Both applications share various modules for image processing and analysis.

More modules exist in our system which can be combined with the systems described in Sect. 4. To give an example, one of the color normalization algorithms described in [6] will be selected for each object in the application in Sect. 4.1; this selection will be annotated to the object in the knowledge base. Moving objects in an object room will be tracked using the methods of Sect. 3.7 after they have been localized. An integration of statistical models (Sect. 3.3) and semantic networks (Sect. 4.1 [19, 27]) will be used for holistic object recognition as in [21]. These models are invariant of the aspect. To apply our appearance based statistical method (Sect. 3.3 [27]), a fixed size of the object within the image is required. Therefore, we need exact knowledge about depth to zoom to the appropriate size. The changing perspective distortions at different object distances are neglected.

References

1. More references to our own work can be found in the publication section of our web site.

2. R. B. Arps and W. K. Pratt, editors. *Image Processing and Interchange: Implementation and Systems*, San Jose, CA, 1992. SPIE, Proceedings 1659.
3. J. Butterfass, G. Hirzinger, S. Knoch, and H. Liu. Dlr's multisensory articulated hand. In *IEEE International Conference on Robotics and Automation*, pages 2081–2086, Leuven, Belgium, 1998.
4. I. C. Carlsen and D. Haaks. IKS^{PFH} — concept and implementation of an object–oriented framework for image processing. *Computers and Graphics*, 15(4):473–482, 1991.
5. D. Crevier and R. Lepage. Knowledge-based image understanding systems: A survey. *Computer Vision and Image Understanding*, 67(2):161–185, August 1997.
6. L. Csink, D. Paulus, U. Ahlrichs, and B. Heigl. Color Normalization and Object Localization. In Rehrmann [29], pages 49–55.
7. J. Denzler. *Aktives Sehen zur Echtzeitobjektverfolgung*. Infix, Aachen, 1997.
8. J. Denzler, B. Heigl, and H. Niemann. An efficient combination of 2d and 3d shape description for contour based tracking of moving objects. In H. Burkhardt and B. Neumann, editors, *Computer Vision - ECCV 98*, pages 843–857, Berlin, Heidelberg, New York, London, 1998. Lecture Notes in Computer Science.
9. J. Denzler and H. Niemann. Real–time pedestrian tracking in natural scenes. In G. Sommer, K. Daniliidis, and J. Pauli, editors, *Computer Analysis of Images and Patterns, CAIP'97, Kiel 1997*, pages 42–49, Berlin, Heidelberg, New York, London, 1997. Lecture Notes in Computer Science.
10. J. Denzler and M. Zobel. Automatische farbbasierte Extraktion natürlicher Landmarken und 3D-Positionsbestimmung auf Basis visueller Information in indoor Umgebungen. In Rehrmann [29], pages 57–62.
11. S.J. Dickinson, H.I. Christensen, J.K. Tsotsos, and G. Olofsson. Active object recognition integrating attention and viewpoint control. *Computer Vision and Image Understanding*, 67(3):239–260, September 1997.
12. B. Draper, A. Hanson, and E. Riseman. Knowledge-directed vision: Control, learning and integration. *Proceedings of the IEEE*, 84(11):1625–1637, Nov 1996.
13. V. Fischer and H. Niemann. A parallel any–time control algorithm for image understanding. In *Proceedings of the* 13^{th} *International Conference on Pattern Recognition (ICPR)*, pages A:141–145, Vienna, Austria, October 1996. IEEE Computer Society Press.
14. T. Garvey. Perceptual strategies for purposive vision. Technical report, SRI AI Center, SRI International, Menlo Park, 1976.
15. R. M. Haralick and V. Ramesh. Image understanding environment. In Arps and Pratt [2], pages 159–167.
16. M. Harbeck. *Objektorientierte linienbasierte Segmentierung von Bildern*. Shaker Verlag, Aachen, 1996.
17. B. Heisele, W. Ritter, and U. Kreßel. Objektdetektion mit Hilfe des Farbfleckenflusses. In V. Rehrmann, editor, *Erster Workshop Farbbildverarbeitung*, volume 15 of *Fachberichte Informatik*, pages 30–35, Universität Koblenz–Landau, 1995.
18. A. Hoogs and D. Hackett. Model-supported exploitation as a framework for image understanding. In *ARPA*, pages I:265–268, 1994.
19. J. Hornegger. *Statistische Modellierung, Klassifikation und Lokalisation von Objekten*. Shaker Verlag, Aachen, 1996.
20. M. Kass, A. Wittkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 2(3):321–331, 1988.
21. F. Kummert, G. Fink, and G. Sagerer. Schritthaltende hybride Objektdetektion. In E. Paulus and F. Wahl, editors, *Mustererkennung 1997*, pages 137–144, Berlin,

September 1997. Springer.
22. C.-E. Liedtke, O. Grau, and S. Growe. Use of explicit knowledge for the reconstruction of 3-D object geometry. In V. Hlavac and R. Sara, editors, *Computer analysis of images and patterns — CAIP '95*, number 970 in Lecture Notes in Computer Science, Heidelberg, 1995. Springer.
23. H. Niemann. *Pattern Analysis and Understanding*, volume 4 of *Springer Series in Information Sciences*. Springer, Heidelberg, 1990.
24. 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)*, 9:883–905, 1990.
25. D. Paulus and J. Hornegger. *Pattern Recognition of Images and Speech in C++*. Advanced Studies in Computer Science. Vieweg, Braunschweig, 1997.
26. D. Paulus and H. Niemann. Iconic–symbolic interfaces. In Arps and Pratt [2], pages 204–214.
27. J. Pösl, B. Heigl, and H. Niemann. Color and depth in appearance based statistical object localization. In H. Niemann, H.-P. Seidel, and B. Girod, editors, *Image and Multidimensional Digital Signal Processing '98*, pages 71–74, Alpbach, Austria, July 1998. Infix.
28. J. R. Rasure and M. Young. Open environment for image processing and software development. In Arps and Pratt [2], pages 300–310.
29. V. Rehrmann, editor. *Vierter Workshop Farbbildverarbeitung*, Koblenz, 1998. Föhringer.
30. R. Rimey and C. Brown. Task–oriented Vision with Multiple Bayes Nets. In A. Blake and A. Yuille, editors, *Active Vision*, pages 217–236, Cambridge, Massachusetts, 1992.
31. R. Ronfard. Region-based strategies for active contour models. *International Journal of Computer Vision*, 13(2):229–251, 1994.
32. S. J. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1995.
33. G. Sagerer and H. Niemann. *Semantic Networks for Understanding Scenes*. Advances in Computer Vision and Machine Intelligence. Plenum Press, New York and London, 1997.
34. J. Shi and C. Tomasi. Good features to track. In *Proceedings of Computer Vision and Pattern Recognition*, pages 593–600, Seattle, Washington, Juni 1994. IEEE Computer Society Press.
35. M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, November 1991.
36. F. Thomanek and E.D. Dickmanns. Autonomous road vehicle guidance in normal traffic. In *Second Asian Conference on Computer Vision*, pages III/11–III/15, Singapore, 1995.
37. S. Ullman. *High-Level Vision: Object Recognition and Visual Cognition*. MIT Press, Cambridge, MA, 1996.
38. V.V. Vinod and H. Murase. Focused color intersection with efficient searching for object extraction. *Pattern Recognition*, 30(10):1787–1797, October 1997.
39. R.G. Willson. *Modeling and Calibration of Automated Zoom Lenses*. PhD thesis, Carnegie Mellon University, Pittsburgh, 1994.
40. Y. Yeshurun. Attentional mechanisms in computer vision. In V. Cantoni, editor, *Artificial Vision, Human and machine perception*, pages 43–52. Plenum Press, New York, 1997.