Videre: Journal of Computer Vision Research

Quarterly Journal

Winter 2000, Volume 1, Number 4

The MIT Press

Article 1

Active Knowledge—Based Scene Analysis

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

Videre: Journal of Computer Vision Research (ISSN 1089-2788) is a quarterly journal published electronically on the Internet by The MIT Press, Cambridge, Massachusetts, 02142. Subscriptions and address changes should be addressed to MIT Press Journals, Five Cambridge Center, Cambridge, MA 02142; phone: (617) 253-2889; fax: (617) 577-1545; e-mail: journals-orders@mit.edu. Subscription rates are: Individuals $30.00, Institutions $125.00. Canadians add additional 7% GST. Prices subject to change without notice.

Active Knowledge—Based Scene Analysis

D. Paulus, U. Ahlrichs[1], B. Heigl, J. Denzler[2], J. Hornegger[2], M. Zobel[2], H. Niemann[2]

We present a modular architecture for image understanding and active computer vision that 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; and the semantic network formalism is used to represent camera actions that are needed in explorative vision.

We use 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, and color segmentation is used to compute object features which are relatively insensitive to small aspect changes.

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

1. paulus@informatik.uni-erlangen.de

2. Lehrstuhl für Mustererkennung (LME, Informatik 5) Martensstr. 3, Universität Erlangen-Nürnberg, 91058 Erlangen http://www5.informatik.uni-erlangen.de

1 Introduction

Autonomous mobile systems with visual capabilities are a great challenge for computer vision systems because they require skills for the solution of complex image-understanding problems, such as driving a car [52] or exploring a scene [56]. In this contribution we present a vision system that 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 that navigate and act based on visual information, such as the system described by Schlegel et al. in "Integrating Vision-Based Behaviors with an Autonomous Robot." Autonomous systems need an explicit representation of camera actions and search strategies. A literature review in [8] on the topic of knowledge-based image analysis gives a comprehensive discussion of the state of the art. Image analysis systems have also have been reported, for example, in [29, 30] and in the other contributions to this volume. In [29] 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, camera actions, and strategies to provide flexible and exchangeable strategies for active vision and scene exploration.

Configuring or implementing image-processing systems is a time-consuming task that requires specialized knowledge on the effects of image-processing algorithms as well as knowledge about the implementation and interfaces. Clearly, software engineering is required for the application programmer of image-processing systems. A software system for image understanding usually has a considerable size. The major problem in software design of general imaging systems are that, on the one hand, highly runtime-efficient code and low-level access to hardware is required, and that, on the other hand, a general and platform-independent implementation is desired that 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 and genericity which can help simplifying code reuse; if applied properly, object orientation unifies interfaces and simplifies documentation by the hierarchical structure of classes. A key mechanism of object-oriented programming is polymorphism [6]. In this article, we give examples of how polymorphism can simplify image-processing programs and how it can keep the required efficiency. Genericity provides an alternative solution to software engineering problems [28]. Both concepts are available in C++ [50]. Object-oriented programming has been proposed for image processing and computer vision by several authors, in particular in the context of the image-understanding environment [21]; this approach is mainly used to represent data. We also use object-oriented programming for operators and devices.

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

The goal of our example application in section 4.1 is to explore an office room. Objects are hypothesized in the image using color. Their 3-D position is estimated from a coarse 3-D map computed from trajectories of colored points that are tracked during a translational motion of the active camera. The objects are chosen in such a way that they cannot be distinguished solely by their color. Closeup views are captured and segmented into color regions and 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 section 4.2, we describe a recent research project in the area of visual guided autonomous mobile systems. 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 section 6.

2 System Architectures

The need for a flexible, knowledge-based, computer vision system with real-time capabilities at least for low-level processing lead to An image analysis system (ANIMALS, [35, 37, 38]) which is implemented in C++. It provides modules for the whole range of algorithms from low-level sensor and actor control up to knowledge-based analysis.

2.1 Data Flow for Knowledge-Based Analysis

The general problem of image analysis is to find the optimal description of the input image content that is appropriate to the current problem. Sometimes, this means that the most precise description of the image data has to be found; in other cases, a less exact result that can be computed faster will be sufficient. In many image-analysis problems, objects have to be found in the images and described by terms that fit to the application.

These general problems can be divided into several subproblems. After an initial preprocessing stage, images are usually segmented into meaningful parts. Various segmentation algorithms create initial symbolic descriptions of the input data [31] which we call *segmentation objects* [35]. Models in a knowledge base containing expectations on the possible scene in the problem domain are matched with segmentation objects to provide a final symbolic description. This is best achieved if the representation for the models is similar to the structure of segmentation results. A definition fulfilling this similarity constraint is shown for segmentation objects and for semantic networks in [31].

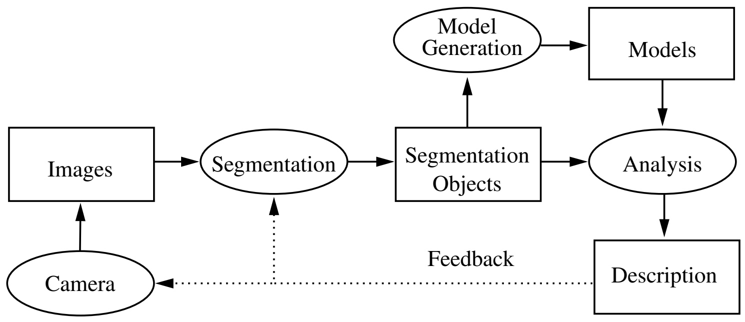


Figure 1. Data flow in an image-analysis system.

Modern architectures for image analysis incorporate active components such as pan/tilt devices or cameras on a robot. Such devices lead to feedback loops in the course from image data to image descriptions. A top-level view of the main components in our image analysis is shown in figure 1; data is captured and digitized from a camera and transformed to a description that may cause changes in camera parameters or tuning of segmentation parameters. Models that are collected in the knowledge base are created from segmentation results (in section 3.3) or at least have similar structure (in section 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 depending on the state of analysis (or the values of the camera and lens can be modified). Changes to parameters of the image-input process, selection of 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 combine their efforts for a common image-understanding environment [21]. The system was planned as a basis for image processing, computer vision, and knowledge-based image analysis. The system covers all of imaging with many applications; because there are many contributors and the goal is to use an object-oriented implementation, a large set of ideas has to be united into a common hierarchy of classes. The design goals are object-oriented programming, graphical interfaces, extensibility and program exchange, and performance evaluation.

Real-time processing was explicitly excluded from the original goals [21, p. 160]. In the present version, no classes are provided for devices such as camera. This environment is applied, for example, in [24] to the analysis of satellite images.

The other widely used system in Khoros [41], which provides a nice graphical user interface for image-processing systems. The data-flow architecture of the system facilitates complex combinations of image-processing modules. Data structures beyond images and matrices are not available to the user. Therefore, knowledge-based processing is not integrated in the system.

Real-time processing as well as symbolic data structures are crucial for active exploration. Both systems would have to be modified or extended to be used for our purpose of active scene exploration and robotics. Our system, which is described next, provides general interfaces to devices used in active vision as well as interfaces to knowledge-based analysis. The design goals noted above are valid for this system as well. However, the graphical user interface is decoupled from the algorithmic part because it is not required in some real-time applications.

2.3 Object-Oriented Design for Image Analysis

The algorithms and data structures of our system are implemented in hierarchy of picture processing objects (HIPPOS, written as ἵππς [35, 38]), an object-oriented class library designed for image analysis that is based on the NIHCL C++ class library [20]. In [38], the data interfaces were defined as classes for the representation of segmentation results. The segmentation object (SegObj in figure 2) [38] provides a uniform interface between low-level and high-level processing; its components can be arbitrary objects resulting from point, line, region, or surface segmentation. Images, segmentation objects, and additional classes for representation of segmentation results (such as chain codes, lines, polygons, and circles) are derived from the HipposObj (top of figure 2). In [4], this system is extended to a hierarchical structure of image processing and analysis classes and objects (cmp. [7]). Objects are the actual algorithms with specific parameter sets that are also objects (OperatorOpt in figure 2, [4]). Classes as implementation of algorithms are particularly useful, when operations require internal tables that increase their efficiency (because 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 (for example, *EdgesToSegObj* in figure 2), gap closing, and corner and vertex detection. for each processing step (implemented here as a class), there exists a large variety of choices in the literature. When the whole sequence of operations is subjected to optimization—either manually or automatically—it is crucial to have similar interfaces to exchangeable single processing steps, such as several corner-detection algorithms. As shown here, this is greatly simplified by object-oriented programming and polymorphic interfaces. 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 segmentation objects.

Many operations in image processing can be structured hierarchically in a straightforward manner. Some edge-detection operators can be specialized to edge mask operators; one of them is the Sobel operator class. Filters can be linear or nonlinear; morphological operations are one type of nonlinear operators; the median is one special case of this type. Such operators can be implemented in a hierarchy of classes for operations in a straightforward manner (cmp. [7, 17]). Objects are the actual algorithms with specific parameter sets that are also objects [4]. Classes as implementation of algorithms are particularly useful when operations require internal tables that increase their efficiency. Using inheritance and virtual functions in C++ for the interface declaration, possible new derived classes are forced to obey the interface definition.

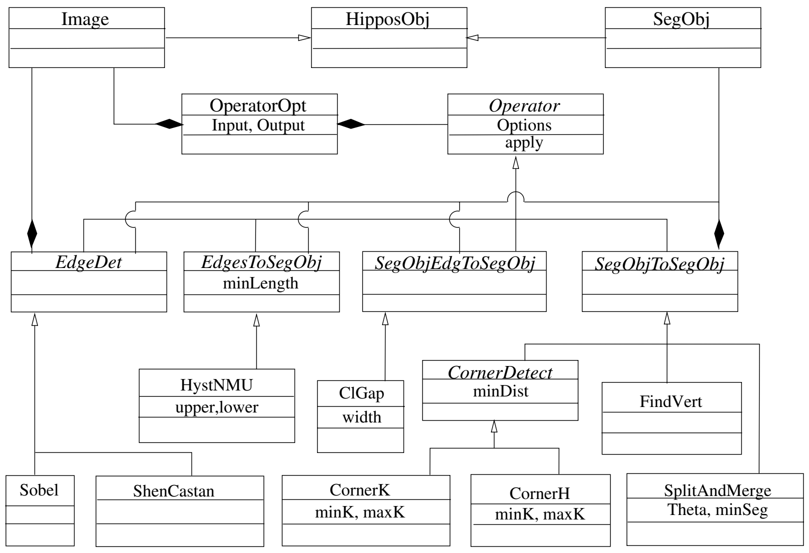


Figure 2. Part of a class hierarchy for image operators from [4] in UML notation

The call to a newly integrated operation in an existing program can thus be performed with small changes—or even without the need of changing the existing program syntactically. This greatly simplifies the extension and modification of image-analysis systems. When, instead of operator classes, function definitions collected in an interface definition, new implementation for similar purposes (such as a new edge detector as an alternative to an existing one) may change the type and number of parameters compared to another existing function. If this function is now to be integrated into an existing program, the old calling syntax has to be replaced by the new one. If it is desired to have the alternative of calling one or the other function, the code will not only have to be changed but extended. Both is mostly avoided by the operator hierarchy, yet providing exactly the same functionality without loss of runtime efficiency. An example is given in section 3.4.

Several segmentation algorithms implemented in our system operate on graylevel, range, or color images using the interfaces provided by the segmentation object. To give an example, the application in section 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 can be summarized as follows.

- 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 [4]).

1. Of course, this is not always possible.

- Information about the operator that is actually used is available. For example, a program may just reference a filter object; during runtime, it will be decided which concrete filter should be used. When evaluating the performance of a system, several possible implementations of one algorithm can be tested, and the program reports which implementation was used.

In section 2.4, we argue that these advantages produce no additional runtime overhead. Similar hierarchies as the one in figure 2 exist for filters and for region-based segmentation.

2.4 Software Experiments and Experience

Our system is compiled and tested for various hardware platforms, including HP (HPUX 9.07 and 10.20, both 32-bit and 64-bit versions), PC (Linux), SGI (IRIX 6.5) and Sun (Solaris).

The NIHCL class library served as a tool for most of the 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 in [38][2] 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 nontrivial (as is the case in the examples). 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 runtime overhead is provided as described in [35]. The application in section 4.2 shows that real-time processing for low-level vision is possible using this approach.

3 Modules of ANIMALS

Various modules for common computer vision algorithms are provided in AMIMALS. These modules were implemented for several applications. Because all segmentation algorithms use the common interface for data exchange provided by the segmentation objects, the compatibility is high. This flexibility first requires additional effort for portable software. In the long run, it reduces the effort of software maintenance. In addition to data algorithms that were represented by classes in section 2.1, devices are also encapsulated by classes in section 3.1.

3.1 Sensors, Actors, and Calibration

Due to the variability of the hardware connected with an image-analysis software system, interfaces to several types of frame grabbers, cameras, lenses, stereo devices, and so forth have to be provided by the system. In order not to burden the programs by a large number of switches, all these special devices are encapsulated as classes that share their common behavior in a base class. Whereas the internal implementation may be sophisticated and thereby provides the required performance, the user interfaces for these classes are simple in C++. For example, the vergence axis of the stereo head is equipped with a stepper motor, as is the pan axis of a surveillance camera. Both motors have completely different control syntax. They can, however, be encapsulated in an abstract

2. The first classes in C++ for this system were written in 1988; some of them are still used.

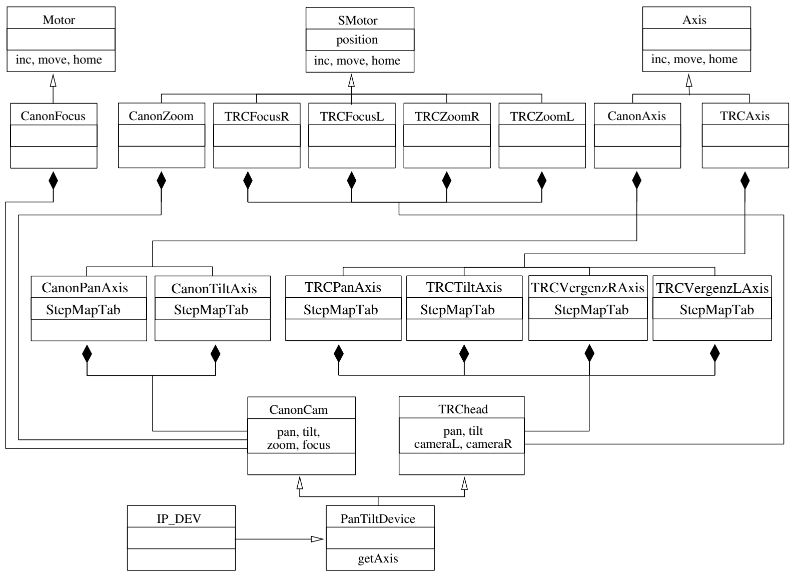


Figure 3. Class hierarchy for motors used in active vision systems

"stepper motor object" (SMotor in figure 3), which can be assigned the desired position by the assignment operator in C++. Thus, both axes are now addressed by the same syntax, and the functions using them are much more portable. An example for the efficient use of an generic implementation for an actor, which relates the two applications in section 4.1 and section 4.2, is given at the end of section 4.2.

Initialization of frame grabber cards usually requires many function calls and choices from many parameters. This is easily hidden in a class for a frame grabber object which provides default values. The basic functionality such as the selection of the image size is provided by an abstract base class that releases the application programmer from the machine-dependent part of the code or image input.

Calibrating zoom lenses is a complex problem because the motors used in consumer cameras will not always lead to accurate positions. For the object-localization system in section 4.1, we need the enlargement factor related to a given position of the zoom stepper motor. The calibration according to [55] is hard to do fully automatically, because 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 an image's lines, defined by two corresponding points each. The corresponding points are found by tracking color points (section 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 robust because local errors accumulate. We took the logarithm of the factors and approximated these values by a regression line (figure 4 (left)). The enlargement factors are computed by the exponential function (figure 4 (right)). The focal length of the minimal zoom position has been determined using a calibration pattern. All other focal lengths equal the product of the enlargement factor and the minimal focal length. In the experiments in section 4.1, this method provided reasonable accuracy.

The results of the calibration algorithm is recorded together with the stepper motor object (SMotor) which is part of a pan/tilt unit (a Canon

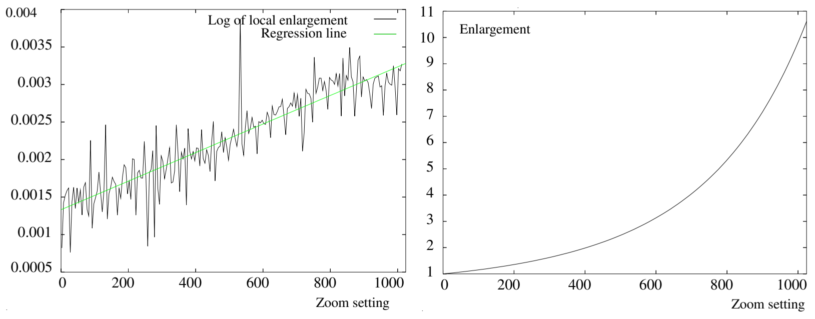


Figure 4. Logarithm of local enlargement factors and estimated enlargement factors

Figure 5. Localization of objects by histogram backprojection according to [51].

```
Given: image histogram $T = [T_l]l = 1 \dots N_L$ of an object,
Wanted: object position $(i_t, j_t)$
Compute color histogram $S = [S_l]l = 1 \dots N_L$ of given image
FOR Each bin $l \in \{1, \dots, N_L\}$
    $Q_l = \min\{\frac{T_l}{S_l}, 1\}$ (compute ratio histogram $Q = [Q_l]l = 1 \dots N_L$)
FOR All positions $(i, j)$ in the image
    $g_{i,j} := Q_{\zeta(f_{i,j})}$, where $f_{k,j}$ denotes the color vector at position $(i, j)$
$h := D * g$, where $*$ denotes convolution
$(i_t, j_t) := \text{argmax}_{i,j}(h_{i,j})$
```

pan/tilt device CanonCam in figure 3, which has a tilt axis Canontilt-Axis). This axis contains the calibration information StepMap as a table mapping the step number to an angle). Persistent storage of this object records all available calibration information.

3.2 Color

Many algorithms in ANIMALS operate on color images as well as on graylevel images. Color provides simple cues for object localization because it is insensitive to aspect changes of moderate angles. Color histograms are used in [51] to form hypotheses for the position of an object in the 2-D projection in an image. A color image $[f_{ij}]1 \leq i \leq M, 1 \leq j \leq N$ is searched for an object which is characterized by its histogram $T = [T_l]l = 1 \dots N_L$ in some quantization N_L. In addition, the approximate size of the object in the image is needed for the algorithm; this size is represented by a mask D_r covering the object. The function ζ maps a color vector f to an index $l = 1 \dots N_L$ in the histogram and thus permits us to use arbitrary quantizations. The principle of the algorithm is shown in figure 5. The histogram S of the image is used to produce an output image B of the size of the input image; internally, an intermediate image A of the same dimension is used.

Color histograms for different 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 5 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], [25], statistical model generation, classification, and localization is based on a set of projected feature vectors O. We assume that the image

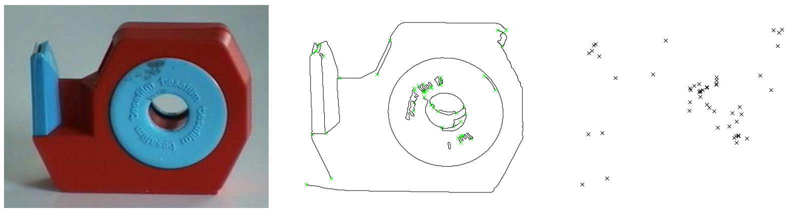


Figure 6. Object (left) segmented to lines and vertices (center) and points (right)

$[f_{i,j}]1 \le i \le M, 1 \le j \le N$ is transformed into a segmentation object of 2-D feature vectors $O = \{O_k \in \mathbb{R}^2 | 1 \le k \le N_m\}$ consisting of points (such as corners or vertices) or lines that can be detected by several combinations of segmentation operators from the class hierarchy shown in figure 2 (for example, by the operator *EdgeToSegObj*). Results are shown in figure 6. 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_k is defined by the density $p(O \mid B_k, R, t)$, and discrete priors $p(\Omega_k)$, $1 \le k \le K$, if only single objects appear, or $p(\Omega_{k_1}, \Omega_{k_2}, \ldots, \Omega_{k_q})$ for multiple object scenes; the priors for the occurrences of an object are estimated by their relative frequencies in the training samples. The parameter R denotes rotation and projection from the model space to the image plane, and t denotes translation. The parameter set B_k contains the model-specific parameters that model the statistical behavior of features as well as the assignment. In the special case of segmented point features, B_k statistically models the accuracy and stability of the detected object points.

Let us now assume that the parametric density of the model feature c_{k,l_k} corresponding to O_k is given by $p(c_{k,l_k} \mid a_{k,l_k})$, where a_{k,l_k} are the parameters for the density function of the feature c_{k,l_k}. A standard density transform results in the density $p(O_k \mid a_{k,l_k}, R, t)$ which characterizes the statistical behavior of the feature 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 for which the parameters R and t are known from calibration of sensor and actor. 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 maximum likelihood estimator. For object recognition and pose estimation, the parameters R and t are unknown as well and have to be estimated. Optimization classes (section 3.8) were developed for this application [25].

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 [53] 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 1,600 randomly chosen views of objects, the

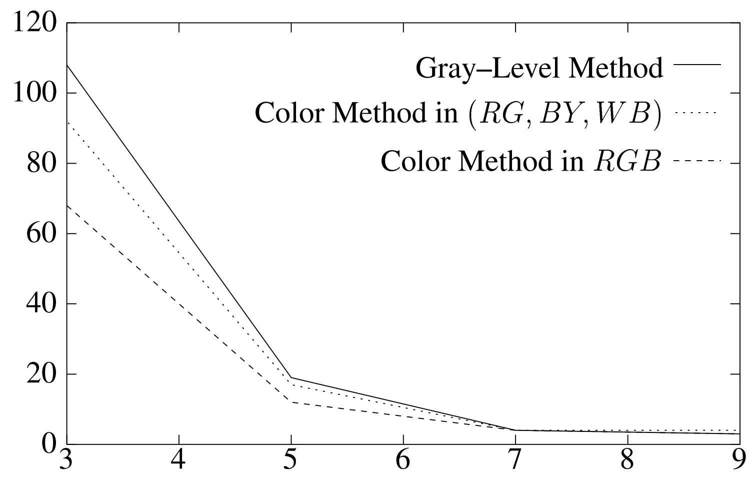


Figure 7. Number of points lost during tracking by different methods; 1st: using the graylevel method; 2nd: using the color method in the *RG*, *BY*, *WB* color space; 3rd: using the color method in *RGB*

recognition rates for the statistical approach were in the range of 95% for 2-D recognition.

3.4 Point Tracking

In the previous section, interesting points were detected in an image and used as features for object recognition. In section 3.1, points also had to be detected and then tracked in an image sequence. The basic idea in [49] is to select those points in a sequence of graylevel images that exhibit features for stable tracking. Thereby the two questions formerly posed independently are now combined to one problem: the question of 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 spatiotemporal gradient in the Taylor expansion. By applying this method iteratively, the displacement is determined in subpixel accuracy.

We extended this different 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 corresponding points only to a given orientation assuming pure translational camera movement. In this case, the Jacobian matrix is substituted by the directional derivative for the orientation vector of the epipolar lineup this line links each pixel to the epipole. In the case of restricted search space, the criterion for point selection results in a large gradient in the search direction. Therefore, a large number of points can be selected.

The criterion for tracking has been adapted for both extensions. The use of color values shows that more points can be selected and the tracking is more stable (figure 7). Better and more robust tracking is possible in *RGB* than, for example, in the perceptually motivated *RG*, *BY*, *WB* color space or in graylevel image sequences; the number of points lost during tracking in *RGB* is more than 20% smaller than for graylevel images [22].

The computing times on an SGI O_2 for a window size of 5×5 and an image sequence of 31 frames in PAL resolution are 90 sec. for graylevel images and 187 sec. for color images.

The times for feature selection and point tracking increases linearly with the number of pixels within the window. Therefore, it is a proper decision to choose a small window size, color tracking, and the *RGB* color space for reliable results and fast computation.

The general class implementing the point tracker has two derived classes for tracking in color and in graylevel. When the tracker is applied to an image, no syntactical difference can be observed between the two. Only when the object representing the tracking operation is initialized by the constructor in C++ do the arguments differ slightly.

3.5 3-D Reconstruction

The module for point tracking described in section 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 and the displacement is small, no such correspondence problem (as in stereo vision) has to be solved explicitly. If the optical axis is orthogonal to the motion vector, stationary points in the scene lie on a straight line in the spatiotemporal domain [26]. The regression line through the trajectory of a point is computed to eliminate errors and to obtain subpixel accuracy. The inclination of this line determines the stereo disparity of the point. The regression error can be used as a measure for the certainly of the disparity value. The reliability of the range value is proportional to the disparity multiplied by the reliability of the disparity multiplied. The approach is also applicable to the situation in which the optical axis of the camera is not perpendicular to the motion vector. Then, each feature point can be transformed to coordinates of a virtual camera with perpendicular orientation. The algorithm accepts an abstract point-tracking operator (object); that is, the same algorithm can be used for graylevel as well as color images. Camera calibration and orientation are known from objects describing the camera properties.

Results of reconstruction are shown in figure 13. the time for reconstructing depth just depends on the time for tracking points (section 3.4).

3.6 Knowledge Base

In section 3.3, we represented single objects by model densities. Structural knowledge about scenes, objects, as well as camera actions and strategies can also be represented in a semantic network. Semantic networks have been applied successfully—for example, in [29, 30, 33]—to pattern-analysis problems. They provide an intuitive formalism for the representation of objects and facts that are required for computer vision. Our goal is to continue the work on knowledge representation concerning the semantic network formalism ERNEST [47, 48, 33] by integrating camera actions into the knowledge base. We also reuse the existing control algorithms for the semantic network. Examples of alternative solutions for the explicit representation of actions are and–or trees [19] hidden Markov models [44], or Bayesian networks [46] (as also presented in the other contributions of this volume).

Using the notation and structure defined in [31, 48], a semantic network is a directed acyclic graph consisting of nodes and labeled edges. The nodes are concepts composed of attributes and are used for the representation of objects and camera actions. The edges denote specialization that implies inheritance of attributes, part–of relations, and the concrete link that joins entities of a different level of abstraction. Since image processing is never exact, all entities in the semantic network have an attached certainty that measures the degree of certainty. The certainty values are used to guide either an A* graph search algorithm or another ??? process. The expansion of nodes in the A* search tree during analysis

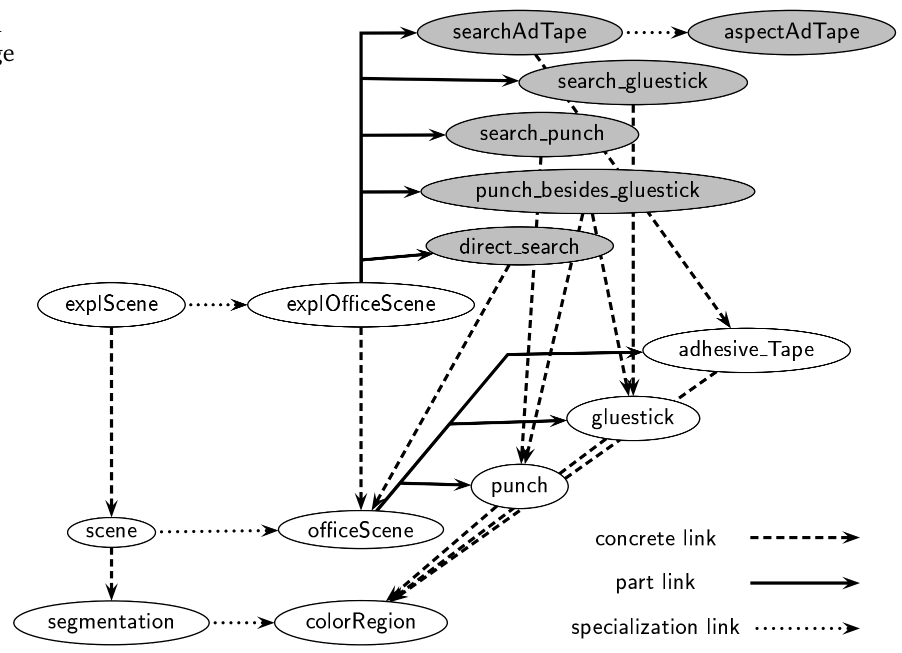


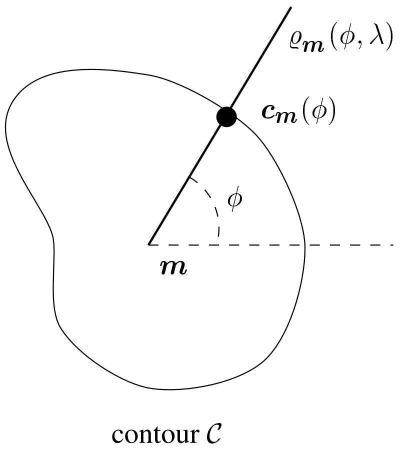
Figure 8. Combined representation of actions and objects in a knowledge base

follows six inference rules [48; that is, the control works independently of certainty functions and of the particular semantic network used in an application. alternative possibilities for control strategies such as Markov decision processes are discussed in [16]; we also provide a second control algorithm called *parallel iterative control* [32], 18], which is based on combinatorial optimization.

Figure 8 shows a typical semantic network. The lower part of the network contains objects that are used in the application in section 4.1. the shaded ovals represent different camera actions in which each competing action (`direct_search` for a search without intermediate object [19], `punch_besides_gluestick` for a search for a punch using the intermediate object glue stick, and so on) is linked to the concept `explOfficeScene` by competent part links; the control resolves these alternatives that are collected in so-called sets of modality. (See 33 for details). In other words for each alternative a new node in the search tree is generated during analysis. Concrete links relate the camera actions (such as `direct_search`) to the knowledge on objects. An instantiation of the concept `direct_search` calculates a new value for the pan position attribute. The same holds for `punch_besides_gluestick`. both instances have associated certainty, which are now used by the control in combination with the certainty 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 certainties of concepts get worse, the search tree node that contains the instance of the other camera movement becomes the node with the highest certainty 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 provides classes for all major items in the semantic network. A formal language definition is used to generate a parser

Figure 9. Representation of a contour point by active rays



and code generator that creates C++ code from the knowledge-base definition. Two implementations of control strategies—one for A* control and one for iterative-optimizing control—have been used for 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 Contour-Based Real-Time Object Tracking

A simple and thereby powerful method for fast object detection and tracking in image sequences is based on active contours or snakes [27]. A number of control points has to be chosen to 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 [10].

In [10], we proposed the notion of active rays that can also be used for the description of an object's contour; this approach 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 (figure 9). The image is sampled along each ray. The contour point $c_m(\phi)$ in direction ϕ from the reference point m is now located in a 1-D signal: the sampled image date on each ray, at position $\lambda(\phi)$, which corresponds to the distance of the contour point from the reference point m.

Similar to active contours, an internal and external energy is defined for active rays. The contour represented by the function $\lambda^*(\phi)$ is extracted by minimizing the sum of these energies. In contrast to active contours, the energy minimization now takes place along only 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 (figure 10) [13]. Both ideas, active contours and active rays, are represented as classes and related by a common base class; thereby, they share many lines of code.

Compared to active contours, active rays have several advantages for real-time applications. First, due to the representation of the contour points, parameterized by an angle defining the direction of each ray, no crossing can occur in the contour nor can the contour points move along the contour of the objects. Thus, prediction steps can be robustly applied.

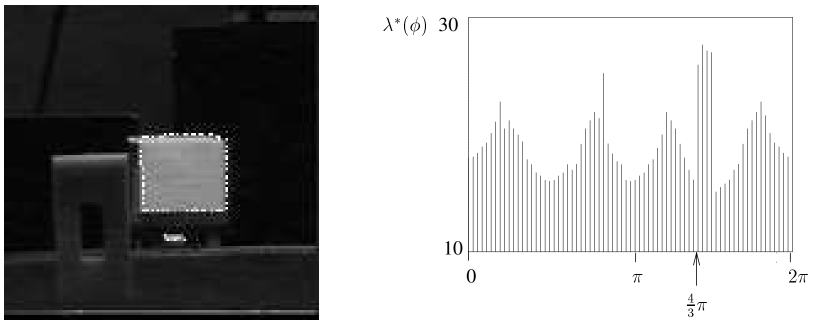


Figure 10. Left: 2-D contour extracted by active rays. Right: 1-D function λ* of the corresponding 2-D contour shown on the left.

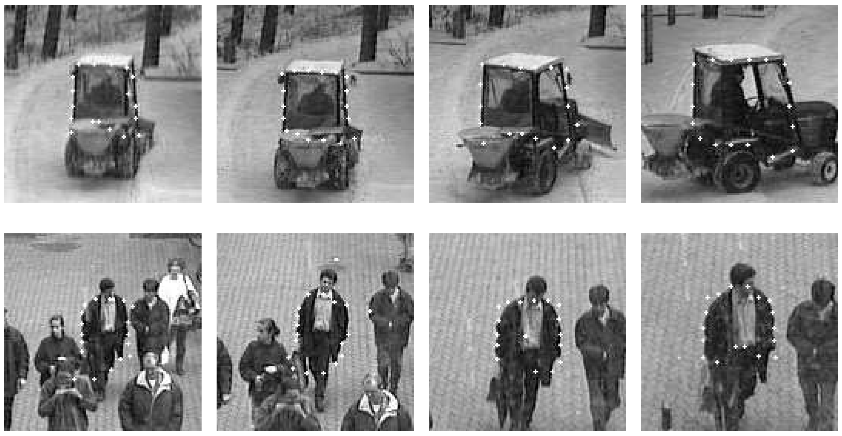


Figure 11. Results for tracking moving objects in real time

Second, an anytime algorithm can be defined that allows for an iterative refinement of the contour depending on the time that 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 computationally expensive task for active contours due to the independent search in the 2-D image plane [45]. Finally, an efficient combination with a 3-D prediction step is possible by using a similar radial representation of 3-D objects [11].

A complete object-tracking system—COBALT (contour-based localization and tracking, [10]) has been implemented in the ANIMALS framework to evaluate the new approach in the area of real-time pedestrian tracking in natural scenes [12]. A pan-tilt camera supervises a place in front of our institute. The application program is independent of this device; either the Sony camera or the TRC head shown in figure 3 can be used. A motion-detection 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 section 3.1 to keep the moving objects in the center of the image (figure 11). In five hours of experiments under different weather conditions, tracking was successful 70% of the time. On an SGI Ones with two R10000 processor, 25 images per second could be evaluated in the complete system that summarizes the image grabbing, tracking, and camera movement. For contour extraction alone using 360 contour points, approximately 7 ms are needed per image.

3.8 Optimization

The solutions of the optimization problems outlined in sections 2.1, 3.3, and 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 are each included in the optimization process.

Probabilistic optimization routines that allow practically efficient solutions for object recognition are discussed in [25]. 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 that is to be optimized. An abstract base for all optimization strategies has an internal variable that 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 the gradient of the function which yields another class in the hierarchy). Procedures that use the function directly are, for example, the combinatorial optimization, the downhill 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 in [25], the algorithm of Fletcher and 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 [25] for the optimization classes were approximately 1 min. for 10,000 calls to the function to be optimized. This class hierarchy for optimization has also been used in speech-analysis applications, naturally without the need to include any image-processing modules; this is an empirical proof for the flexibility of this software approach.

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 section 3 can be effectively combined to build a system that finds objects in an office room. In [54], 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.

Knowledge on objects is represented in [15] by a hierarchical graph that could be formulated as a semantic network; the viewpoint control described there is part of the control algorithm for evaluating the graph. In contrast, [43] 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). In our work, camera actions as well as top-level knowledge are explicitly represented in one semantic network; the control algorithm of the semantic network used to guide the search is independent of the camera control

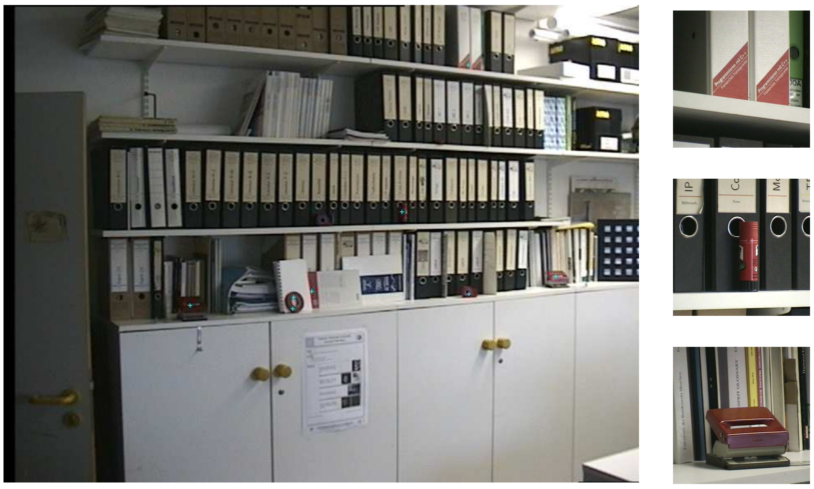


Figure 12. Scene overview and hypotheses for objects

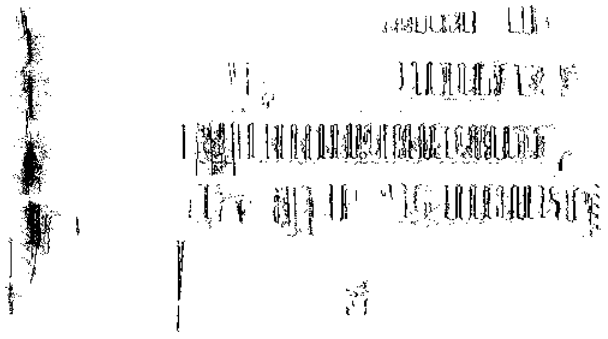


Figure 13. Two projections of 3-D points for figure 12

Real-world office objects (the tape roller in figure 6, the glue stick and punch in figure 12) are used in our experiments. The objects occupy only a few pixels in the wide -angle views captured initially (figure 12, 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—as well as the control structure—allows for other arbitrary object models, such as aspects as applied in [15]. Such a model will be simple for the object in figure 6 (left) but will be complex in the case of objects such as the punch in figure 12 (right). Hypotheses for object locations in the scene are generated based on color using the algorithm outlined in section 3.2. Results are shown below in figure 14 for the object from figure 6. An evaluation of six color-normalization algorithm with respect to object localization depends on the particular object. In most cases, UV histograms performed better than RGB [34]. Object hypotheses computed from histogram intersection or other histogram comparisons such as the EMD distance [40] gave approximately the same results but higher computation times.

The pan-tilt unit mounted on the linear sledge is moved to estimate 3-D information using the ideas outlined in section 3.5; we compute a set of scattered 3-D points. The focal length is known from calibration (section 3.1). A result is shown in figure 13; neither the accuracy nor the number of points is sufficient to estimate 3-D surfaces of the objects.

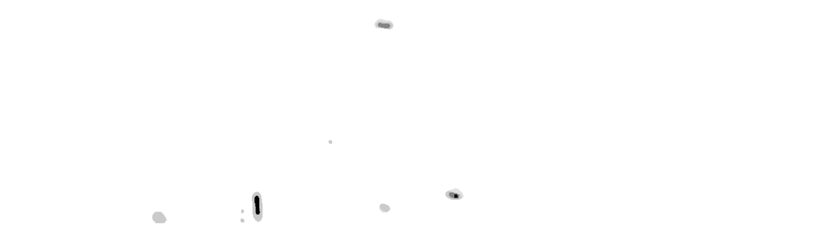


Figure 14. Backprojection of red object

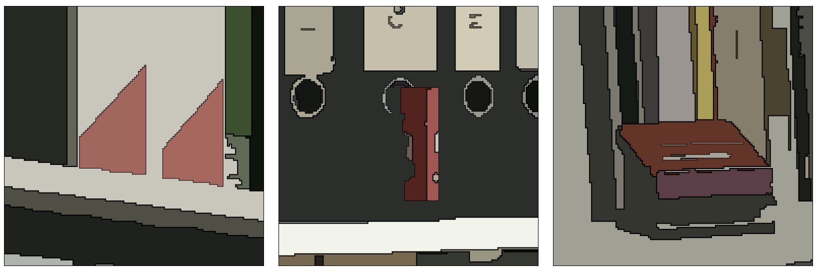


Figure 15. Segmentation of object hypotheses; shown in figure 12 (right).

The subsequent goal now is to fovealize each object hypothesis and to generate closeup views. This is required because in the overview image, no stable features based on segmented regions can be detected for the object's size.[3] Figure 12[4] shows an overview of the scene captured with minimal focal length. Figure 12 shows three hypotheses in the closeup 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. Considering the calibrated relationship between zoom position and enlargement factor (section 3.1), 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 closeup view of the object. In [57, p. 45], three methods are listed to do fovealization technically. The method above using a pan-tile device on a linear sledge adds a fourth method to this list. This operation is an example of the feedback arrow in figure 1, the camera move is determined by the results of image analysis.

The segmentation of color regions on the detail view of the scene now uses the color segmentation mentioned in section 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 15.

The goal of the module for object verification is to find an optimal match of segmented regions in the color closeup views and the gray ovals in figure 8. The search is directed by the certainty of the concept `officeScene` which is the goal concept for the subgraph of gray ovals

3. An MPEG video will show camera motion, 3-D points rotated in space, fovealization, and so forth in the online version.

4. The 3-D information is more visible in the video (c.f. footnote 3) when the point cloud is rotated.

in figure 8. For the computation, we use the attributes, relations, and their certainties in the parts and concrete concepts. In the concept colorRegion, we use the Euclidean distance of the mean color vector to a prototype of the color red, which is previously determined by the mean of a histogram. 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, and no pose estimation is performed. An extension is envisaged in combination with the project in section 3.3. The actual scale is determined by the 3-d information computed in section 3.3. The actual scale is determined by the 3-D information computed in section 3.5.

For the search during matching of the regions, we use an A*-based control [31] for the graph search, computing an instantiation path through the network. The concepts are then instantiated from the bottom up in the order determined by the path. This means that no restrictions are propagated during the analysis. The certainty of the nodes in the search tree is equal to the certainty of the goal concept; in order to guarantee an optimistic estimate, all certainties of noninstantiated parts are set to one as long as the goal concept is not reached. The computation of the certainties is deferred to computations of certainties for instances that in turn use their attributes. The similarity measure for color regions influencing the certainty is part 6 of the task-specific knowledge and can thus be exchanged easily.

In seventeen experiments for evaluation of this application, we used the punch, gluestick, 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 certainty function that weights regions of the object's color higher than regions of other colors. The rate of correct verifications of the hypotheses is approximately 70%. This figure includes the frequent confusions of stapler with the large tape roller. If we leave this object out, the recognition rate is approximately 80%. The total time for processing is around 2 min. for one scene on an SGI)2 (R10000, 195 MHz). Much of this time is spent for the color segmentation, backprojection, and waiting for the camera to reach a stable position. The convolution in figure 5 was replaced by a 21×21 median filter to obtain good local maxima. The A* search was replaced by our second control strategy in [2]; using this integrative-optimizing control, the same recognition rates could be reached.

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 DIROKOL project,[5] a service root will provide employees in hospitals with "fetch and carry services." The system is equipped with a couple of classical robotic sensors (such as sonar and infrared, a four-finger hand [5] mounted on a robot arm, and a TRC stereo system for visual navigation, object recognition, and active knowledge-based scene exploration. Similar to the application described in section 4.1, semantic networks are

5. The project is funded by the Bavarian Research Foundation (bayerische Forschungsstiftung).

used for object representation. 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 an explicit knowledge base that intuitively corresponds to the description given by humans and an adjustable, trainable part, especially for the active components in an active vision system. In this section, we present one aspect for the vision submodule of an autonomous system; this issue's article by Schlegel et al. provides a broader discussion of vision and robotics including navigation and robot control.

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 [14]. The landmark definition is based on color histograms (section 3.2) which have been extended technically by deriving a new extended class from the class for histograms to contain additional information on the distribution of the pixels position of a certain color bin as in [23]. Such histogram bins are defined as landmarks that contain a certain amount of entries; in addition, the distribution of the $(x, y)^{\mathrm{T}}$ position in the image plane indicates a local color region. this means that the variance in $(x, y)^{\mathrm{T}}$ positions must be small. The mean $(m_x, m_y)^{\mathrm{T}}$ of the $(x, y)^{\mathrm{T}}$ positions indicates the position of the landmark in the image.

For automatic self-localization based on natural landmarks defined by color histograms, a probabilistic approach has been used. Assume now, that a landmark is given (the histogram bin, the covariance matrix, (C, and the mean position $(m_x, m_y)^{\mathrm{T}}$ of the histogram entries in the image). The landmarks can be detected in the image by randomly sampling positions from that normal distribution given by the mean $(m_x, m_y)^{\mathrm{T}}$ and the covariance matrix C. For each sampled position, the histograph bin corresponding to the color pixel at that position is compared with the prerecorded histogram bin. The positive hits are counted, and the probability for seeing the landmarks is computed by the relative frequency. Assuming independence between the landmarks, the probability for looking at a certain scene position can be computed by multiplying the probability for each single landmark. The unknown position is then found by a maximum-likelihood estimation over all possible positions.

For our experiments, we moved the camera around our department's floor. We took a sequence of 300 color images at a frame rate of 1 Hz. The experiments have shown that, on average, 4.2 (variance: 11.2) landmarks per image were automatically defined. An example image with the detected landmarks is given on the left of figure 16. The computation time for landmarks definition was 230 ms. With these landmarks, self-localization by stochastic sampling have been evaluated. One hundred samples were drawn, which took 14 ms. The results of these experiments are visualized on the right side of figure 16 by using a confusion matrix of size 300×300. On the axis of the matrix, different areas of the department's floor are represented by different colors. Each element of the matrix represents the quality of the correspondence between the original image (row) and the test image (column). A dark entry stands for a good correspondence and a light entry for a bad one. As expected, the diagonal of the matrix has many dark entries, indicating a good self-localization ability. The blockwise structure of the rest of the dark entries

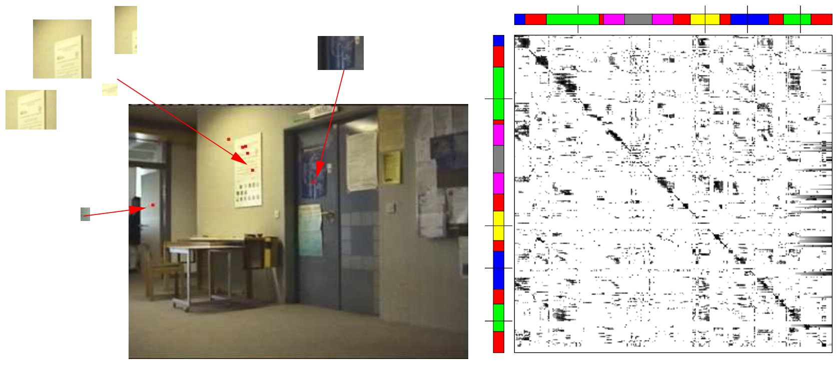


Figure 16. Left: Example image with automatically detected landmarks. Right: Confusion matrix showing the quality of self-localization. Dark entries indicate high correspondence between two positions.

indicates that similar appearing areas of the department's floor are also recognized.

It is worth noting that the achieved results are based only on color histograms without and 3-D 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 current work.

5 Active Search for Objects with Mobile Robot

A mixture of the two applications has been realized as follows: a class was defined on top of the robot control, allowing the robot to be placed on a straight line at a discrete position. They syntax for moving the robot was forced to be identical to that of positioning the linear sledge, as virtual operators defined in a common base class (*SMotor* in figure 3) were used in C++. The same common interface syntactically linked the TRC head on the robot and the Canon camera in the office. Only three lines of the application program for the office exploration had to be changed for the experiments described in section 4.1 on the robot.

6 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 camera 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 and gave examples for the expressional power of the proposed approach, especially when applied to unusual areas such as device interfaces. 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 that can be combined with the systems described in section 4. To give an example, one of the color-normalization algorithms described in [9], will be selected for each object in the application in section 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 section 3.7 after they have been localized. An integration of statistical models (section 3.3) and semantic networks (section 4.1 [25, 39]) will be used for holistic object recognition as in [29]. These models are invariant of the aspect. This extension will be used in the system in section 4.1 as well as in section 4.2. Pose estimation in the verification stage will be integrated into the matching process.

Acknowledgement

This work was funded partially by the *deutsch Forschungsgemeinschaft* (DFG) under grants SFB 182 and SFB 603. Only the authors are responsible for the contents. This work was funded partially by the Bayerische forschungsstiftung (project DIROKOL).

Special thanks to colleagues in Bielefeld for discussions on the use of semantic networks for computer vision.

References

[1] More references to our own work can be found in the publication section of our Web site http://www5.informatik.uni-erlangen.de.

[2] Ahlrichs, U., Fischer, J., Paulus, D., and Niemann, H. Approach to active knowledge based scene exploration. In *ES99*-19th *SGES International Conference on Knowledge Based Systems and Applied Artificial Intellignece*.Pages 289–301. Cambridge 1999.

[3] Arps, R. B., and Prat, K. K., (Eds.). In *IN SPIE Image Processing and Interchange: Implementation and Systems*, vol. 1659, 1992.

[4] Beß, R., Paulus, D., and Harbeck, M. Segmentation of lines and arcs and its application to depth recovery. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 4: 3161–3165 Computer Society Press, Munich, April 1997.

[5] Butterfass, J., Hirzinger, G., Knoch, S., and Liu, H. Dlr's multisensory articulated hand. In *IEEE International Conference on Robotics and Automation*, pges 2081–2086. Leuven, belgium, 1998.

[6] Cardelli, L. and Wegner, P. On understanding types, data abstraction, and polymorphism. *Computer Surveys*, 17(4):471–522, 1985.

[7] Carlsen, I. C. and Haaks, D. IKS^{PHF}—concept and implementation of an object-oriented framework for image processing. *Computers and Graphics*, 15(4):473–482, 1991.

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

[9] Csink, L., Paulus, D., Ahlrichs, U., and Heigl, B. Color normalizaiton and object localization. In Rehrmann [42], pages 49–55.

[10] Denzler J. *Aktives Sehen zur Echtzeitobjektverfolgung*. Infix, Aachen, 1997.

[11] Denzler, J., Heigl, B., and Niemann H. An efficient combination of 2d and 3d shape description for contour based tracking of moving

objects. In H. Burkhardt and B. Neumann (Eds.), *Computer Vision-ECCV 98*, Lecture Notes in Computer Science, pages 843–857. Berlin, 1998.

[12] Denzler, J., and Niemann, H. Real-time pedestrian tracking in natural scenes. In G. sommer, K. Danmiliidis, and J. Pauli (Eds.), *Computer Analysis of Images and Patterns, CAIP'97, Kiel 1997*, Lecture Notes in Computer Science pages 42–49. Berlin, 1997

[13] Denzler, J. and Niemann, H. Active rays: Polar-transformed active contours for reali-time contour tracking. *Journal on Real-Time Imageing*, 5(3):203–213, June 1999.

[14] Denzler, J. and Zobel, M. Automatische farbbasierte Extraktion natürlicher Landmarken und 3D-Positionsbestimmung auf Basis visueller Information in indoor Umgebungen. In Rehrmann [42], pages 57–62.

[15] Dickinson, S. J., Christenesn, H. I., Tsotsos, J. K., and Olofsson, G. Active object recognition integrating attention and viewpoint control. *Computer Vision and Image Understanding*, 67(3):239–260, September 1997.

[16] Draper, B., Hanson, A., and Riseman, E. Knowledge-directed vision: Control, learning and integration. *Proceedings of the IEEE*, 84(11):1625–1637, November 1996.

[17] Faasch, H. Konzeption und Implementation einer objektorientierten Experimentierumgebung für die Bildfolgenauswertung in ADA. PhD thesis, Universit="at Hamburg, Hamburg, 1987.

[18] Fischer, V. and Niemann, H. A parallel any-time control algorithm for image understanding. In *Proceedings of the* 13th *International Conference on Pattern Recognition (ICPR)*, pages A:141–145. IEEE Computer Society Press, 1996.

[19] Garvey, T. Perceptual stragegies for purposive vision. Technical report, SRI AI Center, SRI International, Menlo Park, 1976.

[20] Gorlen, K. E., Orlow, S., and Plexico, P. S. *Data Abstraction and Object-Oriented Programming* in C++. John Wiley and Sons, Chichester, 1990.

[21] Haralick, R. M. and Ramesh, V. Image understanding environment. In Arps and Pratt [3], pages 159–167.

[22] Heigl, B., Paulus, D., and Niemann, H. Tracking points in sequences of color images. In *Proceedings* 5th *German-Russian Workshop on Pattern Analysis*, 1998.

[23] Heisele, B., Ritter, W., and Kreßel, U. Objektdetektion mit Hilfe des Farbfleckenflusses. In V. Rehrmann (Ed.) *Erster Workshop Farbbildverarbeitung*, volume 15 of *Fachbverichte Informatik*, pages 30–35. Universität Koblenz-Landau, 1995.

[24] Hoogs, A. and Hackedtt, D. Model-supported exploitation as a framework for image understanding. In *ARPA*, pages I:265–268, 1994.

[25] Hornegger, J. *Statistische Modellierung, Klassifikation und Lokalisation von Objekten*. Shaker Verlag, Aachen, 1996.

[26] Jähne, B. *Spatio-Temporal Image Processing*. LNCS 751. Springer, Heidelberg, 1993.

[27] Kass, M., Wittkin, A., and Terzopoulos, D. Snakes: Active contour models. *International Journal of Computer Vision*, 2(3):321–331, 1998.

[28] Köthe, U. Reusable components in computer vision. In B. Jähne, H. Haussecker, and P. Geissler (Eds.), *Handbook of computer Vision and Applications*, pages 103–132. Academic Press, London, 1999.

[29] Kummert, F., Fink, G., and Sagerer, G. Schritthaltende hybride Objektdetektion. In E. Paulus and F. Wahl, (Eds.), *Mustererkennung 1997*, pages 137–144. Springer, Berlin, September 1997.

[30] Liedtke, C.-E., Grau, O., and Growe, S. Use of explicit knowledge for the reconstruction of 3-D object geometry. In V. Hlavac and R. Sars (Eds.), *Computer analysis of images and patterns—CAIP '95*, number 970 in Lecture Notes in Computer Science, Springer, Heidelberg, 1995.

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

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

[33] Niemann, H., Sagerer, G., Schröder, and Kummert, F. Ernest: A semantic network system for pattern understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 9:883–905, 1990.

[34] Paulus, D. and Csink, L. On color normalization. In T. Szirányi and J. Berke (Eds.), *Magyar Képfeldolgozók és Alakfelismerök Országos Konferenciája, Konferenciakiadvány*, pages 222–229. Keszthely, 1997.

[35] Paulus, D. and Hornegger, J. *Pattern Recognition of Images and Speech in C++* Advanced Studies in Computer Science. Vieweg, Braunschweig, 1997. (Now available as [36].)

[36] Paulus, D. and Hornegger, J. *Applied pattern recognition: A practical introduction to image and speech processing in C++*. Advanced Studies in Computer Science Vieweg, Braunschweig, 2nd edition, 1998.

[37] Paulus, D., Hornegger, J., and Niemann, H. Software engineering for image processing and analysis. In B. Jäyne, P. Geißler, P., and Haußecker, H., (Eds.), *Handbook of Computer Vision and Applications*, 3, pages 77–103. Academic Press, San Diego, 1999.

[38] Paulus D. and Niemann, H. Iconic-symbolic interfaces. In Arps and Pratt [3], pages 204–214.

[39] Pösl, J., Heigl, B, and Niemann, H. Color and depth in appearance based statistical object localization. In H. NIemann, H.-P Seidel, anhd B. Girod, (Eds.), *Image and Multidimensional digital Signal Processing '98,* pagess 71–74. alpbvach, austria, July 1998. Infix.

[40] Buhmann, J. Puzicha, Rubner, Y., and Tomasi, C. Empirical evaluation of dissimilarity measures for color and texture. In H. burkhard and B. Neumann, (Eds.), *Computer Vision—ECCV]98,* number 1406 in Lecture Notes in Computer Science, pages 563–577. Springer, Heidelberg, 1998.

[41] Rasure, J. R. and Young, M. Open environment for image processing and software development. In Arps and Pratt [3], pages 300–310.

[42] Rehrmann, V. (Ed.). *Vierter Workshop Farbbildverarbeitung*, Koblenz, 1998. föhringer.

[43] Rimey, R. and Brown, C. Task-orinted vision with multiple Bayes nets. In A. Blake and A. Yuille (Eds.), *Active Vision*, pages 217–236. Cambridge, MA, 1992.

[44] Rimey, R. D. and Brown, M. Controlling eye movements with hidden Markov models. *International Journal of Computer Vision*, 7(1):47–65, January 1991.

[45] Ronfard, R. Region-based strategies for active contour models. *International Journal of Computer Vision*, 13(2):229–251, 1994.

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

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

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

[49] Shi, J. and Tomasi, C. Good features to track. In *Proceedings of Computer Vision and Pattern Recognition*, pages 593–600. Seattle, Washington, June 1994. IEEE Computer Society Press.

[50] Stroustrup, B. *The C++ Programming Language*, 3rd *edition*. Addison-Wesley, Reading, 1997.

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

[52] Thomanek, F. and Dickmanns, E. D. Autonomous road vehicle guidance in normal traffic. In *Second Asian Conference on Computer Vision*, pages III/11-III/15. Singapore, 1995.

[53] Ullman, S. *High-level Vision: Object Recognition and Visual Cognition*. MIT Press, Cambridge, MA, 1996.

[54] Vinod, V. V., Murase, H., and Hashizume, C. Focussed color intersection wit6h efficient searching for object extraction. In *Pattern Recognition*, 30 pages 1787–1797, 1997.

[55] Willson, R. G. *Modeling and Calibration of Automated Zoom Lenses*. PhD thesis, Carnegie Mellon University, Pittsburgh, 1994.

[56] Wixson, L. Gaze selection for visual search. Technical report, Department of Computer Science, College of Arts and Science, University of Rochester, Rochester, New York, 1994.

[57] Yeshurun, Y. Attentional mechanisms in computer vision. In V. Cantoni (Ed.), *Artificial Vision, Human and Machine Perception*, pages 43–52. Plenum Press, New York, 1997.