A Hierarchical World Model with Sensor- and Task–Specific Features*

Alexa Hauck, Norbert O. Stöffler

Department of Process Control Computers
Prof. Dr.-Ing. G. Färber
Technische Universität München, Germany

Abstract

The design of mobile robots which can cope with unexpected disturbances like obstacles or misplaced objects is an active field of research. Such an autonomous robot assesses the situation by comparing data from one or more sensors with an internal representation of its environment. In this paper we present a hierarchically structured world model that combines a general geometric object representation with sensor- and task–specific features and therefore can be used for various sensors and perception tasks. By predicting only those features that, first, can be detected by the sensor and, secondly, are relevant for the current perception task, sensor data interpretation gets faster and more robust. This is illustrated at an exemplary perception task concerning the video–based determination of the joint states of articulated objects.

1 Introduction

This work is part of a research project towards development of autonomous mobile robots (AMR) which can fulfil service and transport tasks in structured environments like office buildings and industrial plants. For planning its actions such an autonomous robot needs an internal representation of its environment (*environmental* or *world model*). Most modelling techniques described in literature are specialized on a certain application; thus such models are well adapted to specific perception tasks and sensors [1] or environments [4] but cannot be used in a general way. Information is often stored merely on feature level; sometimes several models are held in parallel and used independently for different tasks. In contrast to this our approach aims at designing generally applicable world models that form the central knowledge base for autonomous mobile robots. For this we have developed a general modelling framework, which combines information needed for various sensors and robot tasks into one consistent description on different levels of abstraction. Continuous update of the model in the case of dynamic or partially unknown environments is realized by considering the model as a set of hypotheses regarding the positions and states of the elements of the world, including the robot itself, which are tested and eventually modified by comparing them with sensor data. This is performed by several perception tasks that operate in parallel and interact with the model on different levels of abstraction, according to the parts they regard as being hypothetic. Typical tasks are the localization of the robot itself, recognition and registration of objects and reconstruction of unknown environments (exploration). To enable the coupling of the model with planning tasks, a symbolic interface allowing to access information on a more abtract level is integrated.

The paper is organized as follows: Section 2 describes the internals of the model, detailing the object structure (section 2.1), features (section 2.2), modelling conventions and visibility calculation (section 2.3). Section 3 deals with the application of the model, first describing the different ways to access it (section 3.1), then illustrating them by an application framework containing several perception tasks (section 3.2) and finally describing an experimental example (section 3.3). Though the model is designed for multi–sensor systems, in this paper we concentrate on its application to video sensors.

*The work presented in this paper is supported by the *Deutsche Forschungsgemeinschaft* as part of an interdisciplinary research project on "Information Processing in Autonomous Mobile Robots" (SFB 331).

2 The model

2.1 Object structure

To permit sensor independent abstractions the model structure is based on three–dimensional solid modelling techniques.

Elements of the world influence sensor images in two ways. They can be the source of sensor–specific features and they can hide other elements. The latter aspect is modelled by a polyhedral boundary representation.

If possible, sensor–specific features are calculated from the boundaries using the corresponding sensor model. In the case of a video sensor such a sensor model is difficult to obtain because of its dependency on various factors like colour and illumination, which aren't easy to model. Therefore in a first step a very simple camera model is used, assuming perspective projection and neglecting illumination effects. Video–specific features, up to now solely edges, are modelled by line–segments which are based on the same set of vertices as the boundaries but do not necessarily coincide with boundary edges. This dualism allows the representation of the boundaries by exclusively convex polygons which facilitates the visibility calculation. In a second step sensor–specific information is integrated by comparing model data with a collection of images as described in section 2.3.

To initiate a prediction the boundaries and features inside the vision pyramid are determined and their visibility is tested as described in section 3.1. To access the vertices which are located inside the vision pyramid appropriate index structures are necessary. In first experiments demonstrating localization in a static environment, a two–dimensional spatial–tree has proven to be an efficient index structure for otherwise unrelated world elements [12].

To allow for more complex perception tasks and non–static environments a hierarchic structure with additional symbolic information has been developed [7] (see figure 1).

Elements of the world are aggregated to form *named objects*. Because it is neither possible nor necessary to describe the complete environment of the robot in terms of distinguishable, named objects, a pseudo–object called *background* is introduced. It encompasses all world elements without special object assignment.

The description of a named object is built up recursively. An object can contain so–called *member–objects*, which are connected by a joint which exhibits exactly one rotatory or translatoric degree of freedom, following the conventions used in manipulator kinematics. Each object or member–object has its own coordinate system or *frame*, whose relation to that of the parent–object is described by a homogenous transform matrix, following the Denavit–Hartenberg–formalism [3]: The z–axis coincides with the axis of rotation respectively translation, its orientation is chosen in a way that the joint variable appears with the correct sign. Since objects may possess more than one member–object on each level and thus the kinematic chain may fork, the Denavit–Hartenberg–conventions for the x–axis can not be applied. The possible positions of a joint are normalized to the unit interval allowing a unified treatment of joint–states; additionally there exists a state called *unknown*. To deal with unknown states during a prediction, the space potentially being occupied by a moving member–object is stored as an additional boundary, called *mask*.

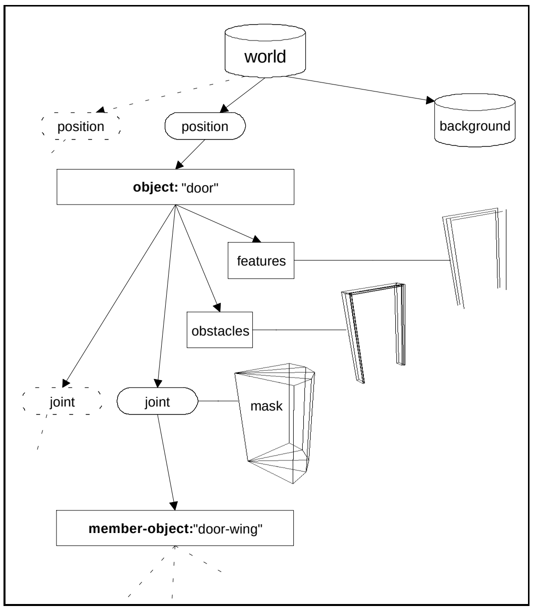

Figure 1 Object structure

Each branch in the object–tree carries its own boundary and feature description. The representation of features is extended by the possibility of defining aggregations of simple features and attributes to form complex ones. Those complex features may be task–specific to alleviate special matching problems (see section 2.2).

Geometrically identical objects form an *object class*. The invariant parts of an object description are stored only once for each class; the objects (i.e. the instances of a class) differ in their individual positions and joint states.

Figure 2 shows an exemplaric model of a corridor,

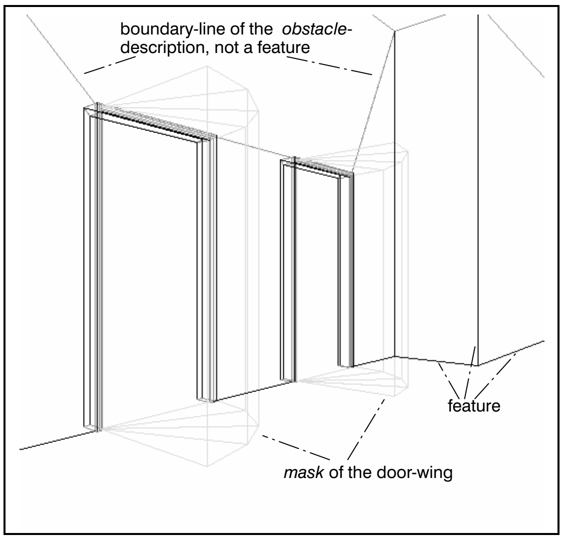

Figure 2 Model of a corridor

consisting of two doors of the same class and some walls which are part of the background. For this depiction no hidden lines were removed. Black lines are video–specific features, dark–grey lines stand for boundaries which do not coincide with features and the light–grey lines represent the masks of the door–wings. The door–wings themselves are left out for simplification reasons.

2.2 Features

Raw sensor data is preprocessed to extract sensor–specific features, in the case of a grey–scale video sensor typically edge segments. In order to enable an easy match and comparision of sensor and model data, the predicted model features should resemble the extracted ones. In our first approaches video–specific features therefore consisted of 3D line–segments, which were derived from the boundary description and then projected into the image plane for a given view point. Predicting model features alleviates some of the typical problems in image interpretation: *Segmentation* is aided by providing areas of interest, the location and direction of a model segment is used for the *grouping* e.g. of broken edge chains, and at the same time enables the *quantitative evaluation* of a prospective match. The validity of this approach has already been shown in experiments on a mobile plattform equipped with a video camera and a microwave radar [15].

Since the video–specific features were derived using a very coarse sensor model, the matching process would sometimes get disturbed by model features that could not be seen by the sensor due to lack of contrast. In addition, information that could alleviate the matching problem even further, as e.g. relations between edges, could not be accessed at feature level even though it was actually available in the model. These considerations lead to the development of a new hierarchical feature structure.

Based on fundamental features like the aforementioned line–segments new feature types can now be defined which can contain other features and attributes, forming *aggregated features.* A typical example would be a type *contour* containing a list of line–segments and an attribute stating if the contour is closed. To enable an automated derivation of features from the boundary description, feature types have to be defined geometrically.

In a first step feature types can be divided into two classes: *topological features* and *task–specific features.* Topological feature types model relations between features. Up to now only pairs have been considered, further aggregation will be examined. The grouping of two or more line–segments in 2D and 3D has been discussed widely (see e.g. [11]). Generally speaking a pair of line–segments can either be *parallel*, *intersecting* or *skew*; parallelity can be further differenciated in overlapping and non–overlapping, intersections can be classified according to their shape (*L–shape, V–shape, T–shape, λ–shape, ...*) and according to the location of the point of intersection, which can be lying on none, one or both line–segments (the latter case can be termed a *junction*). A combination of parallelity and intersection yields a *collinear* pair. Of all possible relationships currently only three are modelled by feature types: Parallel overlapping pairs (*parallelity*), collinear junctions (*continuity*) and non–collinear V- or L–junctions (*corner*). T- and λ–junctions are modelled as three line–segments forming one continuity and two corners. It has to be noted that these 3D–features are only quasi–invariant under perspective projection: Parallel line–segments are parallel in the image only if they are parallel to the image plane, non–collinear junctions may be collinear after projection. Since these effects are position–dependent, they are treated when calculating the visibility of features for a given viewpoint (see section 3.1).

Predicting those topological features instead of their parts reduces both the number of correspondences to test and the probability of mismatches. This is especially efficient when using edge extraction methods that preserve the corresponding topological image information (e.g. see [14]). Otherwise a specialized matching routine is assigned to the feature type. In the case of a parallel pair for example this would mean looking for a pair of image edges and assigning the "left" image edge to the "left" model edge, which is faster and more robust than single line matching using a minimum distance criterion.

Task–specific feature types are specialized on a certain perception task, they contain the relevant model information for that task. Examples are given in section 3.3, that describes the realization of an exemplary perception task. Task–specific feature types, too, possess a specialized matching routine, that builds upon the ones of its members and uses the knowledge about their relation. By integrating this procedural information into the model a mixed top–down/bottom–up matching process with feedback is implemented implicitely.

2.3 Modelling objects

The geometrical information about the object to model can be won in two ways: By exact measurement or by reconstructing it from sensor data. For the latter a kind of inverse sensor model is needed to convert sensor–specific features into sensor–independent data; possibly several sensors have to be used to get complementary results. This task is addressed in various fields of research, from remote sensing to autonomous mobile robots exploring their environment, the latter being a current research topic in our group [2].

In the absence of a precise sensor model the question arises how it can be asserted that the modelled features can actually be seen by the sensor. The visibility of a feature is position–dependent in a double way: A feature can be invisible because it is hidden by another object from the current point of view or because it cannot be detected by the sensor. The latter occurs e.g. due to a lack of contrast between two parts of an object.

The first case is treated on–line while predicting object views (see section 3.1), the second while constructing the model representation. Each object is equipped with a map that denotes which of its features can be seen from a certain position relative to the object. The map is object–centered, positions are stated two–dimensionally in polar coordinates. This simplification reflects the situation that the robot and its camera are moving in a plane; a generalization to unconstrained movements would require the transition to three–dimensional spheric coordinates. An implicit assumption is that the camera is always looking directly at the object, which is the case for most perception tasks. This map can also be seen as modelling the *aspects* [9] of the object. It can therefore be used to calculate the *aspect graph* or *tree* which is the basis for many object recognition algorithm (see e.g. [13]).

The notion of visibility can be extended by a quantitative dimension, called the *quality* of a feature. This facilitates the matching process considerably by establishing a ranking of features, which can be used to establish the order of processing and to decide in the case of contradictory match results. The quality of a feature reflects the expected probability of a correct match; in the case of task–specific feature types quality also encompasses the suitability of a feature for the specified task at the given position.

For the modelling of video–specific features a hybrid mechanism is used: First a set of prospective features is derived from the boundary description by computing all boundary edges of all faces of the object. The visibility and quality of the features is then determined by taking video images of the object from different, known standpoints and starting the matching process. Those features that can be matched correctly are entered in the visibility map for the current position; features that cannot be seen from any position are removed from the model. The quantitative result of the matching algorithm forms the quality attribute of the feature. By this combination of analytical and empirical methods the lack of a complete sensor model is partly compensated; it has to be noted, though, that by using the matching result to express the quality of a feature, features become partly algorithm–specific. This is advantageous as long as just one matching algorithms is used.

Aggregated features require a more complex visibility calculation to cover the case that some members are visible but others are not. Currently a rule–based approach is implemented, which allows customized visibility definitions for basic line features depending on a set of properties like visibility of endpoints, fragmentation and direction. The visibility of the aggregated feature is then derived using logical expressions combining the visibilities of the members.

3 Application

3.1 Model Access

The model can be accessed on three levels: classes, objects and whole world. The most important reading access is the request of a feature prediction, but also the attributes or states of objects like the opening angle of a door can be queried. Writing accesses include changing the state of an object, updating the boundary or feature description and inserting newly explored elements.

The access is controlled by two virtual pointers called *focus* and *zoom*. The focus points on the task relevant part of the model, i.e. on the world, on a single object or a single class. After "focussing" e.g. on an object the states of a private copy of this object are accessible. This allows testing hypothetic states without changing the world model. The zoom influences the result of the feature prediction in a way comparable to a camera zoom: After pointing it on a node of the object–tree only features of the downward parts are predicted. Note that still all boundaries are used for the visibility calculation. Both, focus and zoom, can be moved step by step up and down the object–tree. The use of those virtual pointers also encapsulates the internal representation of the object–tree, allowing further optimizations without changes in the model access interface.

To initiate a sensor view prediction the boundaries inside the vision pyramid are used to generate a depth map by a so called z–buffer algorithm [6]. The polygons are projected on the viewing plane and rendered into a two dimensional array of pixels. During the rendering only the depth of the nearer pixel is stored in the array. The result is a two–dimensional, rastered map which contains the nearest depth value for each pixel. In the second stage the features inside the vision pyramid are tested against this map. The three–dimensional line segments are also projected and rendered. For each pixel which has a lower or equal z–value than stored in the depth–map the corresponding three–dimensional line subsegment is declared visible.

In difference to computer graphic applications the same line rendering algorithm is used for polygon edges and feature lines. This guarantees that a feature, which always has to coincide with an boundary edge, can never be hidden by adjacent boundaries due to quantization errors. This combined two–dimensional/three–dimensional algorithm has several advantages: Z–buffering is quite fast and widely supported by computer graphics hardware because of its simplicity. The pixel size can be controlled dynamically by the number of polygons in the vision pyramid to guarantee real–time behaviour [12]. The result of the visibility calculation are three–dimensional line segments which can still be represented by six parameters and matched by either two- or three–dimensional algorithms.

3.2 Application Framework

An application framework for the world model is shown in figure 3.

Several perception tasks are engaged in keeping the internal representation of the robot's environment up to date by testing stored information against sensor data. They interact with the model on different levels of abstraction according to the parts of the model they regard as hypothetic. Each perception task is implemented by a separate client module which extracts its own relevant features from the sensor data and simultanously requests predictions from the model server. Then it compares the two sets, interpretes the difference and updates the model accordingly. Interferences between the quasi–parallel model accesses of different tasks are avoided by private communication channels, called *accessors*. These accessors contain the current set of parameters, like assumed camera pose, states and the two pointers focus and zoom. To allow successive tests of similiar hypotheses, the parameters are handed to the model incrementally: E.g. a client can first set the camera pose, focus on the object of interest and then alternatingly set the state of a joint and get a prediction without having to specify pose and object again each time. Changes made by the client remain private until the client explicitely requests to make them public.

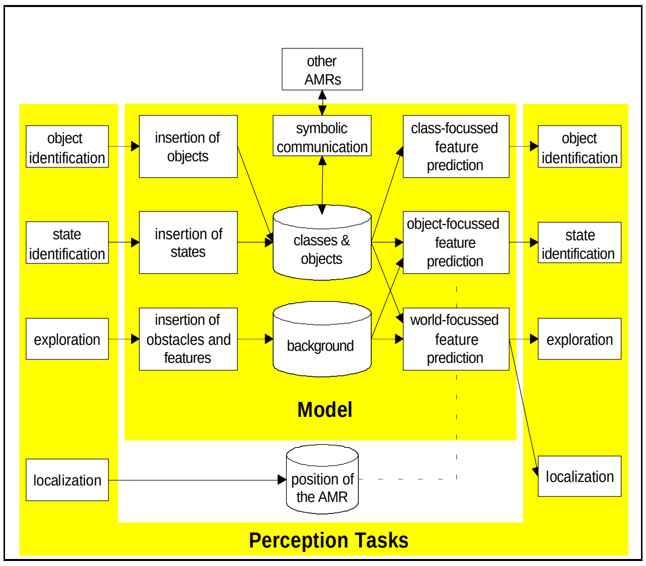

Figure 3 Application framework

A common parameter which is assumed to be known by several perception tasks is the position of the robot itself. It is updated by a task called *localization* [15]. For the prediction all world elements belonging to objects or the background are taken into account at their current states. Thus the accessor of this task is "focussed on the world". Since the quality of the pose estimation depends on an accurate match only features with guaranteed visibility should predicted. Member–objects with unknown joint state are replaced by the appropriate *mask*. The projection of this mask into the z–buffer literally ”masks out” features which may be hidden by the moveable member–object.

Severe mismatch of the features predicted for the current robot–position and the extracted ones indicate either a change of the features' position or the presence of new objects. Therefore a second task called *exploration* is charged with evaluating these mismatches and eventually updating feature positions respectively inserting new elements on various levels into the model [2, 5]. For this purpose it tracks and consolidates new features, tries to reconstruct the boundary description and initiates object identification. If none of the known object classes can be matched, the new features and boundaries are inserted into the model as part of the background. This at least prevents collisions and further mismatches.

Following object–oriented literature the *state* of an object can be defined as the set of the current values of its attributes. Applied to our object structure the state encompasses pose and motion of an object and the current states of its joints; *state identification* therefore includes pose and motion estimation and the identification of joint states. It is the task with the most model interactions, including focussing on the regarded object, recursive testing of hypothetic states and movements in the object–tree, thus being an ideal application for testing and illustrating the model. An experimental example is described in section 3.3.

In the experimental framework, several algorithms for *object identification* have been examined. All have the need for feature predictions for the assumed object in common [2, 10]. The accessor is “focussed on object classes” and poses are supplied in class–relative coordinate systems. If an object finally is identified, a new instance is created and inserted into the model.

In addition to these sensor–specific access channels, information can be retrieved and manipulated on a symbolic level. Independently operating robots can communicate about the environment by exchanging object names and attributes, i.e. states and positions, via a symbolic communication medium [16].

An experimental version of this framework has been realized as part of the interdisciplinary research project SFB 331 (”Information Processing in Autonomous Mobile Robots”) with different groups working on the individual tasks.

3.3 Application example

How to access the model and make use of the information stored in it shall be demonstrated on an exemplary perception task, the video–based determination of the joint states of an articulated object. In contrast to 3D-pose estimation this problem is only rarely addressed in literature (e.g. in [8]), not least because an articulated object may present many different views to a camera, thus soon leading to a combinatorical explosion in conventional localization and matching algorithms. The presented recursive object structure on the other hand allows breaking down this complex perception task into simpler ones: First the the static part of the object is localized and with it the joint axes of its member–object, then the joint state of the first member–object is determined, followed by that of its own member–object and so on, recursively fixing the degrees of freedom.

Our experimental setup consisted of a single off-the-shelf grey–scale CCD–camera without dedicated image processing hardware. Figures 5 and 6 show video images of a whiteboard, an object with one translatory and two rotary degrees of freedom; extracted features[2] are inserted in white, the model features in black.

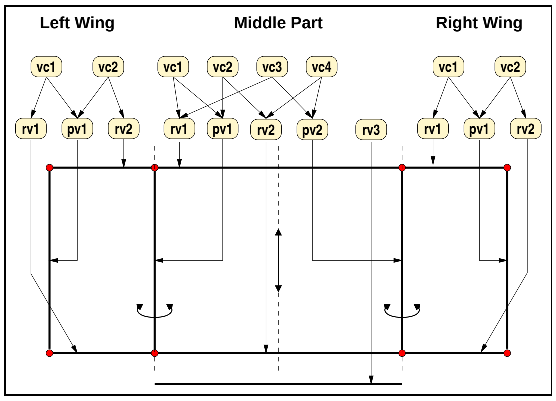

Figure 4 Variant features of the board

To identify a joint state three task–specific types of features have been found to be appropriate: *Radial variant edges* (edges whose starting point coincides with the axis), *parallel variant edges* (edges that are parallel to the axis) and *variant corners*, which connect a radial and a parallel variant edge. The video–specific features of the board are depicted in figure 4 for one side. The middle part possesses five line–features, that form three radial, two parallel variant edges and four variant corners. The two wings each possess two radial variant, one parallel variant edge and two variant corners each. To determine a joint value it is suffi-

[2]We currently use an algorithm motivated by an electrodynamic analogy for the extraction of line segments which preserves topological information[5].

cient to locate one variant corner in the image; the state can then easily be calculated by intersecting the path of the variant corner (circle in the case of a rotary, straight line in the case of a translatorical joint) with its corresponding projection ray. This again is facilitated by using model information, in this case the transformation matrix between camera and object frame.

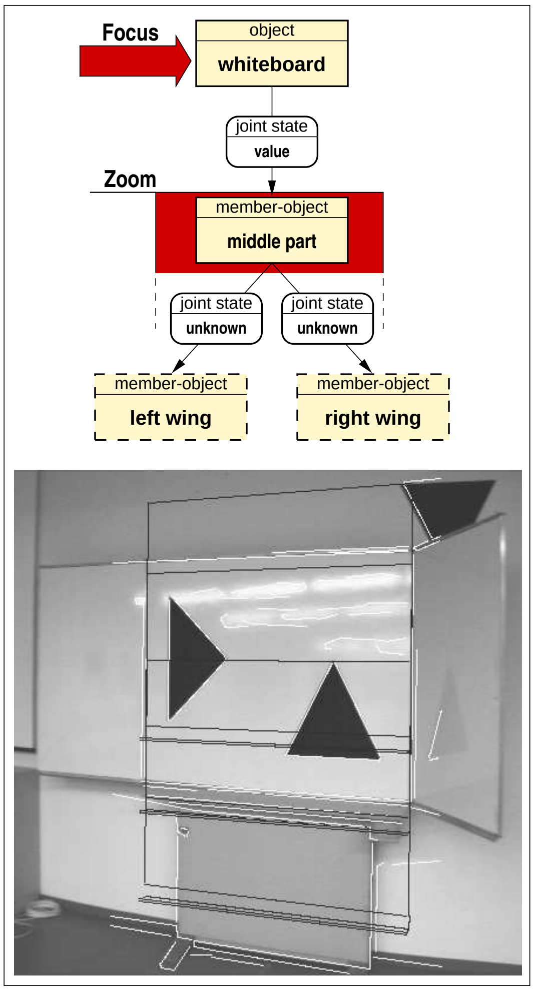

Figure 5 Determining the state of the middle part

At the beginning all states are unknown. After the localization of the invariant part (not shown) the first member–object is zoomed on and predictions for several states are requested (figure 5). The features of the wings are not predicted since their state is still unknown. For each "snapshot" model and image edges are matched; the corner with the best match result is used to calculate the joint state. The snapshots are also used to define the region of the image where edges are to be extracted, thereby integrating model information on both sides in the matching process. This could also be realized by projecting the corresponding mask.

After updating the model the focus is moved to the middle part, the zoom to its first member–object, the left wing, and again snapshots are requested (figure 6).

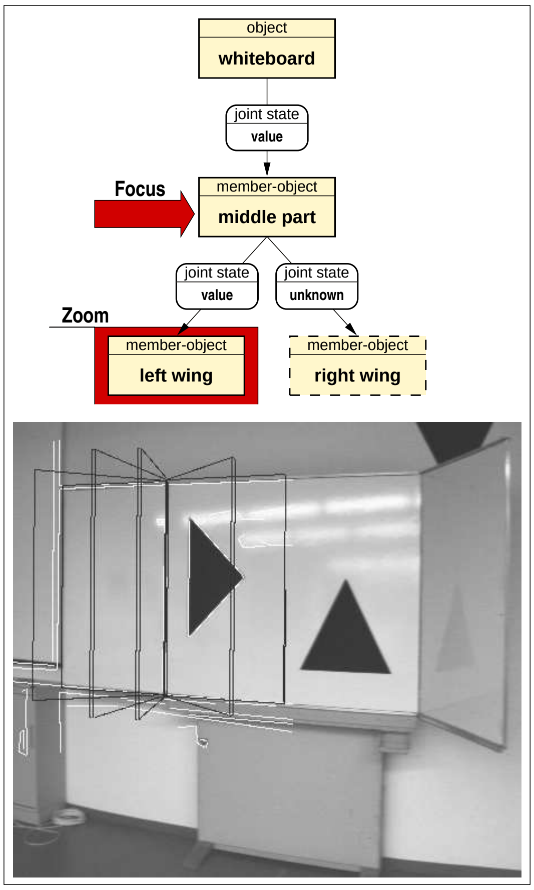

Figure 6 Determining the state of the left wing

After an analogous treatment of the right wing the identified states are made public; the information can now be accessed by other users, too.

This recursive algorithm has been found to be very robust; even occlusion by member–objects can be handled. Currently the determination of one joint state can be accomplished in less than $0.5s$ ($0.3s$ for 6 snapshots, $0.1s$ for edgel extraction, $50ms$ matching and joint state calculation). For the simpler case of determining the opening angle of a door (only one degree of freedom) this algorithm has been successfully implemented on a mobile robot that has to navigate in an office environment.

4 Conclusion

We have presented a hierarchic world model with sensor- and task–specific features which facilitates sensor data interpretation by predicting only those features, that can actually been "seen" by the sensor. By "focussing" on the object of interest and "zooming" on parts of it, task–relevant information can be accessed easily. This has been demonstrated on an exemplary perception task, the determination of joint states for objects with multiple degrees of freedom.

Future work will include the analysis of task–specific visibility and the design of further aggregated feature types for other perception tasks like visual servoing. The object–oriented structure will be expanded to derivation concepts to allow the fusion of similiar classes to abstract ones. To achieve real time behaviour of the model accesses further index structures and stategies for incremental visibility calculations according to incremental changes of the parameters will be evaluated.

References

[1] M. Buchberger, K. Jörg, und E. von Puttkamer. Laserradar and sonar based world modelling and motion control for fast obstacle avoidance of the autonomous robot mobot-iv. In *Proc. IEEE Int. Conf. Robotics and Automation, Atlanta*, 1993.

[2] D. Burschka und C. Eberst. Exploration of Unknown or Partially Known Environments. In *2. Asian Conference on Computer Vision*, pages 727–731, 1995.

[3] J. Denavit und R. Hartenberg. A kinematic notation for lower–pair mechanisms based on matrices. *Journal of Applied Mechanics*, pages 215–221, june 1955.

[4] E. Dickmanns. Active vision through prediction-error minimization. In *Active Perception and Robot Vision*, pages 71–90. Springer Verlag, 1992.

[5] C. Eberst, D. Burschka, G. Färber, A. Hauck, G. Magin, und N. O. Stöffler. A System Architecture Supporting Multiple Perception Tasks on an Autonomous Mobile Robot. In *4th Int. Symp. on Intelligent Robotic Systems, SIRS'96*, 1996.

[6] J. Foley, A. van Dam, S. Feiner, und J. Hughes. *Computer Graphics - Principles and Practice.* Addison Wesley, Reading, Massachusetts, 1990.

[7] A. Hauck und N. O. Stöffler. A Hierarchic World Model supporting Video–Based Localization, Exploration and Object Identification. In *2. Asian Conference on Computer Vision*, pages 176–180, 1995.

[8] Y. Hel-Or und M. Werman. Constraint–Fusion for Interpretation of Articulated Objects. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition*, pages 39–45, june 1994.

[9] J. Koenderink und A. van Doorn. The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211–216, 1979.

[10] S. Lanser, O. Munkelt, und C. Zierl. Robust video-based object recognition using cad models. In U. Rembold, R. Dillmann, L. Hertzberger, und T. Kanade, editors, *Proc Conf. Intelligent Autonomous Systems*, pages 529–536. IOS Press, 1995.

[11] D. G. Lowe. *Perceptual organization and visual recognition.* Kluwer Academic Publishers, 1985.

[12] G. Magin, A. Ruß, D. Burschka, und G. Färber. A dynamic 3d environmental model with real-time access functions for use in autonomous mobile robots. *Robotics and Autonomous Systems*, 14:119–131, 1995.

[13] O. Munkelt. Aspect–trees — generation and interpretation. *CVGIP: Computer Vision and Image Understanding*, 61(3):365–386, may 1995.

[14] C. Rothwell, J. Mundy, und B. Hoffman. Representing objects using topology. In *Proceedings of the International Workshop Object Representation for Computer Vision*, 1996.

[15] A. Ruß, S. Lanser, O. Munkelt, und M. Rozmann. Kontinuierliche Lokalisation mit Video- und Radarsensorik unter Nutzung eines geometrisch-topologischen Umgebungsmodells. In *9. Fachgespräch "Autonome Mobile Systeme", München*, pages 313–327, 1993.

[16] J. Schweiger, A. Koller, und K. Ghandri. A distributed real-time knowledge base for teams of autonomous systems in manufacturing environments. In *Proc. of the Seventh Int. Conf. on Industrial and Engineering Appl. of Artificial Intelligence and Expert Systems*, May–June 1994.