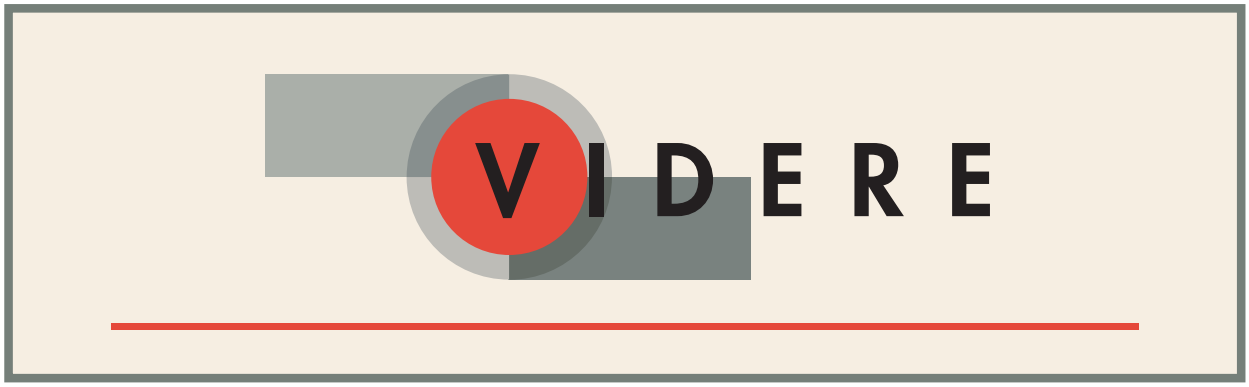


Videre: Journal of Computer Vision Research

Quarterly Journal
Winter 2000, Volume 1, Number 4
The MIT Press

Article 2

Integrating Vision Based Behaviors with an Autonomous Robot

Christian Schlegel
Jörg Illmann
Heiko Jaberg
Matthias Schuster
Robert Wörz

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.

Integrating Vision-Based Behaviors with an Autonomous Robot

Christian Schlegel[1], Jörg Illmann[1], Heiko Jaberg[1], Matthias Schuster[1], and Robert Wörz[1]

Although many different vision algorithms and systems have been developed, the integration into a complex intelligent control architecture of a mobile robot remains in most cases an open problem. In this paper, we describe the integration of different vision-based behaviors into our architecture for sensorimotor systems. This raises new questions and requires the consideration of significant constraints that are often not in the main focus of vision but nonetheless play a major role for the overall success. By means of different scenarios like person tracking, searching for different objects, and achieving different object configurations within stock areas, the structure of the vision system and the interaction with the overall architecture is explained. The interaction of vision-based modules with the task-level control and the symbolic world model is an especially important topic. The architecture is successfully used on different mobile robots in natural indoor environments.

Keywords: robotics, autonomous robots, system architecture, vision integration, object recognition, person following.

1. Research Institute for Applied Knowledge Processing (FAW), Helmholtzstraße 16, D-89081 Ulm, Germany, <last-name>@faw.uni-ulm.de

1 Introduction

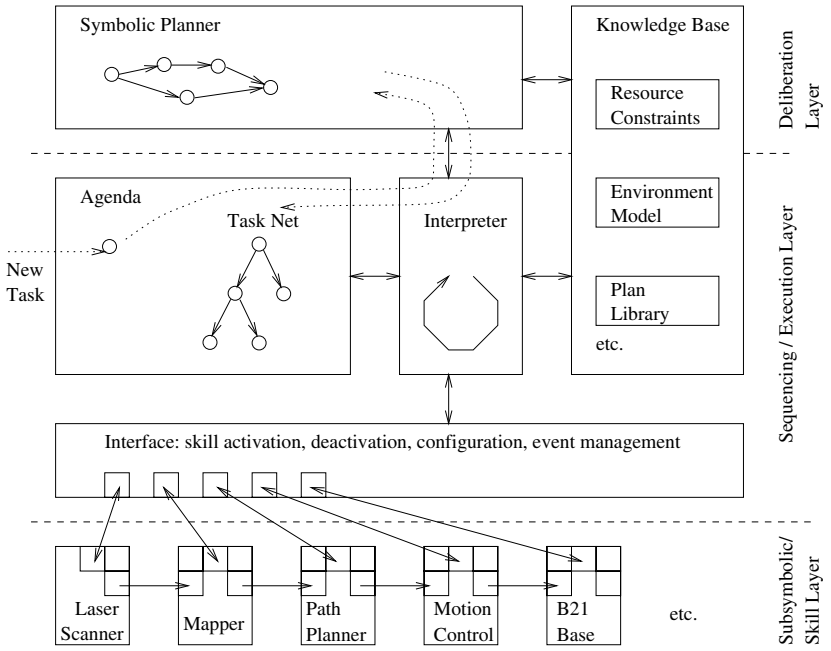
Integration of vision modules into a control architecture for an autonomous mobile robot is more difficult than just adding the vision components. This is due to the high bandwidth of vision sensors, which require a task-specific configuration of vision-based behaviors. Another important aspect is the lack of processing power on a mobile robot, which often prevents the use of existing powerful algorithms or requires them to be adjusted to the specific needs of a mobile platform. Therefore, integrating vision-based behaviors into an intelligent architecture for mobile robots is not only adding different components, but it affects many parts of the architecture itself. Until integration of vision algorithms becomes a topic, many often-neglected aspects turn out to be significant challenges. This in many cases requires the first approach to be modified with respect to changed priorities.

In this paper, we describe aspects of the integration of vision modules into a robot control architecture. The focus is not only on architectural aspects and problems arising from the interaction of vision modules with the overall system, but also on details of the final algorithms used to meet the requirements of a mobile platform and to achieve the required robustness. It turned out that, in many cases, the task-dependent configuration of vision mechanisms including an appropriate setting of parameters is a crucial point for achieving robust vision-based behaviors. In this point the vision mechanisms benefit from the overall system architecture. The architecture supports a declarative representation and a task-dependent configuration of modules taking into account resource constraints and conflicts among competing settings. Thus, even complicated relationships between parameters can be conveniently represented. A single vision approach has to cope with all kinds of vision-related tasks, but we can improve the robustness of the vision-based tasks by selecting adjusted settings and algorithms.

The mobile robot described in this paper is used as a demonstrator within the SFB 527. The SFB (collaborative research center) is a basic research project on the integration of symbolic and subsymbolic information processing in adaptive sensorimotor systems. In particular, vision plays a central role, because camera systems' high bandwidth of data contain a rich variety of relevant information for the tasks of a mobile system. It is, therefore, an important source of information.

The paper is organized as follows. First, we describe the main parts of the currently used architecture and the integration of vision-based behaviors into the overall system. Then we describe in more detail two main vision-based behaviors that are particularly important for the tasks

Figure 1. Architecture overview.



the system can perform. The first one is used to detect and locate different objects within a natural indoor environment (to be used, for example, within a cleanup procedure, putting different objects into wastebaskets). The second behavior is to track and follow a person. This behavior is useful for service robotic applications in which a person is guided by a mobile robot or in which a person shows the robot where to execute a specific task. Finally, a scenario is presented in which the robot has to achieve different object configurations in several stock areas. This requires the close interaction of different modules with the vision system.

1.1 System Overview

The basic idea of the system architecture is described in [16]. At the implementation level, this architecture currently comprises three layers as shown in figure 1. The subsymbolic level, where real-time capabilities are an important topic, consists of continuously working modules. Many skills of the robot are implemented at this level including, for example, algorithms for motion control and map building. Those modules work on raw sensor data and generate commands for the actuators. In some sense, they contain no state and just build a reactive system that has to be configured appropriately. The second layer is the execution layer, which is responsible for appropriate configuration and synchronization of the activities of the subsymbolic level. It supervises tasks performed on the subsymbolic level. In particular, it selects appropriate parameters for skills taking into account the current execution context. It works on only discrete states that are used to synchronize real-world execution with the symbolic task description. Because the outcome of a selected behavior could be different from the expected one, a simple linear sequence of primitive behaviors is not sufficient. The execution layer must conditionally select appropriate behaviors for the current situation. The third layer contains time-consuming algorithms. Normally, different transitions occur at the execution layer while the deliberation layer tries to generate a solution. One of the modules at this layer contains the symbolic planner. The generated plans describe the desired plot only at

a very abstract level and constrain the selection of execution possibilities at the execution layer. The subsymbolic level is often called *the skill layer* and the intermediate level *the sequencing layer.*

1.2 Related Work

Because in this paper both the architecture of the mobile robot and the vision component is part of the reported work, we describe related work in both areas. Due to space restrictions, we concentrate on person-following approaches in the field of vision because the algorithms used in the other vision modules are mainly off-the-shelf and differ particularly with respect to the integration into the system architecture.

1.2.1 Architecture

The kind of architecture we use is very similar to 3T [2]. This is not surprising because this kind of architecture seems to be extremely well suited to implement an intelligent control architecture for a mobile robot. The main strength of this kind of architecture is that, by dividing the whole system into three layers, each layer is significantly easier to implement than the whole system. Another important topic is that the sequencing layer allows abstraction from details that would complicate the planning layer. This is the key to bridge the gap between the skill layer and the deliberative layer. With respect to vision, a three-layer architecture provides the glue component to bridge the gap between low-level building blocks (like filter operators) and the representation of required parameters for task-dependent configurations. Implementations of multilayer architectures are different with respect to the algorithms used at each layer. The sequencing layer often consists of a variant of the RAP system [7]. Recently, new programming languages like ESL [8] have been proposed, but they are not especially designed to be used with a symbolic planner. A very good overview on multilayer architectures is given in [14]. An unsolved topic, however, is the kind of interaction between different layers. Therefore, many existing layered architectures are different with respect to this point. In particular, the interaction of vision-based behaviors like object recognition with the geometric and symbolic world model of the robot is still challenging.

1.2.2 Person following

Much work in the field of person detection and tracking is based on vision systems using stationary cameras [21]. Many of the approaches assume a stationary background because moving objects are detected by subtracting two frames [4, 17]. Other methods require that the tracked person is never occluded by other objects [1]. Therefore, they are typically not suitable for person following on a mobile robot because, at the same time the robot tracks the person, it has to move into the person's direction to stay close enough.

Many approaches can be classified regarding the kind of models that are used. Models range from complex 3-D models [9] to simpler 2-D models [17, 21]. The basic idea is to use deformable models to represent the boundary of the segmented body. Even those approaches assume one or more static cameras.

Several approaches use color-based image segmentation for object-ground separation [1, 3]. Persons to be tracked are represented by a collection of color blobs. A color-based system, able to track colored blobs in real time, is presented in [22]. It is implemented on a mobile

robot that locates a person in dynamic environments but requires the individual to wear a shirt of one of four predefined colors. Drawbacks are that the approach is not adaptive to illumination changes and, because it relies solely on the color and does not consider any shape, it requires that the person can be uniquely identified by color. Therefore, the color should not occur in the background.

A model-based approach for object tracking using two-dimensional geometrical models can be found in [11]. A binary contour model is matched against the current edge image using the Hausdorff distance, and a successive match leads to an update of the contour model. This approach adapts to continuous deformations of the person's image during movement, thus overcoming the difficulties that person tracking imposes on rigid-model approaches. The main drawback of this approach is its computational complexity. Without any restrictions in the search space, it seems to be unsuitable for real-time application.

2 Details of the Architecture

Although the basic architecture is based on a well-known three-layer model, there are significant differences that mainly concern the interaction between different layers. These differences are described in more detail in this section.

The basic entity of the robot software is a module, which is equivalent to a process and consists of different threads. The external interface of a module is built using predefined communication templates. The software framework SMARTSOFT [20] also defines the basic internal structure of each module, which allows any communication details to be completely hidden from the module implementation and ensures standardized module interfaces. This includes a configuration interface for each module (used, for example, by the execution layer to coordinate module activations). The modules form a client-server architecture and can be distributed transparently over different computers.

The execution layer is very similar to the RAP-system [7]. It has knowledge about default execution of different tasks. By appropriately configuring the modules at the subsymbolic level, different behaviors can be realized using the available modules. Different behaviors can be built based on a limited set of skills. The configuration of a behavior not only consists of activating and deactivating skills. Moreover, because the whole system is based on a client-server architecture, module interconnections can be changed online. We do not use a precompiled wiring of data flow between modules. This is particularly relevant for the vision modules for which a basic set of operators has to be connected in the appropriate way to achieve the currently required image-processing pipeline.

The execution layer ensures that bounded resources are allocated appropriately and constraints of behaviors are met. Then the subsymbolic layer has to operate within the boundaries of the configuration. Once enabled, such a configuration is active as long as no ending condition becomes true. It is assumed that each skill is able to detect any malfunction or condition where its usage is not appropriate. This is also known as the concept of cognizant failures [15].

Events are used to report any relevant change of state, which also includes the successful completion of a task. Because verifying the event condition often requires skill-specific data, events are not standalone or equivalent to skills but part of a skill. This allows the inclusion of

extended status information about the situation that fired the event. If events are separated from skills, many private details of modules have to be exported to events. Events are also relevant for the vision modules, where, once configured, image-processing pipelines can run autonomously until a specified condition becomes true (which is then reported to the sequencing layer).

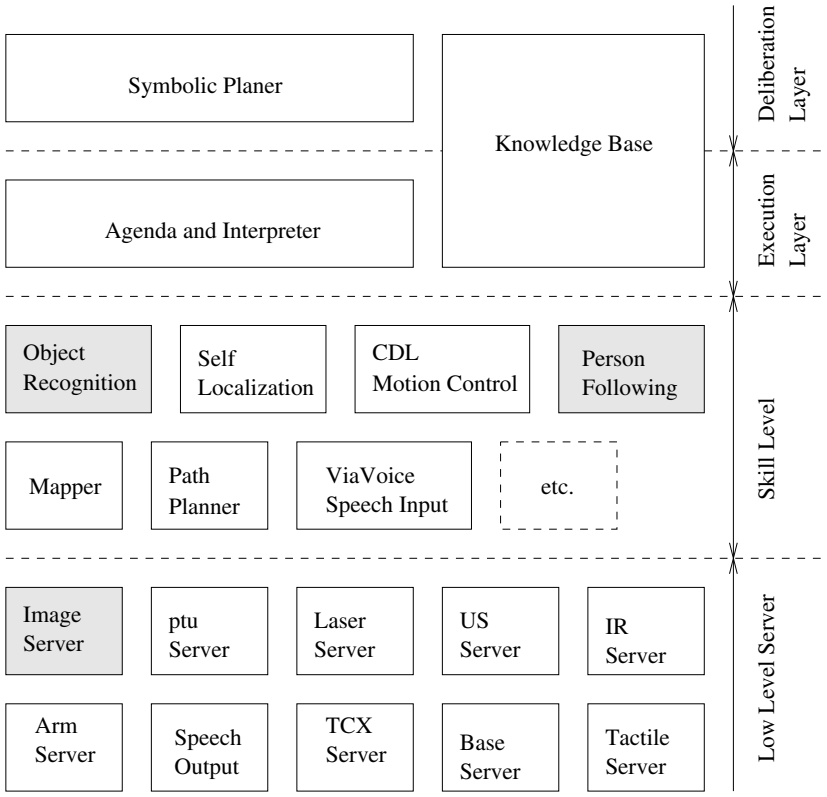
Another important detail concerns transient conditions, which can occur when switching between configurations and have to be avoided. For example, having activated the path planner together with the event `no path available` before the mapper can provide the requested part of the map would fire the event unmeant. Therefore, the skill interface has been redesigned to provide more options to specify the order of skill activation and deactivation than the original RAP system. For example, each module has an uniform finite-state automation to ensure that parameter settings are done only when not affecting any calculations.

The currently used syntax to represent task networks is very close to the RAP notation. The main differences are the kind of interface to the subsymbolic level and the structure of the knowledge base. The knowledge base is organized as a frame system with single inheritance of object features. This allows a more structured representation (which is important as soon as the knowledge base grows). The knowledge base is separated from the agenda interpreter and is accessed by a very simple tell-and-ask interface. Therefore, different kinds of knowledge representations can be used without reimplementing the whole execution layer. The agenda interpreter has been extended to handle some more ordering constraints in task nets like *try-in-order* or *parallel-or*. These modifications allow a more convenient formulation of task nets. A consistent representation of different configurations and activation sequences is achieved by using unified module structures.

As interface to the symbolic planner, an abstract STRIPS-like [6] description of the tasks available at the execution layer is used. With the current implementation, the planner is called from the execution layer for specific, predefined problems. Whenever the planner operator occurs within a task net, the agenda interpreter calls the symbolic planner together with the appropriate part of the knowledge base. Because we know for which problems we call the symbolic planner, we can also define which part of the knowledge base has to be transformed into the planner representation for each problem instance. The generated plan defines the expansion of the planner operator and substitutes it in the task net. A typical subproblem solved by the planner is generating a sequence to get a specific box out of a stack of boxes. The currently used planner is IPP [12, 13]. Of course, this kind of planner interface is one of the simplest, but it has specific advantages as well. By selecting only problem-relevant parts of the knowledge base and hiding from the symbolic planner as many details as possible, the planner is able to produce plans within a reasonable time.

Some of the currently implemented modules at different layers are shown in figure 2. Many modules at the subsymbolic layer deal with motion control and map building. Another important module is self-localization, which ensures that the position error of the robot stays within certain boundaries [10]. Details of the motion-control skills can be found in [18]. The vision system consists of two different modules: The first module is able to search and locate different objects, whereas the second module is able to track a person. The vision modules are highly configurable by the sequencing layer and are used in combination

Figure 2. Modules at different layers.



with other modules of the skill layer to build complete behaviors. With respect to vision, the symbolic description provided by the sequencing layer of all the different constraints and parameter settings significantly eases the consistent administration of the various task-dependent configurations. In particular, the declarative representation of configurations is very useful to easily reuse basic building blocks for different tasks. Vision algorithms especially often require a very accurate situation-dependent adjustment of parameters which can be handled conveniently within the knowledge base. The execution layer, the knowledge base, and the symbolic planner are each implemented in a separate module.

3 Model-Based Object Recognition

The vision component is used to detect and locate different objects in a natural indoor environment. One task includes cleaning up an area, which means that, while moving around and exploring the workspace, objects have to be recognized on the fly. The objects are regular and complex everyday objects like a trashcan, but normally with a relatively simple background. Significant difficulties arise from the moving robot and the movements of the pan-tilt unit that are necessary to fully explore the actual workspace. The movement of the robot also requires the adaptation to changing illumination conditions. The following section describes details of the object recognition-process, which is specifically adjusted to the requirements of a mobile robot.

3.1 Interface of the Vision Module

In figure 3, the structure of the vision module is shown. The vision module always has to be connected to the image server. Depending

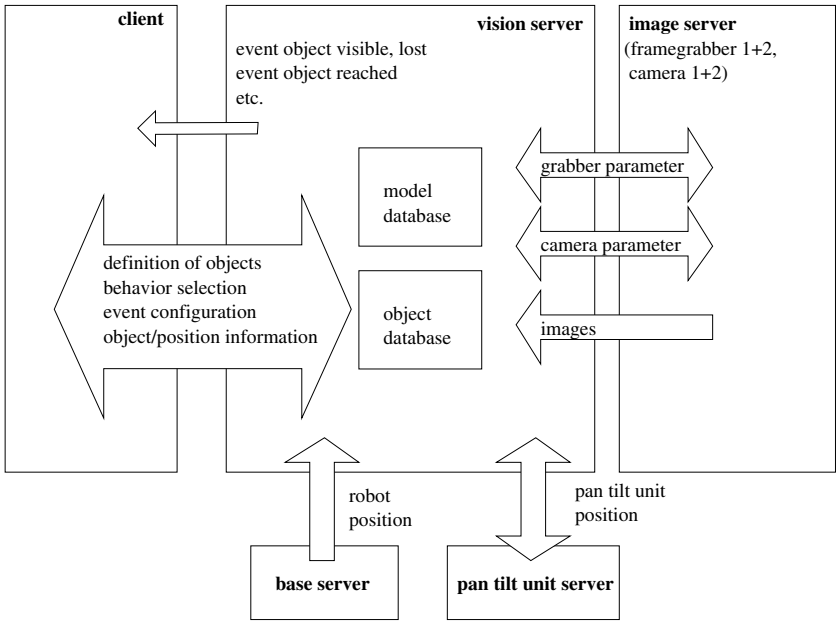


Figure 3. Structure of the vision module.

on the current behavior, it is also connected to the server of the pan-tilt unit (ptu) and the base server, which provides the current robot position. The vision server itself can provide heading information for other modules when tracking an object. The client interface allows the selection of different behaviors which either require an object list or an object identifier. The object list consists of different object types together with object attributes and restricts the behavior to objects matching the list. The more attributes specified for an object the more restricted the behavior is. If, for example, specifying *(search (type ball))*, all balls are searched. If specifying *(search (type ball)(color red))*, only red balls are searched. The object list consists of class descriptions of objects and must not be confused with object instances. An object instance is referenced by the object identifier.

Some more details of the interface of the vision module are shown in figure 4. Commands are used to set the configuration for the next behavior. Events are generated by the vision system as soon as important changes occur that have to be reported to the execution layer for task-coordination purposes. Requests are used to ask for attribute values of a specific instance referenced by its identifier. Updates are generated autonomously by the vision module depending on the selected behavior. For example, while tracking an object, the position of this object is continuously provided to subscribed clients. Normally, the motion control uses this information to move the robot into the desired direction. In figure 5, the configuration sequence for searching red balls and blue trashcans is shown in detail. After being activated, the behavior quits as soon as one matching object is found. In this case, an event is generated reporting the object identifier of the found object.

The following behaviors are implemented in the vision server and provide basic capabilities that can be used to realize more-complex vision-based tasks. The kind of behaviors are directly related to the robot's tasks and are based on the same basic building blocks. They are mainly different with respect to their internal course of processing steps and reported events.

Figure 4. The vision module interface.

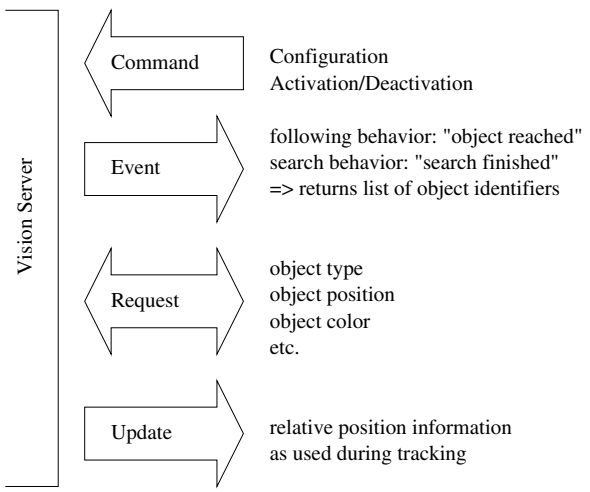
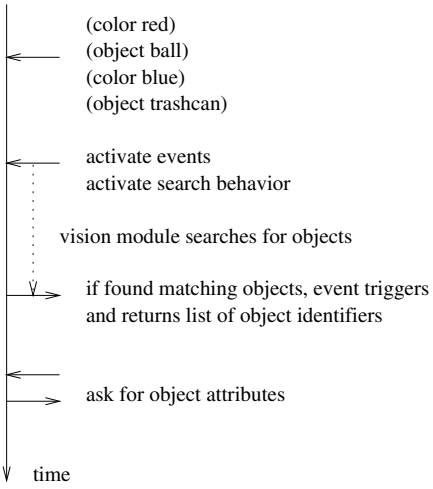


Figure 5. The search behavior.



search *objectlist*	Search the specified objects and stop searching as soon as one matching object is found. This skill can be used in parallel to exploration skills to search a complete area.
track *objectid*	Track the specified object but without providing heading information for motion control. This skill allows the continuous update of an object position, therefore keeping its identifier even when the object is moving.
follow *objectid*	Follow the specified object, and stop the robot if the object is reached or lost. If lost, search for it using the pan-tilt unit. This configuration is used together with the motion control to form an object-following behavior.
fullsearch *objectlist*	Use the pan-tilt unit to fully explore the visible area without moving the robot. This skill reports all matching objects.
reset	Reset the vision module.

3.2 Object Classification

The vision module uses a model base to perform object recognition. Each object type is specified by a set of feature values. Features used to describe objects include form descriptors (rule-based form classifiers) and color (color thresholds, color histograms). Because object recognition must be fast and robust on a mobile platform, we can use only computationally inexpensive features. However, these have to be discriminant with respect to the occuring objects. Approved features are:

size	Number of pixels belonging to the object
realsize	Real visible area of the object
boundingbox	Image coordinates of the upper-left and the lower-right corner of the color blob
eccentricity	Measures the eccentricity of the object. The range is between 0 and infinity. (The value 1 signifies a perfectly round object.)
theta	Angle between the major axis
fill	Size (bounding box)/size (blob)

It depends on the object which features are necessary for classification, and not all features are used with all objects. According to the tasks to be performed by the robot, the following objects are currently supported. Not all combinations of objects and colors can be recognized.

object	bottle, ball, trashcan, cylinder, balloon, block, door, door plate, palette, table, notype
color	red, green, blue, yellow, brown, orange, pink, lightgreen, white, black, lightorange, turquoise, lightblue, nocolor

The classification process works as follows. If an object is specified, the corresponding feature values are looked up in the model base. These values form an object-specific classifier that decides whether this object is in the image. In our scenario, only colored objects exist. The colortype `nocolor` means that the object can have any of the above colors. Therefore, if an object only is specified by type, all supported colors are checked. Colors can be defined in different ways, for example, as upper and lower bounds in RGB or in NCC color space. Classification consists of three steps (also shown in figure 6).

1. color segmentation
2. feature extraction
3. classification with object-specific classifiers based on extracted features

The classification process mainly relies on color. Color proves to be a robust feature to select regions of interest to perform the computationally more expensive steps. This aspect is particularly important on a mobile platform. In many cases, two other features besides color (such as size and eccentricity) are sufficient for the discrimination of a small set of objects. Depending on the object, nearly all features vary—also as a result of the perspective—with the distance to the robot. Therefore, the boundaries of the feature values used for classification have to be selected depending on the distance between object and camera. Because we assume that all objects rest on a flat floor (ground-plane constraint),

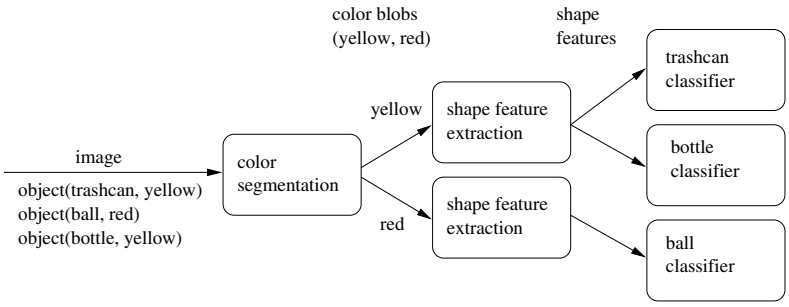


Figure 6. Procedure of object classification.

the tilt angle directly corresponds to the object distance and is used for selecting appropriate boundaries for the classification process. A feature provides good discrimination information if it differs significantly for different objects over all tilt angles.

If the command *(search (type ball)(color red), (type trashcan)(color yellow))* is given, a new image is shot. First, all red color blobs are detected by a simple color thresholding. After thresholding, the shape features for these blobs are computed. If a blob is smaller or bigger than a given threshold, it is deleted. If the shape of the blobs fulfill the criteria for balls, a ball with its attributes and coordinates is added to the object database. The same procedure then starts from the beginning for all other objects specified in the objectlist.

3.3 Object Database

The object database plays an important role in the interface of the vision modules with other parts of the architecture. The object database maintains markers for those objects that already have been seen. It is therefore used to generate "hints" regarding where to expect an object and what it should look like. This allows the generation of updates for the overall knowledge base when a new object arises or an expected object cannot be acknowledged. This in particular is a basic problem regarding the integration of object recognition with a world model regardless of the object recognition itself because the vision system always has only a partial view of the world. In the object database maintained by the vision module, each object instance has a unique object identifier. An entry in the object database contains the following information.

identifier	Unique identifier of this instance
objectclass	The object type
objectcolor	The color of the object
features	All computed features
pos3d	Cartesian 3-D world coordinates
pos2d	Polar coordinates
distance	Distance to the camera
refpoint	Location of the spatial reference point in relation to the floor

If a behavior with an object list is executed, the vision system normally generates a new unique identifier for each matching object. The only exception is when an already known object falls into an object-dependent circular region around the object's position. If the object type and the attributes besides position are compatible, the old identifier is

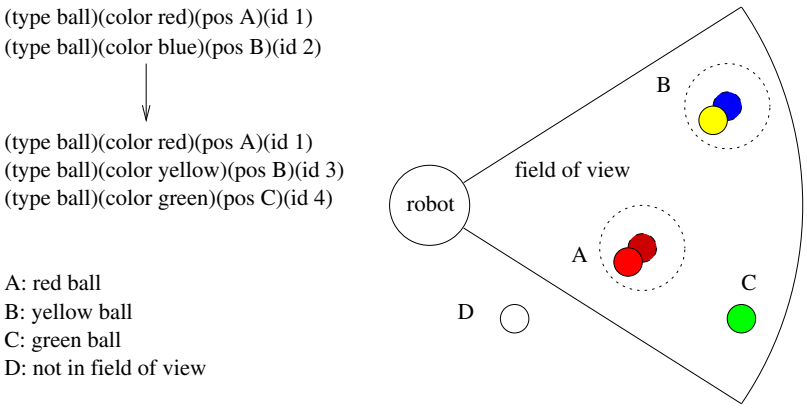


Figure 7. Update of the object database.

reused and the object's position is updated. If the objects are not compatible, the old identifier is deleted and a new entry is generated.

Figure 7 shows an example of the update process when performing a *fullsearch* for balls. The object database assumes a red ball at position A and a blue ball at position B. The current situation is a red ball at position A, a yellow ball at position B, and a green ball at position C. The object at position D is not inside the field of view and is therefore not considered. Because the object type and attributes at position A are compatible, the red ball is kept in the object database and only its position is updated. Although the object type at position B is compatible, the color attribute does not match. Therefore, the object identifier 2 is deleted, and the new identifier 3 is introduced. Because the object at position C is new, the object identifier 4 is introduced.

Each assertion or deletion in the object database is reported to the knowledge base. In the above example, one has to delete the entry with the vision object identifier 2 and assert new entries for the objects with the vision object identifiers 3 and 4. The knowledge base maintains only references to objects and symbolic descriptions of object attributes. For example, the geometric model of a bottle is private to the vision system and of no use at the symbolic level. The bottle model is totally different in the module that has to plan grasping operations. The symbolic level maintains only the links to the distributed representations.

4 Person Following

Because the person-following behavior includes a moving platform, the person-tracking approach has to cope with varying illumination conditions and a changing background. It must also be able to track different persons. Furthermore, the available processing power is limited and has to be shared with other modules. To fulfill these requirements, we use a combination of a fast color-based approach with a contour-based method. The color-based method provides regions in which the computationally expensive contour-based approach is applied. A startup phase is used to generate both the color and the contour model of the person to be tracked. During the tracking phase, these models are continuously updated.

4.1 Color-Based Person Tracking

We developed two color-based methods which can be used interchangeably. Further details can be found in [19]. Both methods are based on the NCC color space, where the intensity information is removed. The two

Figure 8. Startup following behavior.

methods are different with respect to the representation of the person-specific color model. In contrast to many other approaches that use algorithms to detect skin color, we pursue an approach relying on the color of the persons clothes. Skin-color algorithms normally do not work with black people and are not view invariant.

4.1.1 Modified NCC method

We first extended the approach presented in [22] to allow tracking of any unicolored shirt. As shown in figures 8 and 9, the person is presented to the robot, and a predefined region is used to generate the person-specific color model. The thresholds (r_l, r_h, g_l, g_h) for the red and green color band are computed as $r_{l/h} = \mu_r \mp \sigma_r$ and $g_{l/h} = \mu_g \mp \sigma_g$, respectively. We assume a normal distribution of the color values of each color band. The thresholds define a rectangular area in the NCC color space describing the color distribution to be tracked.

During the tracking phase, the current image is transformed into a binary image. A pixel is set to 1 in the binary image if the corresponding NCC color value is within the rectangular area specifying the color model. The center of gravity of the largest blob in the binary image determines the position of the person to be tracked.

4.1.2 Histogram-based method

Because the modified NCC method can track only unicolored shirts, we introduced color histograms. Based on an image representation in the NCC color space, we compute a 2-D histogram h_{2D} in which $h_{2D}(\hat{r}, \hat{g})$ specifies the number of image pixels with the color value $(\hat{r}, \hat{g})$ (figure 10). To detect and locate the person in the image, the histogram values are backprojected on the image. The pixel value $p(x, y)$ in the backprojection image with the color value $(\hat{r}, \hat{g})$ is set to the histogram value of $(\hat{r}, \hat{g})$: $p_{\hat{r},\hat{g}} = h_{2D}(\hat{r}, \hat{g})$. The resulting image specifies the frequency

Figure 9. Predefined region to extract color model.

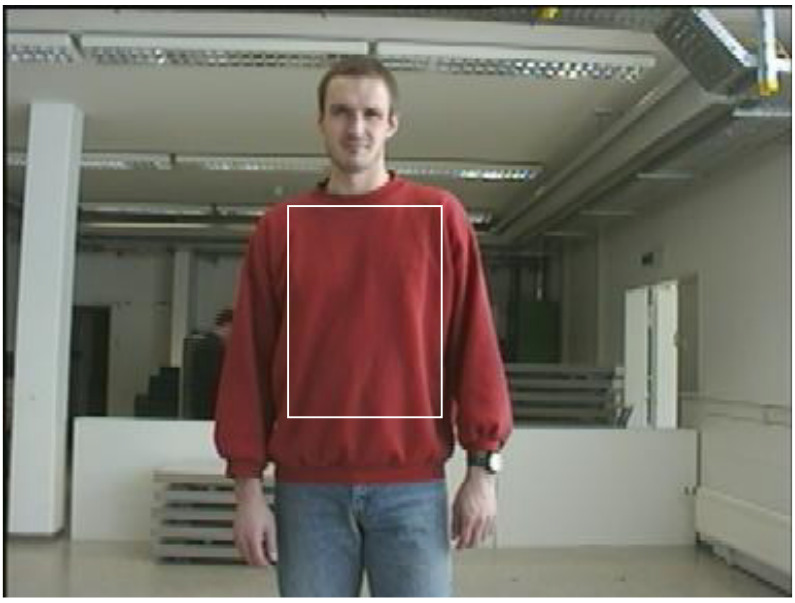
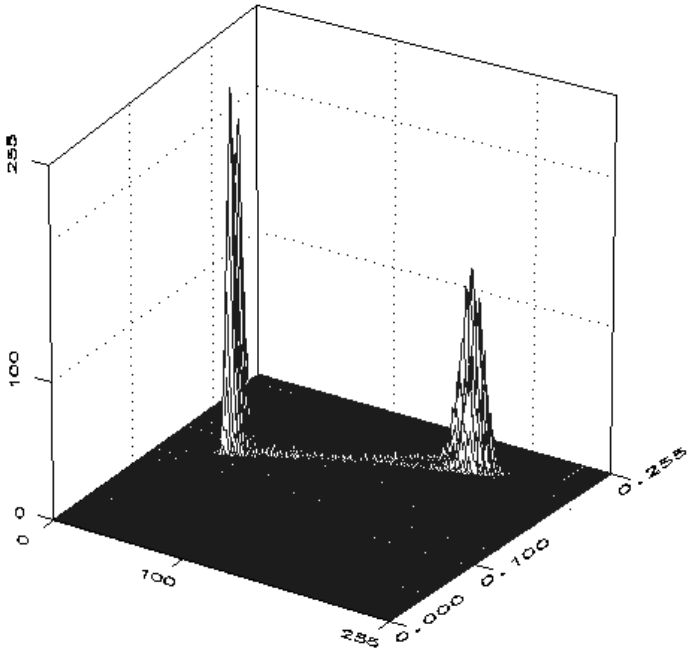
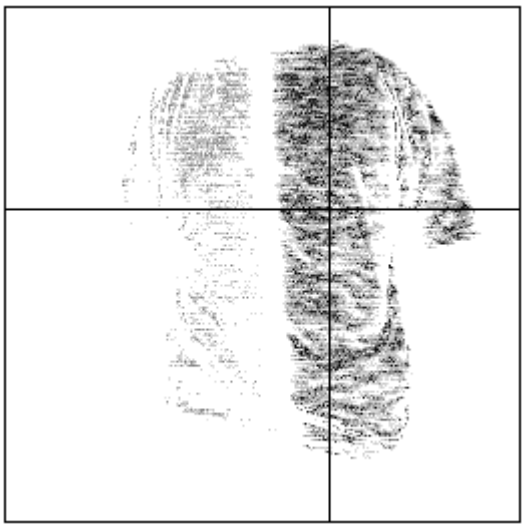


Figure 10. Normalized 2-D histogram of red and green color bands.



that the image point $p(x, y)$ belongs to the tracked person. Subsequently, pixels with low probability values are eliminated. The target position is estimated by a weighted center of gravity in the backprojection image. This is shown in figure 11 where the person is wearing a two-colored shirt. As long as the person's shirt has the same major color distribution regardless of a frontal or dorsal view, tracking is unaffected by the person presenting either view.

Figure 11. Backprojection image.



4.2 Contour-Based Approach

While the presented color-based approaches are fast and fairly robust against shape changes of the person, they are not always sufficient. If different persons wear similar-colored dresses, further information is needed. Therefore, we developed a method based on the approach in [11] that allows us to track nonrigid objects. A contour model consisting of a set of edge pixels describes the shape of a person.

4.2.1 Detection of the sought object

In the first step, the RGB image is converted to a graylevel image which is fed into a Canny operator [5] to generate a binary edge image. Let I_t be the binary edge image taken at time step t and M_t be the model at time step t represented by a binary edge image. The model may undergo certain transformations $g(M_t)$. We allow just translations in the image space. Then the sought object is detected by matching the current model $g(M_t)$ against the next image I_{t+1}. To estimate the similarity between model and edge image, we use the generalized Hausdorff-distance [11] as a distance measure.

$$h_k(g(M_t), I_{t+1}) = K^{th}_{p \in M_t} \min_{q \in I_{t+1}} \|g(p) - q\| \tag{1}$$

Minimizing h_k over all transformations, $g(.)$ provides the translation of the model M_t that leads to the best match. Let g^* be the transformation that minimizes (1) and d be the minimal distance.

$$d = min_{g \in G} h_k(g(M_t), I_{t+1}) \tag{2}$$

Descriptively this means that at least K points of the transformed model $g(M_t)$ lie at most a distance d away from any point of the image I_t.

4.2.2 Contour model generation and update

The initial contour model is generated by a startup step. All edges within a predefined rectangular area are defined to belong to the initial contour model of the presented person. To be able to handle shape changes, the model has to be updated. The new model M_{t+1} is built from the points of the image I_{t+1} whose distance to a point of the transformed model $g(M_t)$ do not exceed a threshold δ. The parameter δ controls which

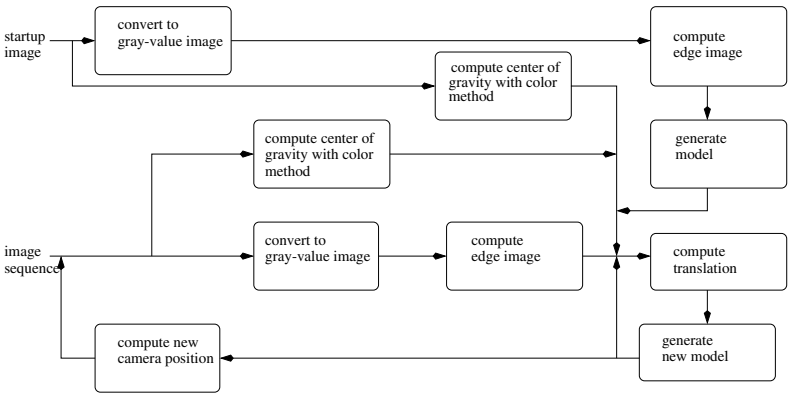


Figure 12. Combined color- and contour-based approach.

Figure 13. Mask image.

shape changes are allowed within one timestep. That also covers small rotations and size variations.

$$M_{t+1} = \{q \in I_{t+1} | \min_{p \in M_t} \|g^*(p) - q\| \leq \delta\} \quad (3)$$

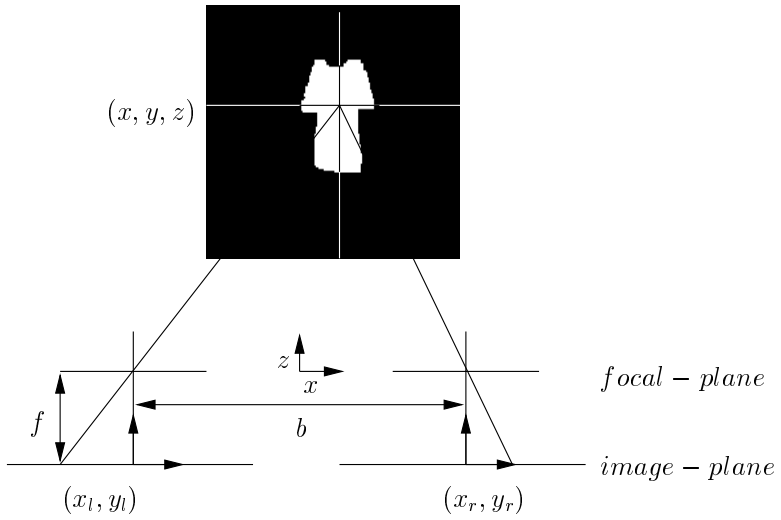
4.3 Combination of Color-Based and Contour-Based Approach

Because the contour-based approach is computationally too expensive, we combine it with the color-based one. The color-based approach is used for detecting regions where it is reasonable to apply the contour-based approach. Figure 12 depicts our concept.

To update the contour model, the result of the color segmentation is used to mask the edge image. A binary image representing the blob belonging to the detected person is first dilated (figure 13 shows the dilated blob image) and a rectangular region is added for the persons head. This considers the characteristic shape of the head and shoulders when generating the initial model (although the color distribution is generated only from the body). This binary mask is used to eliminate edge pixels belonging to the background. In figure 14, the updated contour model is shown. Masking the edge image before updating the contour model prevents the unlimited growth of the contour model. (This has been a problem with the original contour-based approach.) In addition, masking the edge image allows the model to regenerate when a partial loss of the model occurs.

Figure 14. Model update.

Figure 15. Distance calculation.



4.4 Implementation

To implement the person-following behavior on our mobile robot, control commands for the robot's actuators have to be generated. After detecting and locating the sought person within the image, we use a very simple camera model to implement a reactive following behavior. The video cameras of our robot are mounted on a pan-tilt unit (ptu) with two degrees of freedom. The ptu is used to implement a gaze holding behavior, steadily holding the person in the center of the image. To compute the control commands for the ptu, we assume a known focal length f and size of the CCD-Chip. A camera's field of view γ is computed as $\gamma_x = 2 \arctan(\frac{ccdsize(x)}{2f})$ horizontally and vertically. The ptu-angles are modified by $\alpha_x = \frac{\frac{x_B}{2} - x}{x_B} \gamma_x$ where x_B denotes the number of columns and x the estimated center of gravity of the person in the image. These directions are used as input for other modules that are responsible for generating collision-free motions.

For holding a given distance between target and robot, we use a method based on the disparity between two simultaneously shot images. The conjugate pair we use are the centers of gravity of the segmented color blobs in both images. Our model is shown in figure 15. The distance z can be computed as $z = \frac{bf}{x_l - x_r}$, which is precise enough for a

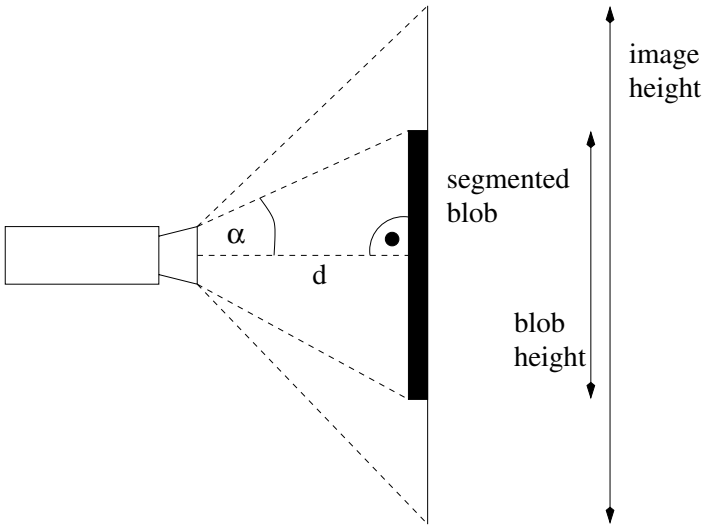


Figure 16. Distance calculation through blob height.

simple distance estimation to decide if the robot has to accelerate or decelerate. If the person is lost more than five times in sequence, a stop command is sent to the motion control. After redetection, the motion control receives new goal information.

A second approach requires only a single camera. The distance to the person being tracked is calculated from the size of the segmented color blob. Because different-colored shirts of several persons are almost identical with respect to their height, we use the assumption of a known height to calculate the distance to the object from monocular images. We use only the blob's height because the height (in contrast to the width) hardly changes when the person moves. This works fine as long as the segmentation of the color blob is appropriate. Figure 16 shows the strategy calculating the distance to the object. As experiments confirm, assuming shirts to be 350 mm high provides good distance measurements also for different persons wearing different shirts. With this information, distance calculation reduces to a simple triangle calculation:

$$\alpha = \text{opening angle} \frac{blobheight}{imageheight}$$

$$d = \frac{\text{assumed size}}{\tan \alpha}$$

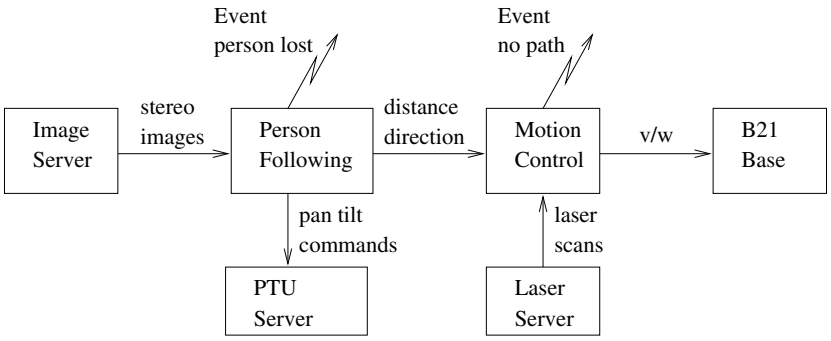
5 Integration on the Robot

5.1 Person Following

At the execution layer, the capabilities of the person-following module are modeled as two different behaviors. A behavior can be executed by putting it on the agenda of the execution layer. The startup behavior allows the model of the person to be built (figure 8). The task net for this behavior provides the following individual steps. First, speech output is used to ask the person to stand in front of the camera and to acknowledge this by speech input. The startup step is then executed within the person-following module. An event reports the result, and speech output is used to inform the person. In case of success, a unique identifier is provided to reference the person, and the startup step is finished.

Figure 17. Person following.

Figure 18. Data flow during person following.



The person-following behavior expects the identifier of the person presented before. Executing the person-following behavior includes a more complicated configuration at the skill level. The person-following module gets images from the image server and provides the ptu with control commands. The motion controller is provided with the distance to, and direction of, the person relative to the robot. This module gets laser scans and tries to move into the proposed direction without collision. Speed is adjusted to the distance between the person and the robot. Different events are used to synchronize execution with the description in the sequencing layer. The person-following module can report whether the person has been lost. In this case, the robot stops but tries to reacquire the person. The motion controller, for example, can report that no path is available. The appropriate event-handling procedure is invoked within the task net at the execution layer. The person-following behavior is shown in figure 17, the data flow among involved modules in figure 18.

5.2 Vision Module

The capabilities of the vision module are also represented by different behaviors at the execution layer. Each behavior includes a task net and a description of the configuration of the skill level. The task net describes the different execution steps and specifies how to handle events reported from the skill level. Depending on the behavior, different skills are involved and different events can occur.

As an example, figure 19 shows the task net of the vision-based object approaching behavior. The error-handling code is not included in this example. The behavior requires an object identifier to be specified (which references a knowledge base entry). First, the `follow-object` behavior moves close to the expected object location as provided by the knowledge base. Then, the `search-object` behavior is used to confirm

Figure 19. Task net for approaching an object.

```
(define-rap (goto-object ?objectid)
  (method
    (context (object ?type ?objectid ?color ?approach-distance))
    (task-net
      (sequence
        (t1 (follow-object ?objectid))
        (t2 (search-object ?type ?color => ?object))
        (t3 (move-to-object-blind ?object ?approach-distance))
))))
```

the expected object. If the object is near the expected position, the current object position is updated in the knowledge base. Because the object is out of sight when the robot has to move very close to it, the last step is a blind move to the updated position.

6 Experiments

The overall system is being used in our everyday office environment, where it has to fulfill different tasks. A RWI B21 system is used as the mobile platform. The robot is equipped with two color cameras with several automatic image-acquisition features that are activated during normal operation. All computations are performed onboard by two dual Pentium Pro 200 MHz systems running Linux. Normally, one processor handles the vision algorithms, and the other modules are distributed over the other processors.

6.1 Person Following

Using 192 $\times$ 144 images (color resolution of 8 bits per band) for person following, the cycle time is approximately 1 sec. for the combined color- and contour-based approach. This cycle time is achieved with the integrated system in which other tasks of the robot are running in parallel. Our robot successfully tracked different persons through our building at speeds up to 400 mm/s. The average speed normally is approximately 300 mm/s; the average length of a run is 5 min.

We carried out several experiments to show the performance and possible failure situations. The person-following behavior has been tested with several people wearing different shirts. A part of their NCC color representation is shown in figure 20. Using solely the NCC or histogram-based method, a person is lost only when he or she is occluded by other objects in the scene longer than five frames, when an object with the same color enters the scene and this color blob is larger than the shirt's blob, or when the person moves too fast and leaves the camera's field of view. Figure 21 shows two people wearing similar green shirts. Their representation in NCC space is shown by the two solid overlapping boxes in figure 20. A segmentation result from a tracking sequence is shown in figure 22. In this case, the left blob denotes the person who had to be tracked. Because the new blob is larger than the original one, the focus of attention jumped to the wrong color blob. Subsequently, the robot tracked the wrong person when relying solely on a color-based approach. The same error occurs when the background contains large color regions with almost the same color as the person's shirt.

Small amounts of varying illumination due to passing beneath lamps or moving next to windows have been handled robustly by removing intensity from the RGB color values. Figures 23 and 24 show different lighting conditions that can be handled without problems. However, errors due to strong illumination changes may still occur.

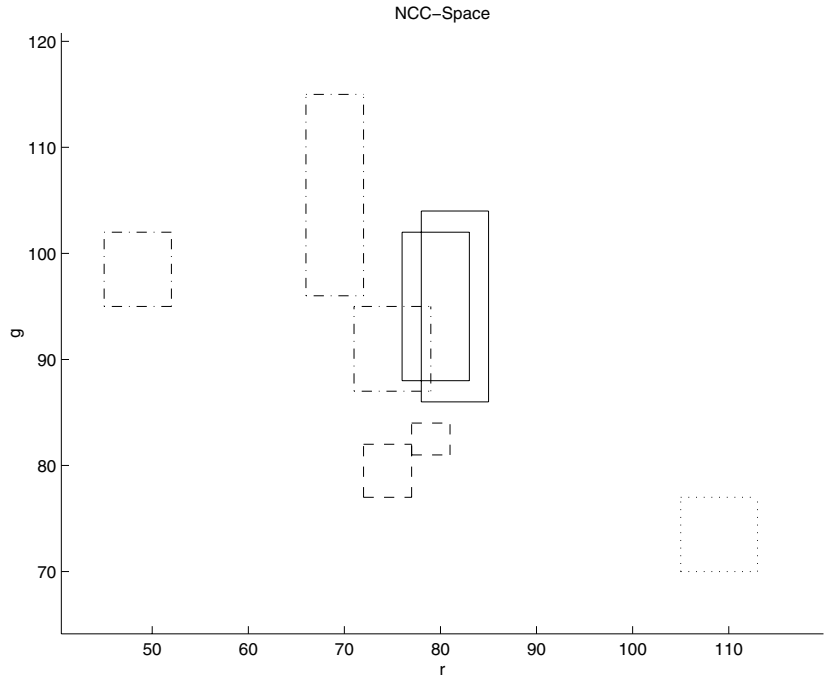


Figure 20. Several shirts in NCC color space.

Figure 21. Two people wearing similar-colored shirts.

Confusions due to persons wearing similar-colored shirts may not occur when using the combined color- and contour-based approach. Swapped persons are always detected by the contour-based approach due to their changed outline. Figure 26 shows a contour model derived from the original image shown in figure 25. One disadvantage, however, is the easy loss of the person's contour model as shown in figures 27 and 28. In this case the person to be tracked was occluded by another person which led to a loss of the model. Therefore the system reports a *lost person event*, stops, and requests a new startup procedure. Altogether this method is much more reliable referring to confusions of persons.

Because the person-following behavior is monitored by the execution layer, losing the sought person can be handled. In such a case, speech output is used to ask the person to present him- or herself once more to acquire a new model. The coordination is accomplished by task nets, in which context-dependent execution orders can be expressed comfortably. As one cannot always prevent the person-following behavior from becoming confused by dynamic obstacles or other similar persons, task

Figure 22. Segmented blobs leading to confusion.

Figure 23. People staying in bright light.

Figure 24. People staying in the dark.

nets are an easy way to specify how to cope with such situations as long as such situations are detected reliably. In particular, speech output is very helpful in interaction with people as the robot can ask the tracked person to present himself again in a suitable way to recapture the person.

In figure 29, the accuracy of distance measurement with stereo cameras is shown. The stereo distance measurement works fine as long as the segmentation is appropriate. Appropriate segmentation denotes that the center of gravity of both blobs (left and right camera) mark the same

Figure 25. Original image.

Figure 26. Contour model extracted from the original image.

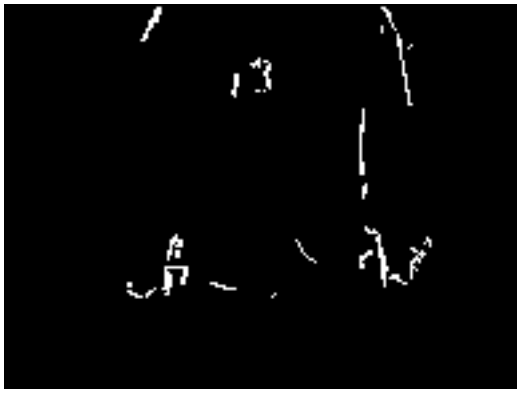


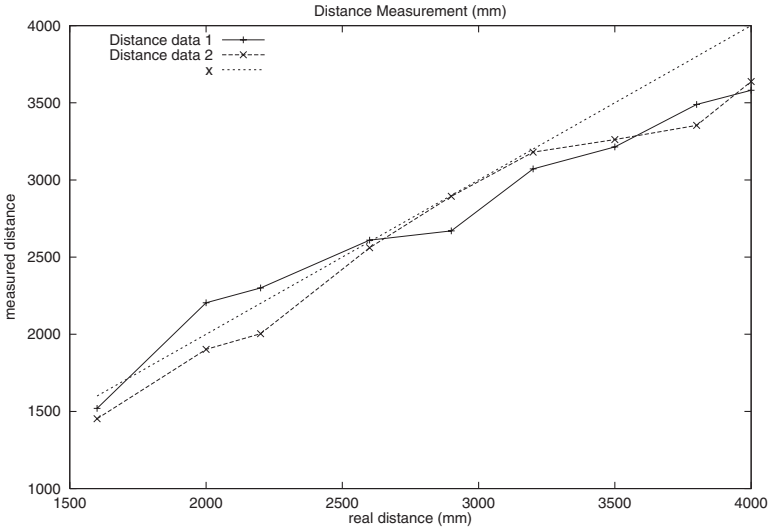
Figure 27. Partly occluded person.

point on the object's surface. If this corresponding pair differs, the distance measurement leads to wrong results. Also, changing illumination conditions can affect the color blob segmentation so that the corresponding blobs do not have the same shape (which leads to an erroneous distance measurement).

Both stereo and monocular approaches provide distance measurements with similar accuracy sufficient for our person-following behavior. The stereo approach is computationally more expensive but does not need the assumption of a known object height. The monocular approach has its advantage in speed but one has to cope with distance-estimation errors resulting from the height assumption. This assumption leads not to a bigger variance but to an always too small or too large distance. The

Figure 28. Loss of the person's contour model.

Figure 29. Accuracy of distance measurement with stereo cameras.



absolute value is not critical because the distance is used only to control the acceleration behavior of the robot.

The distance measurement fails when background blobs with similar color exceed the size of the shirt's color blob. (This is a rare event because the robot's motion control always ensures that the robot resides nearby the person, guaranteeing the shirt's blob is the largest in the field of view.)

6.2 Vision Module

Typical tasks involving the vision module are collecting objects (figure 30) in a certain area and putting them into appropriate trashcans (figure 31). The vision module has to look for objects and trashcans while exploring the area. Each recognized trashcan is stored in the knowledge base, which allows the robot to move to the nearest trashcan as soon as an object has been found and picked up. The robot drives with 200 mm/s when searching for objects. Using the ptu to scan the workspace, it takes approximately 6 sec. to inspect an area of 30 sq m. To pick up an object, the robot has to approach it in such a way that the gripper is placed correctly. This is done by a tracking behavior, in which the vision module provides heading information for the motion controller. An event signals when the object has been reached and can be picked up. The coordination of the different steps is accomplished by the execution

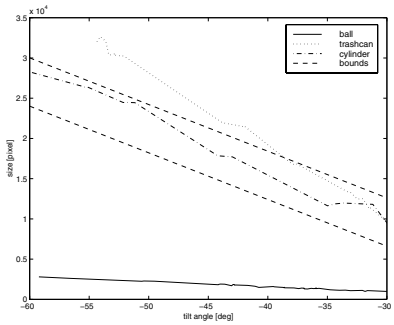
Figure 30. Some example objects.

Figure 31. Putting a ball into a trashcan.

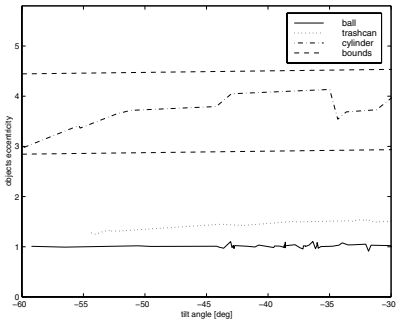
layer guided by the task net. Depending on the required behavior, different configurations of the vision module and other skills are selected and synchronized by events.

Some of the objects detectable by the vision system are shown in figure 30. The discrimination performance of the used features with respect to our objects is now discussed in more detail. A feature provides useful discrimination information if the feature value curve with respect to the tilt angle has a big distance to curves of other objects over all tilt angles. The correlation between the tilt angle and different approved

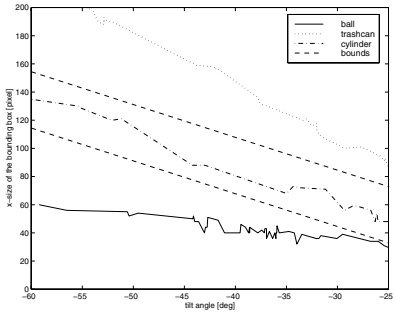
Figure 32. Shape feature curves.



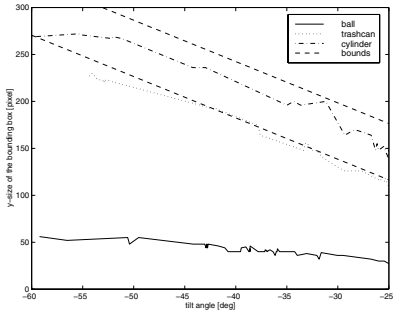
(a) Size depending upon tilt-angle



(b) Eccentricity depending upon tilt-angle



(c) X-size of bounding box depending upon tilt-angle



(d) Y-size of bounding box depending upon tilt-angle

features for the ball, trashcan, and cylinder is shown in figure 32. The object size in figure 32(a) is defined as the size of the color blob after the color segmentation and is measured in pixels. For many objects, the size is already sufficient for classification. The dashed lines in figure 32 show the upper and lower bounds of the feature values of an object that is assumed to be a cylinder. It's obvious that further information is needed for a robust classification of cylinders in this case. The object eccentricity, shown in figure 32(b), provides the necessary information. The eccentricity values of cylinders and trashcans differ strongly over all tilt angles. If color, size, and eccentricity do not allow objects to be discriminated, we use further features like the x size and y size of the bounding box, also shown in figure 32(c) and 32(d). Noticeable are the fluctuations of the feature value curves, which result from changing illumination conditions during measurement.

To show the robustness of our object classification method, we performed several experiments. A typical location is shown in figure 33 and is characterized by varying illumination conditions due to a mixture of synthetic and natural light. The robot drove straight on with different speeds. The objects shown in figure 30 were placed at different distances to the robot's path but in its field of view. The camera was fixed for all experiments and looked sidewards and downwards with an angle about 30 deg. each. The task of the robot now was to continuously look for all known objects within its field of view during its movement. The results are based on 160 different runs each with five objects to be classified. (The results are summarized in table 1.)

As one expects, the recognition rate decreases with increasing speed. These results were obtained without using a gaze control to center on found objects and to reclassify them because we wanted to quantify

Table 1. Classification Results.

Speed [mm/s]	Rate of recognition [%]
0	98.0 %
100	96.1 %
200	85.7 %
300	76.9 %

Figure 33. Cylinder scenario.

(a) Approaching lot in stock 1 (b) Driving to stock 2

the robustness of the object-detection and classification process itself. During these experiments, any object position in the image is allowed including peripheral positions where objects are warped due to lens distortions. During our experiments no false classification of detected object occurred. (This was achieved by restricted settings of interval boundaries in the classificator, which of course caused omitted objects.) Therefore, the results in table 1 show the number of omitted objects during our experiments. This result strongly supports the appropriateness of the color-based selection of regions of interest and the tilt-angle-dependent selection of interval boundaries to cope with sight-dependent object appearances.

However, some critical problems can hardly be handled with our approach. Failures occur if objects overlap and the occluded object cannot be seen completely. In this case, the computed feature values do not fit the intervals stored in the model database and are therefore mainly omitted. Another potential error is caused by the occlusion of an object with the same color. In this case, the color segmentation merges both objects to wrongly form a single large blob whose features are then calculated for the following classification step. Those values normally do not fit any of the known objects and again result in omitted objects. Because the robot moves, occlusion is mostly only a temporary problem at a particular viewing angle. Problems are caused only if the objects are very close to each other.

Another problem is raised by the robot's motion. Depending on the speed, the images are blurred and the feature values of the blurred color blobs do not fit very well. The same problem occurs with reflections caused by smooth surfaces of an object. This also leads to an irregular blob shape with feature values that do not correspond to the reference values. Further problems may appear if the illumination conditions vary strongly. In this case, the color segmentation possibly fails. The color segmentation is normally adjusted so that it is not so sensitive that it omits a potential object. Against that, the subsequent classification step is very restrictive.

The experiments showed that the proposed classification method is particularly suitable for a moving platform. It provides good recognition rates in the static case and acceptable results in the dynamic case in realtime. Therefore, we preferred it to other methods that provide a better rate but are normally much slower.

6.3 Cylinder Scenario

Figure 33(a) and 33(b) show an integrated scenario. The robot has to achieve different object configurations within the two stock areas. All orders are given via speech input. An order is, for example, "Bring all blue cylinders from stock 1 to stock 2". If the order is specified completely, it is translated into an agenda entry. In this particular case, the robot has to drive in front of stock 1. Then the vision-based object classification is activated to report the allocation of stock 1. Depending on the object type, either a vision- or laser-based approaching behavior is used. Cylinders are approached only with a laser-based maneuver because the final position has to be within 1 cm for grasping. When moving to stock 2, motion control has to consider the new shape of the robot. When reaching stock 2, free lots for placing the object are searched with the vision system. All task-relevant expectations are verified by sensing behaviors before executing a behavior relying on this information. This makes sure that, for example, changes of stock allocations during task execution are recognized and handled correctly. The overall coordination of single steps is completely handled by the sequencing layer. This also includes the handling of failure conditions and deciding when to stop execution because the final configuration can't be reached any longer.

7 Conclusion

We have described the vision modules on our mobile robot together with the currently used overall system architecture. The three-layer architecture supports the integration of different modules to form a complex system that can execute a wide variety of different tasks. Practical experience showed that the requirements of a mobile platform and the necessity to build an integrated system make specific demands on the vision component. These requirements are best met by using simple but robust approaches. Due to the high bandwidth needed by vision systems, task-dependent behavior selection is crucial. By a context-dependent configuration of vision-based behaviors, limited resources can be shared adequately. Therefore, the vision components have to be highly modularized and have to provide a configuration facility. In particular, the used architecture shows advantages with respect to the declarative representation of different configurations and the consistent representation of task-dependent parameter settings. This allows the integration of several approaches with overlapping capabilities and specific advantages that can be activated depending on the current situation, thus avoiding the need for a single approach to handle all occuring situations.

The kind of integration described in this paper provides a very close integration at the level of task execution and behavior coordination. However, the interaction between the symbolic environment model and the vision components has to be extended further. Currently, expected objects can be confirmed by the vision system as long as they are within the field of view. Also, newly seen objects or incompatibilities between object attributes can be reported. The more difficult problem, however, is to decide whether an expected object disappeared. This can be done

only for objects in the field of view. For objects lying outside the field of view, nothing can be said. Therefore, it is necessary to have a close interaction with a geometric model that allows the determination of which objects are expected to be visible. Hence, one part of our current work within the SFB aims at a deeper integration of vision at the level of map building and representation of the environment. Those questions are particularly relevant to bridge the gap between standalone vision algorithms and autonomous systems using vision as one of their basic sensing capabilities.

Acknowledgments

This work is part of project C3 (http://www.uni-ulm.de/SMART/Projects/c3.html) of the SFB 527 sponsored by the Deutsche Forschungsgemeinschaft.

References

[1] A. Azarbayejani, C. Wren, and A. Pentland. Real-time 3-D tracking of the human body. In *Proceedings of IMAGE'COM 96*, 1996.

[2] R. P. Bonasso, R. J. Firby, E. Gat, D. Kortenkamp, D. Miller, and M. Slack. Experiences with an architecture for intelligent reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2), 1997.

[3] S. A. Brock-Gunn, G. R. Dowling, and T. J. Ellis. Tracking using colour information. In *3rd ICCARV*, pages 686–690, 1994.

[4] Q. Cai, A. Mitchie, and J. K. Aggarwal. Tracking human motion in an indoor environment. In *2nd International Conference on Image Processing*, October 1995.

[5] J. Canny. Finding edges and lines in images. Technical Report AI-TR-720, MIT Artificial Intelligence Lab, 1983.

[6] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[7] R. J. Firby. *Adaptive Execution in Complex Dynamic Worlds*. PhD thesis, Yale University, 1989.

[8] E. Gat. ESL: A language for supporting robust plan execution in embedded autonomous agents. In *Proceedings of the IEEE Aerospace Conference*, Los Alamitos, California, 1997. IEEE Computer Society Press.

[9] D. M. Gavrila and L. S. Davis. Towards 3-D model-based tracking and recognition of human movement: A multi-view approach. In *International Workshop on Face and Gesture Recognition*, 1995.

[10] J. S. Gutmann and C. Schlegel. AMOS: Comparison of scan matching approaches for self-localization in indoor environments. In *IEEE Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots*, pages 61–67, 1996.

[11] D. Huttenlocher, J. J. Noh, and W. J. Ruckli. Tracking non-rigid objects in complex scenes. Technical Report TR92-1320, Department of Computer Science, Cornell University, 1992.

[12] J. Koehler and B. Nebel. IPP—The Interference Progression Planner. http://www.informatik.uni-freiburg.de/~koehler/ipp.html.

[13] J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos. Extending planning graphs to an ADL subset. In *Proceedings of the European Conference on Planning*, pages 273–285. Springer-Verlag, 1997.

[14] D. Kortenkamp, R. P. Bonasso, and R. Murphy. *Artificial Intelligence and Mobile Robots—Case Studies of Successful Robot Systems*. AAAI Press and The MIT Press, Menlo Park, CA, 1998.

[15] F. Noreils. Integrating error recovery in a mobile robot control system. In *IEEE International Conference on Robotics and Automation*, 1990.

[16] F. J. Radermacher. Cognition in systems. *Cybernetics and Systems*, 27:1–41, 1996.

[17] C. Richards, C. Smith, and N. Papanikolopoulos. Detection and tracking of traffic objects in IVHS vision sensing modalities. In *Proceedings of the Fifth Annual Meeting of ITS America*, 1995.

[18] C. Schlegel. Fast local obstacle avoidance under kinematic and dynamic constraints for a mobile robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Victoria, Canada, 1998.

[19] C. Schlegel, J. Illmann, H. Jaberg, M. Schuster, and R. Wörz. Vision based person tracking with a mobile robot. In *Proceedings of the British Machine Vision Conference*, volume 2, pages 418–427, 1998.

[20] C. Schlegel and R. Wörz. The software framework SmartSoft for implementing sensorimotor systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 1610–1616, Kyongju, Korea, 1999.

[21] M. Sullivan, C. Richards, C. Smith, O. Masoud, and N. Papanikolopoulos. Pedestrian tracking from a stationary camera using active deformable models. In IEEE Industrial Electronics Society, *Proceedings of Intelligent Vehicles*, pages 90–95, 1995.

[22] C. Wong, D. Kortenkamp, and M. Speich. A mobile robot that recognizes people. In *IEEE International Joint Conference on Tools with Artificial Intelligence*, 1995.