Edge Projected Integration of Image and Model Cues for Robust Model-Based Object Tracking*

M. Vincze, M. Ayromlou, W. Ponweiser, M. Zillich

Institute of Flexible Automation, Vienna University of Technology

{ma,wp,vm}@infa.tuwien.ac.at, www.infa.tuwien.ac.at

Abstract

A real-world limitation of visual servoing approaches is the sensitivity of visual tracking to varying ambient conditions and background clutter. This work presents a model-based vision framework to improve the robustness of edge-based feature tracking. Lines and ellipses are tracked using Edge Projected Integration of Cues (EPIC). EPIC uses cues in regions delineated by edges which are defined by observed edgels and a priori knowledge from a wire-frame model of the object. The edgels are then used for a robust fit of the feature geometry, but this sometimes will result in multiple feature candidates. A final validation step uses the model topology to vote for the most likely feature candidates. EPIC is suited for real-time operation. Experiments demonstrate operation at frame rate. Navigating a walking robot through an industrial environment shows the robust-

* This work has been supported by the RobVision project Esprit 28867, the Austrian Science Foundation (FWF) with grant P13167-MAT and project GRD1-1999-10693 FlexPaint.

ness to varying lighting conditions. Tracking objects over varying background indicates robustness to clutter.

1 Introduction

In recent years robots began to invade service applications. First mobile systems, which navigate in supermarkets or hospitals, are commercially available. In many laboratories robots are developed to aid the elderly with tasks for daily life. Specific research efforts envision a closer interaction of humans and robots. The most prominent example is the development of humanoid robots, primarily in Japan [InYo00, Hond01].

A common denominator of these systems is the need to interact with the environment. Two functions of interaction are required in almost all cases of autonomous robotic systems: navigation and object grasping. Both functions require some form of sensing to adapt and to react to present situations. Vision is by far the most powerful sensor to provide the information needed for object grasping and navigation. Hence, several groups are developing systems that use models of the environment or the objects for detection and visual servoing (e.g., workshop at ICRA 1998 [ViHa00] or the ICVS 1999 conference [Chri99]). The technique of controlling a machine using visual input is referred to as visual servoing [HuHa96]. When analyzing the capabilities of the systems presented, two difficulties are encountered regularly:

- Operation of the vision controlled robot arm is relatively slow, and
- Operation of visual sensing is not as robust as it needs to be for realistic applications.

The control problem has received a lot of attention in the literature (e.g., [Cork96, HuHa96]) but robust vision is just as critical and has received little attention to date [ViHa00].

This work addresses both, robustness, to enable tracking of objects in realistic scenarios, and also control, through fast attentive visual processes. A method for robust edge tracking is presented such that objects can be tracked under varying lighting conditions and in front of a cluttered background. The method is embedded in a model-based tracking framework and experiments in realistic environments demonstrate the robustness of object tracking and successful navigation of walking and wheeled mobile robots. The framework is based on the analysis of the dynamics of visual servoing to maximize the target velocity that can be tracked [Vinc00] and has been named "Vision for Robotics" (V4R) indicating its primary purpose. The specific characteristics of V4R are:

- **Edge Projected Integration of Cues** (EPIC), which integrates image and model knowledge to obtain robust image feature tracking, and
- **Integrated primitives for model-based tracking** containing feature tracking in image windows, a pose estimator and the interface to a CAD model facility.

With cues we denote characteristics that can be found or derived from image data. The gradient cue is used to locate edges and region cues, such as intensity, color, texture, motion, or depth, describe characteristics between edges. The term image feature refers to geometric entities such as line or ellipse, which are found in the image by a fit procedure using the results of EPIC. The image features (referred to as features for short) are projected into the image using the model of the target object and its 3D model features. The main contribution of this work is a robust feature extraction/finding method embedded in a framework for real-time tracking. Section 2 describes the architecture of the framework and the relation to the CAD-system and robot. Section 3 outlines the main components of the framework. Feature tracking using the windowing approach, EPIC and a final validation method are described in more detail in Section

4. Section 5 presents the experiments of tracking ellipse arcs, navigation of a walking robot in a ship section and the navigation of a mobile robot in an office setting. We demonstrate an ability to track real-world objects without any artificial markers that is robust to cluttered background and to varying lighting conditions while the object or camera moves in a very non-smooth fashion. The scheme is open to integrate other cues and future work is described in the final Section 6. The next section will present a discussion of related work.

1.1 Related Work

A reference work of model-based vision is the dynamic modeling approach [DiGr88], which uses local feature processing and model prediction. Models explain dynamic features of cars, which are recognized due to the dark area beneath them. Vehicle steering is successfully demonstrated on German highways.

The geometry of CAD models is utilized in approaches for navigation and object tracking. Mobile robots hold (or build up) a wire-frame map of the building and use landmarks, such as walls or pillars, for navigation (e.g., [CrSt92, KoKa92, KiNe98, EbBa00]). Typically the approaches use the full image and therefore allow a relatively slow robot velocity. The common approach is to project the features of the model (commonly lines) into the image and to match the feature with the nearest gradient edge [Lowe92, Genn92, ToSc97, WuHi97]. Close lines are ambiguous and can be handled by elimination [NaMu00]. Nevertheless, approaches lack robustness in environments with cluttered walls or changing illumination.

The difficulties are obvious from implementations. In a space application edge detection is simplified by dark background and an object consisting of parts with different surface characteristics and dedicated hardware is required to achieve frame rate [WuHi97]. In another

application the extraction of the circles around the dark holes of the pistons of a motor block is robust as long as the holes are dark [JoLa00], and image processing requires dedicated hardware. Another example is visual navigation along a corridor using confidence ellipsoids [KoKa92]. A sequel paper [OhKo98] presents another vision technique to faster traverse the same corridor with uncluttered walls. The work in [EbBa00] realizes fast indexing to obtain feature correspondence. It exploits 2D image and 3D stereo lines, though the authors concede the "reliability-bottleneck" introduced by using one type of cue (gradient) and feature.

While in the above works geometric features such as line are extracted from the image for pose estimation, the approach in [Harr92] uses individual control points along object edges. The control points are searched normal to the edge direction and then used for pose estimation. [ThRe01] improves this work by adding a median filter to detect outliers first for fitting the line and then for fitting the pose. They report the improvements using black and white objects with little clutter in the background. The objective of this paper is to extend these results with an image processing method that enables tracking realistic objects in front of cluttered background using cue integration.

Integration is not a new technique to increase robustness of feature detection. First works date back more than ten years. For example, Aloimonos described methods using optical flow as a means of integration to solve a variety of tasks such as the estimation of target motion and egomotion [AlSh89]. Since motion is already a computationally expensive cue, interaction with other cues was restricted. In a second technique shape was obtained from unifying shading and texture. Another early approach used color and disparity to extend the system capability in stereo matching [HoAh89]. The work in [GaGe89] used color, motion, stereo and texture to label edges. The integration links each cue with the intensity edges using a linear classifier.

Strong edges are emphasized to aid the labeling process.

These early approaches used strong gradients to obtain robust results. Simple cues were used in an adhoc manner to label the type of edge. The goal of today's integration methods is to move towards robust edge extraction of weak signals. While strong gradient responses are robust they are typically in real-world scenes not available. Weak signals are a common limitation of robust image processing. The integration approaches attempt to use several cues to obtain robustness. For example, color and image position are used to track persons [WrAz97] or Bayesian classification is used to integrate color, blobs, and motion in face tracking [Schi00]. One of the most advanced approaches is the integration of motion, brightness and depth for person tracking [ShOk00]. The work in [KrCh00] demonstrates that weighted consensus voting of five cues improves the performance of view-based tracking with a fuzzy fusion method. The problems that have been encountered are the difficulties to obtain fast computing and to handle the different functionality of the cues, e.g., local properties such as edges versus area properties of image regions.

This work also profits from a framework that uses a predefined hierarchy of trackers based on more and more specific cue extraction methods to track the target [ToHa99]. Faces are tracked by first locating face color at low resolution and then following the face accurately with a template tracker. To find door handles, the search first looks for vertical edges and then uses an image template to locate the handle [FeMa98]. The idea is to fall back to lower resolution trackers for recovery if a cue or tracker fails. On the other hand, all cues must be salient to finally locate the object.

2 System Overview

Modern robots demand external sensor information to plan their interaction with the environment. The main goal of the software tool V4R (Vision for Robotics) is the pose determination of objects of the environment with respect to the robot. For object tracking as well as for pose calculation models of the objects are essential and will be used to obtain robust signals from visual processing (see Section 4). Figure 1 shows the inputs and the outputs of V4R.

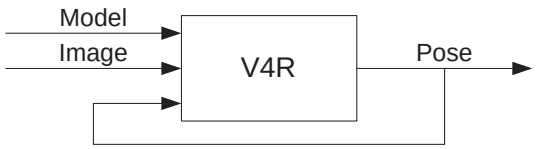


Figure 1: V4R in the user's view.

The inputs to V4R are:

- The **model** expected by V4R is a wire-frame model of the target object, which can be the object to grasp or the environment to navigate in. The object model provides the model features such as line, circle (projected to an ellipse in the image), junction or region.
- The **image** is the intensity/color data captured by a camera.
- The **pose** is the position and orientation of the target object determined in the last tracking cycle of V4R.

The output of V4R is:

- The **new pose** calculated from the new image data using the model and the previous pose estimate. The pose can also be used to locate individual features in the image.

Each input has a different frequency. The wire-frame model is sent once during the startup of V4R or when beginning a new task with a new object. From the model the visible feature

in the image are calculated depending on the camera pose. The visibility information of the model is updated depending on the view change (e.g., when the robot moves a larger distance in the environment) at about a rate of seconds. In contrast, the images are acquired at frame rate and the pose is calculated at the same rate and used as feedback signal.

V4R can be used in combination with a robot in different ways, depending on the placement of the camera. It is possible to mount the camera at a fixed position and detect both the robot and the work piece in one image (end-point closed loop [HuHa96]). Another option is to mount the camera near the gripper of a robot or to fix the camera on a camera head that is mounted itself on a mobile robot and to calculate the relation between the object and the camera (end-point open loop [HuHa96]). In the latter two approaches the pose of the object with respect to the robot can be calculated by transforming the calculated object pose using the pose of the camera relative to the robot. In each case V4R communicates with the components as depicted in Figure 2.

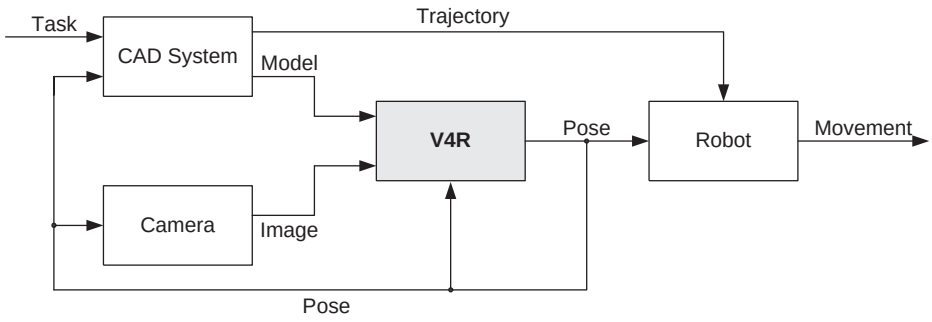


Figure 2: The principal components communicating with V4R.

The main characteristics of the components, briefly, are:

- A **CAD system** provides two functions. First, it is used by the user to define the target robot trajectory for the given task in terms of a sequence of robot poses. And second, it provides the model of the target object, which is projected into the image to obtain

the visible image features depending on the camera pose. The CAD system presents the model in the form of a wire-frame model to V4R. The wire-frame model comprises 3D coordinates of vertices, describes the topology of vertices, edges and regions, specifies the geometry of the model features, and gives the attributes of features. The CAD system computes the visible features for a given camera pose. V4R is able to determine the visibility of feature as part of the tracking procedure only for small pose changes and convex objects. The CAD system will resend the actually visible features in the image, if the robot moves over longer distances and the visibility changes significantly. The objects are considered to be opaque.

- The **camera** is responsible for image acquisition and is characterized by a set of intrinsic and extrinsic camera parameters. An optional camera head can be used to fixate the target object in the image and to increase the search area beyond the fixed view angle of the camera.

- The **robot** is the user of V4R. It takes the pose estimates and the target trajectory and performs the required motion. V4R can be also used to track objects with an active camera head or a pan/tilt unit. In this case the goal is usually to keep the target object in the center of the image using the target's image position determined by V4R.

Optionally a second vision system can be a component which is able to deliver additional image feature data. This auxiliary information can be used for Pose Calculation in V4R. Section 5.2 presents a system where a stereo vision system communicates with V4R. The experiments in Sections 5.2 and 5.3 will present two examples of V4R being used to deliver pose as indicated in Figure 2.

3 V4R Components

The goal of this section is to outline the functionalities of V4R. The main task of V4R is model-based visual feature tracking to estimate the 3D-pose of an object relative to a reference coordinate system. V4R can presently handle geometric features such as lines, junctions, ellipse arcs, and regions, which correspond to a wire-frame model coming from a CAD system.

The primary goal of the visual processing is to provide robust features in real-time. The secondary goal is to deliver the feedback result as fast as possible, because the *latency of the feedback signal is the main factor to influence the dynamic performance*, i.e., the maximal tracking velocity that can be reached [Vinc00] (see Section 4.1.2 for details). This analysis presents a fundamental result for visual servoing tasks and applies to a large range of applications: active vision systems as well as servoing robots using fixed or robot-mounted cameras.

The real-time criterion can only be met by processing a small subset of the total image data. The consequence is to use a windowing method similar to the approach in [HaTo98]. Figure 3 illustrates the functional partitioning of V4R. The following paragraphs outline the main functions:

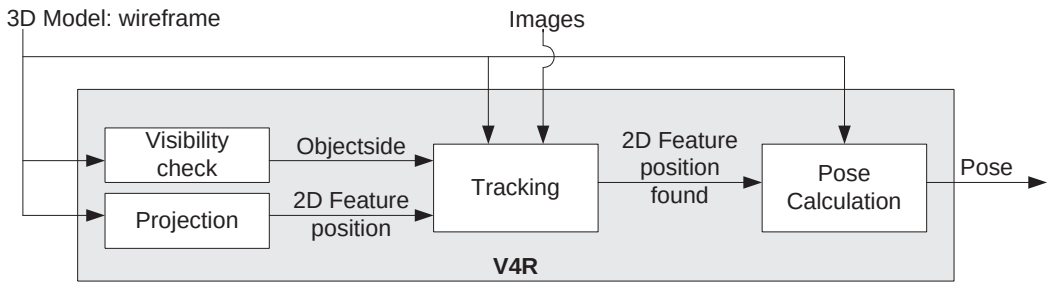


Figure 3: The main functions for object tracking and pose estimation in V4R. *Objectside* $\in$ {left, right, both} denotes which side of an edge is the object.

Visibility Check Before the model features are searched for in the image at each tracking

step a visibility check of these features has to be performed according to the current pose estimation. This ensures that only those features are searched for which are estimated to be visible in the image. For robust edge detection it is also useful to know if an edge is part of the boundary of the object (i.e., an edge between object and background) or an edge within the object. Since the background is changing unpredictably just those sides of an edge (with respect to the edge direction), which are part of the object, are of interest for further examination. This information is extracted when examining the visibility of surfaces and it is then sent to the tracking component (see Fig.4).

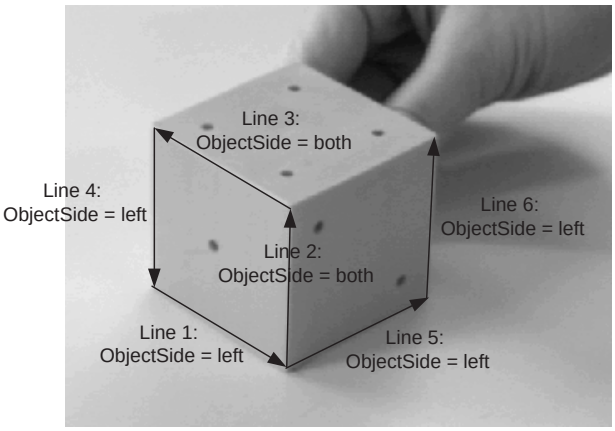


Figure 4: The value of the *ObjectSide* $\in$ {left, right, both} for some lines derived from the model and the camera pose.

Projection The search process is initialized by projecting the model into the image to provide an approximate position of the image features. At this location a tracking window is placed in which the actual edge is searched for by the tracking function.

Tracking The goal of the tracking component is to robustly extract the 2D-position of the features in the image and is described in detail in Section 4.

Pose Calculation The pose calculation function in V4R computes the pose of the camera

with respect to the modelled object. This pose is then transformed into the pose of the robot either relative to the environment (for navigation) or relative to the target object (for object grasping). This is done by fitting the image features extracted by the tracking component to the corresponding model feature received from the CAD system. The algorithm employed is based on a method proposed by Wunsch [WuHi97] and was modified to detect outliers (similar to [ThRe01] using the residuals of each feature [Kr97]. The Wunsch algorithm was chosen because it is fast and therefore well-suited for object tracking. It can handle 2D lines, junctions, and circles (which appear as ellipses in the image) and 3D junctions (which are utilized in the experiment in Section 5.2). Pose estimation finally reports a least squares fit for the object pose. Outlier detection is most effective in case of redundant features, which can be expected for non trivial objects. Examples are given in the Section 5.

4 Robust Feature Tracking

This section will outline the approach proposed to improve the robustness of feature tracking. The main idea is (1) to integrate several cues in EPIC and (2) to use model knowledge to disambiguate multiple feature candidates in a validation step. These two functions are integrated into a scheme to track objects described by a wire-frame model. The principal operation of the tracking function of Fig. 3 is summarized as follows (see Fig. 5):

1. *Set* the tracking window for each feature of the wire-frame model *and warp the image window* along the main feature extension (line length or ellipse circumference),

2. Extract relevant edgels using *EPIC*,

3. *Fit the feature geometry* to the edgels using a probabilistic RANSAC [FiBo81] scheme and obtain feature candidates, and

4. *Validate feature candidates* using the topology of the features in the model.

The next four subsections describe these steps in greater detail. The emphasis of this paper is to describe the improvement of step 2 towards threshold-free adaptation to changing conditions (Section 4.2) and the enhancement of robustness when adding step 4 (Section 4.4). The other steps will be described briefly to provide a complete overview of the approach.

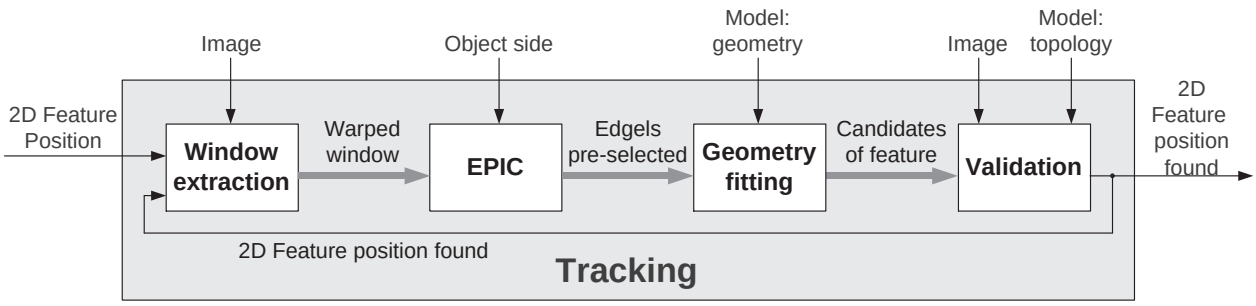


Figure 5: The functions to obtain robust edge tracking of one feature. *ObjectSide* denotes which side of an edge is the object.

4.1 Window Extraction

The window extraction consists of two steps: (1) The definition of position and size of the window, where the feature is expected and (2) the acquisition of the image data. The following subsections describe the procedure.

4.1.1 Placing the Window

Three cases are distinguished for placing the tracking windows in the image. The first case is the initialization. It can be executed either manually by placing the window in the image or

it can be performed using a projection of the model features of the wire frame model into the image (indicated in Figure 3). The projection uses a pose estimate, which is either given by the CAD system or provided by the user. The experiments in the RobVision project used a second vision component to fulfil this task (Section 5.2).

The second case is regular tracking. Once the feature has been found, in the next tracking cycle the window is placed at the same location (assuming a zero velocity image feature motion model) or at the predicted feature location (e.g., using a Kalman filter for prediction as in [DiGr88]).

The third case is the response to the detection of mistracking. This can be a result of not finding the feature in the tracking window or when the feature has been recognized as an outlier in pose estimation. If only a few features are mistracked and sufficient features are found for pose estimation, the window of the mistracked feature(s) is placed in the image using a re-projection. This enables recovery from mistracking.

4.1.2 Window Warping

A design principle of the V4R Framework is the window-based tracking approach. This approach is recommended from studying the dynamics of visual tracking [ViWe97] and visual servoing [Vinc00]. The studies indicate that highest target velocity of tracking is achieved when processing time is equal to image acquisition time. Hence, the window size must be set such that the windows can be processed within 40 ms, corresponding to the frame rate of standard video cameras.

The consequence of the dynamics analysis requires us to set the window size to obtain the small calculation time needed. If several windows are used, the sum of the time to process all

windows must be smaller than the frame time [ViWe97].

Tracking searches for the desired feature by processing all pixels in the image window. Therefore the processing time is proportional to the number of pixels in the image. For line tracking V4R realizes on a PC with 350 MHz a time of about 0.003 ms per pixel. Therefore one window of size 40×40 can be tracked in about 5 ms, which includes the time for all tracking steps within V4R.

Line features are warped within windows resulting in the line being roughly vertical within the tracking window. Edge finding can now operate along each horizontal row, a technique inspired by XVision [HaTo98]. While the localization perpendicular to the edge direction is determined in the tracking window, the localization along the edge direction can only be achieved by intersecting the edges at junctions (refer to Section 4.4.1).

In the experiments we used a width of 40 pixels and a height of nominally 40 pixels to achieve the calculation time requirement. If lines are longer, the window is sub-sampled in the vertical direction (along the extension of the line) to keep processing time constant (see Figure 6). This sub-sampling is executed automatically using powers of 2, such that line length in the warped window varies between 20 and 40 pixels. This effect is indicated by the black area at the bottom of the warped windows in the Figure.

Ellipse features are tracked by placing tracker lines normal to the circumference of the contour. Tracker lines are tracking windows of height one. The edge is vertical in the warped image row vector. Figure 7 shows an example of placing tracker lines for ellipse tracking. The processing time depends on the number and width of the tracker lines. The method was tested on a Pentium 350MHz PC. Tracking of one ellipse or ellipse segment with 30 tracker lines of 60 pixels width and with automatic handling of occlusion (see Experiments in 5.1) takes about

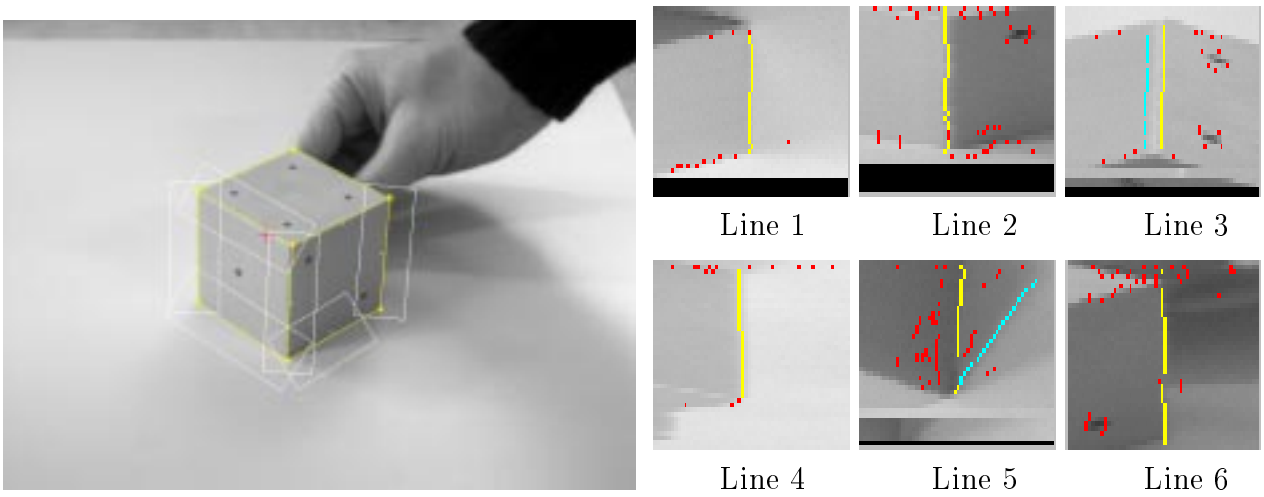


Figure 6: Placing the tracking windows for lines. The small images to the right show the warped windows (Line 1 to 6 of Fig. 4 from top left to bottom right). Windows for long lines are sub-sampled in the line direction.

30 ms.

Figure 7: Placing the tracking windows for an ellipse: 30 tracker lines (the dark lines in the image) are placed normal to the circumference of the ellipse. A tracker line is a row vector with the vertical edgel representing a warped window of height one. The width of the windows can be set and in the example is 60 pixels.

4.2 EPIC

The aim of EPIC is to provide robust edgel matching by using region cues such as intensity, color or texture in the regions adjacent to the edge. By using a rich variety of cues and integrating them we achieve greater robustness than using classical edge magnitude cues alone.

The integration uses the model knowledge to select which side of the edge belongs to the object and which side is background. EPIC attempts to find (track) edgels that exhibit the most similar edge cue values as the last tracking step at the object side indicated by the model. As long as object and background differ in intensity (or color), the edgels describing the target object are found and can be distinguished from background.

The rationale of using EPIC to match edgels is to reduce the number of edgels found in the window to the "good" edgels, that is, the edgels that indicate with high likelihood the same feature tracked in the last cycle. The result is a more robust fit of the feature geometry and an increase in effectiveness leading to faster processing (refer to Section 4.3).

The EPIC procedure is as follows: the edge cue from the last tracking step is stored as mean and standard deviation values. New edgels are found in each row of the tracking window (see Fig. 8). For each interval between the edgels a new interval cue value is calculated. The interval cue value c_k is calculated from the maximum value of the histogram of the cue within the interval k. Experiments with median values gave results of similar robustness, however, the computations needed to calculate the median increase more than linearly with the number of pixels.

The interval cue value $c_{k,i}$ is calculated for each cue, with $i = 1, ..,$ number of cues. The interval cue values found in the present tracking cycle $c_{k,i}$ are then used to find the weighted

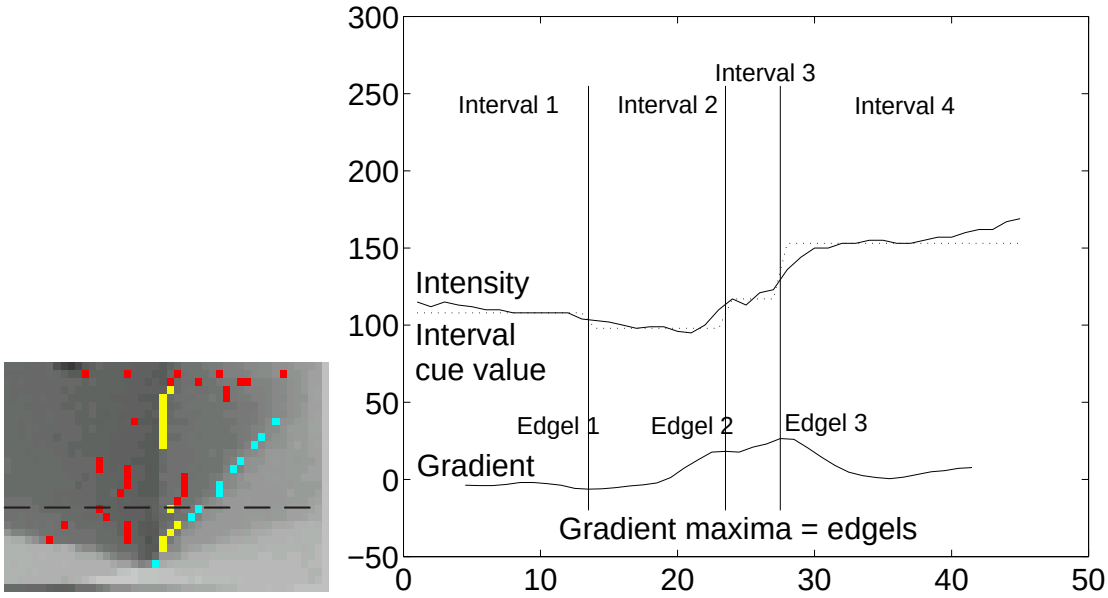


Figure 8: Example of determining the interval cue values for the intensity cue in the dashed row of the tracking window (left) of Line 5 in Fig. 6. The gradient maxima determine the positions of the k edgels. Between edgels a histogram technique is used to find the $k+1$ interval cue values.

cue values $C_{k,i}$ as follows

$$C_{k,i} \quad = \quad \exp\left(-\frac{(\mu_i - c_{k,i})^2}{2\sigma_i}\right) \tag{1}$$

where the reference edge cue values from the last tracking cycle have been stored as the mean value μ_i and the standard deviation value σ_i for the cues i.

Finally, for all edgels in a row (see Fig. 8) and for all rows in the warped window the likelihood l_k that the edgel k is a "good" edgel is evaluated to

$$l_k = \frac{1}{W}\sum_{i=1}^{n} w_{left}C_{k,i} + w_{right}C_{k+1,i} \tag{2}$$

with

$$W = \sum_{i=1}^{n} w_{left} + w_{right} \tag{3}$$

The index $left$ and $right$ of the above weights w refers to the two possible sides of an edgel (refer to Fig.4). Knowledge of the model is used to select only the side that belongs to the target object while the background is not regarded. If the model indicates an object, the respective weight $w_{left/right}$ is 1, otherwise it is 0. If the edge is a contour edge, the object can be only on one side and the respective weight is set to one. In case an edge is the edge between two visible surfaces of the object, then both weights are set to one.

As a result Eq. (2) calculates the likelihood of an edgel as the sum of the Gaussian weighted cue values $C_{k,i}$ depending on the *ObjectSide* given by the model. The advantage of this scheme is that each value $C_{k,i}$ is calculated using the Gaussian distance measure of Eq. (1), which eliminates the need to set threshold parameters.

Using the result of feature detection, new values μ_i and σ_i are calculated from the $c_{k,i}$ values of all the good edgels that voted for the finally selected edge (please refer to the next two Sections for details). As a result EPIC can adapt to varying reflections along the edge and to slowly varying lighting changes in the environment (see Experiments, Section 5).

Other cues than intensity, color, and texture have been tested [Zele98, ViBe01], for example, the maximum gradient, the type of transition at the edge location, and the distance to the previous edge location. However these cues did not improve robustness. In particular the magnitude of the gradient provides no information about an edgel belonging to the object edge and is not robust to background changes.

The EPIC scheme is very effective because the edge features contain attributes that give indications to select the cues (intensity, color, texture, ...) of the object. Based on the localization of edgels, the cues can be easily integrated and the list of cues given above can be easily increased with other cues. The principal idea is to use these cues to limit the selection of

edgels. For example, incorporating color added robustness in distinguishing the correct edges from shadows and highlights [Zele98]. This limitation to good edgels renders the next step of fitting the feature geometry to the edgels very efficient.

4.3 Fitting the Feature Geometry

In this step, the good edgels are used as input to a RANSAC voting scheme [FiBo81] to fit the edgels to the feature geometry. The RANSAC idea is to randomly select a subset of sufficient size to estimate feature pose. The edge is re-projected into the window and the votes of all edgels sufficiently close to the projected edge are counted.

In this work line and ellipse features are considered. For a line all good edgels in the warped window are used. For the ellipse all good edgels found in all tracker lines are used. RANSAC selects the features with the highest number of votes. Often several feature candidates have similar high values (for example, Fig.9). In this case several feature candidates are retained and the final selection will be performed in the validation step (Section 4.4). When the feature is finally selected unambiguously, the new feature pose and the new edge cue values (μ and σ) for all cues are calculated for this feature and stored for use in EPIC in the next cycle.

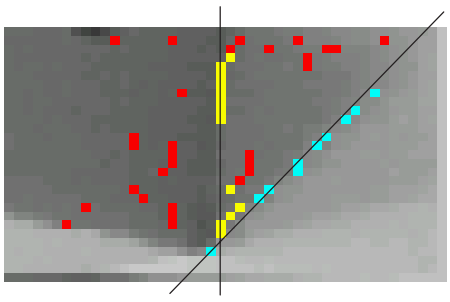


Figure 9: The two line candidates found by fitting lines to the edgels selected with EPIC. The line feature is Line 5 in Fig. 4.

The rationale of combining EPIC with the RANSAC scheme (or using a least median square

line fit with a limited number of samples [1, ThRe01]) can be demonstrated when investigating the likelihood of finding the correct feature. A good feature is found if n good edgels are found to support the feature geometry, where $n = 2$ for a line and $n = 5$ for an ellipse. If the likelihood that an edgel is "good" is given by g, a good feature is found with likelihood g^n. Repeating the random selection of n edgels to define an edge k times gives the likelihood of finding the correct edge l to

$$l = 1 - (1 - g^n)^k. \tag{4}$$

This relationship depends strongly on g. Due to the higher power (g^5) this effect is more pronounced for ellipses than for lines. Therefore limiting the number of "good" edgels using EPIC is extremely effective to reduce k (exponentially for the same l). The result is the capability to find lines and ellipse arcs more robustly and at real-time (calculation times have been given in Section 4.1.2). In summary, the overall computations to achieve a given likelihood in Eq. 4 are highly reduced when using EPIC to extract "good" edgels. The additional computations required for EPIC need only a fraction of the savings.

4.4 Validation

EPIC and fitting feature geometry to the edgels diminishes the number of possible feature candidates significantly. Nevertheless, in many cases several feature candidates for one model feature are found. Reasons can include a shadow, a cut or scratch on the object surface, a chamfered edge or a similar background. In these cases a more restrictive edgel selection method makes no sense since correct edgels would also be eliminated. The strategy is to express this ambiguity with several potential candidate features, for example the two line candidates of the warped window in Fig. 9.

In this section a simple approach is described to use model knowledge for disambiguating the multiple feature candidates. This ambiguity can not be resolved locally by regarding solely the feature itself without any additional information. It is necessary to take more global knowledge of the object into account to decide, which of the feature candidates is correct.

The idea is to check the plausibility of each of the feature candidates by using topological relationships to other features (for example, lines and junctions) given by the object model. This step is referred to as *validation*. Validation exploits two simple topological relationships between edge features:

- Connectivity, for example, several lines intersecting at one junction, and
- Parallelism, for example, two parallel line features in the model.

The next two subsections will shortly outline how simple procedures can be implemented to select the most likely configuration of feature candidates.

The use of topological information is related to work in computer vision using the Gestalt ideas (for example, [DiGr88, BoWe89, Lowe92, LeMe99]). The edge features are grouped using the Gestalt properties and then these groupings are used for tracking or object recognition. This bottom-up approach suffers from the difficulty to limit the combinatorial explosion of the calculations. The approach taken here purposively chooses Gestalt criteria from the model of the target object to disambiguate the data coming from feature detection. The computations are limited to a few validation steps, which are processed in a fraction of the time needed for EPIC and fitting of the geometry.

4.4.1 Connectivity

The connectivity between edges is given through junctions which are extracted from the wire-frame model. The model knowledge specifies the edges, which intersect at one junction. The intersection delivers the new 2D-position for this junction without tracking the junction itself.

The validation procedure for connectivity at junctions is as follows. Figure 10 gives an example, where connectedness is used to disambiguate the two feature candidates of Line 5 in Fig. 4. It is tested if the end of the edges actually extends to the calculated 2D junction position. This is done by examining if at least one edgel lies on the edge next to the newly calculated junction position. This verification step is applied to all combinations of edge candidates. The one combination of candidates that receives the best evaluation wins. A junction votes for the respective edge candidates.

The validation steps to evaluate correct connectedness at one junction can therefore be summarized as follows:

1. Select one candidate of each edge connected to a junction.
2. The selected edge candidates are intersected and the 2D intersection point is calculated.
3. The endpoints of the edge candidates are examined: A window is warped perpendicular to the edge a few pixels away from the calculated 2D intersection point and edgel finding is performed.
4. Steps 1-3 are repeated until all combinations of edge candidates are tested.
5. The best set of edge candidates is voted for.

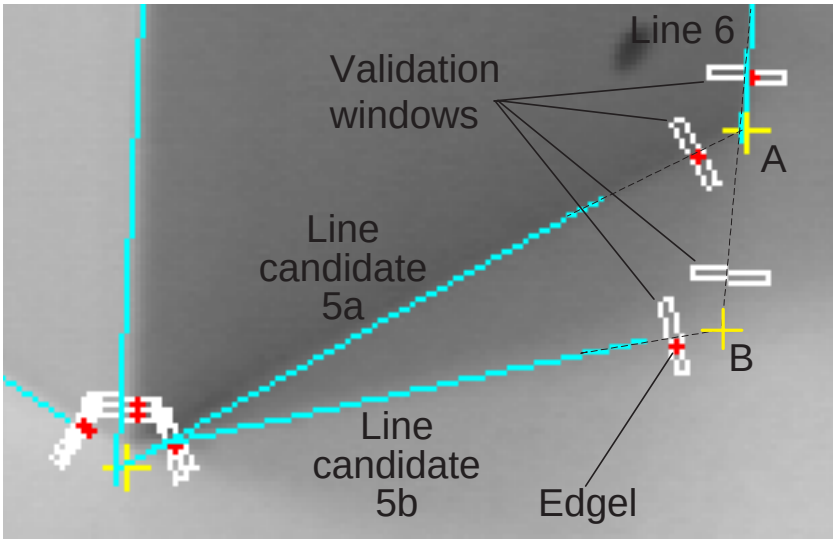


Figure 10: Validation using the connectedness criteria of lines at junctions. The small rectangular validation windows (displayed in white) indicate the image areas where the continuation of the edge towards the junction is calculated. The dark crosses within these windows show the edgels found in these validation windows. The bottom Line 5 (refer to Fig. 4) has two candidates, 5a and 5b. Therefore there are two potential intersections with Line 6, junction A and B. Junction B with the lower horizontal line candidate cannot be verified (there is no edgel in the validation window above B) and therefore the feature candidate 5b is rejected. Fig. 9 shows the warped image of Line 5 with the two candidate lines.

4.4.2 Parallelism

The constraint of parallel edge features in the model is exploited by a similar procedure. Of particular interest are line features. From the model of the object a pair of parallel lines is extracted, which projects to nearly parallel lines in the image. The validation of parallel edge candidates in the image uses two criteria in relation to the expected values: (1) the difference in the orientations of two candidates and (2) the normal distance from a line center to the other

line. The expected angle and distance are extracted from the projection of the edge features in the image.

The validation of parallel edges is of particular importance, if the edges appear close to each other in the image. For example, two parallel lines close to each other will appear in the tracking windows of both features (see Fig. 11). Disambiguation in each individual tracking window is difficult at this level. One solution could be to eliminate one of the edges from the model. But this solution leads to decreased accuracy in pose calculation since the redundancy of features is diminished and the danger of "jumping features" (since there are two similar edges in the tracking window it can happen that tracking switches between the two possibilities) is increased. Hence a validation of parallel edges is a very effective means to avoid such difficulties. The steps to validate parallel edge candidates are summarized below.

1. A pair of parallel edge candidates is selected from the model.

2. The parallelism constraint of edge candidates is examined by comparing the difference between the angles of the candidates to the expected values. The normal distance between the parallel edges is projected into the image and compared to the distance measured in the image. Fig. 11 gives an example of three parallel lines, which are disambiguated using the parallelism constraint three times.

3. Steps 1-2 are repeated until all pairs of parallel edge candidates are tested.

4. Vote for the pair of candidates that best fulfills the constraint.

Very often one edge has several relationships (connectedness and parallelism) to other features that are validated. For example, one line has two junctions (one at each end) connected to other edges and can at the same time be parallel to another line. All these relationships are

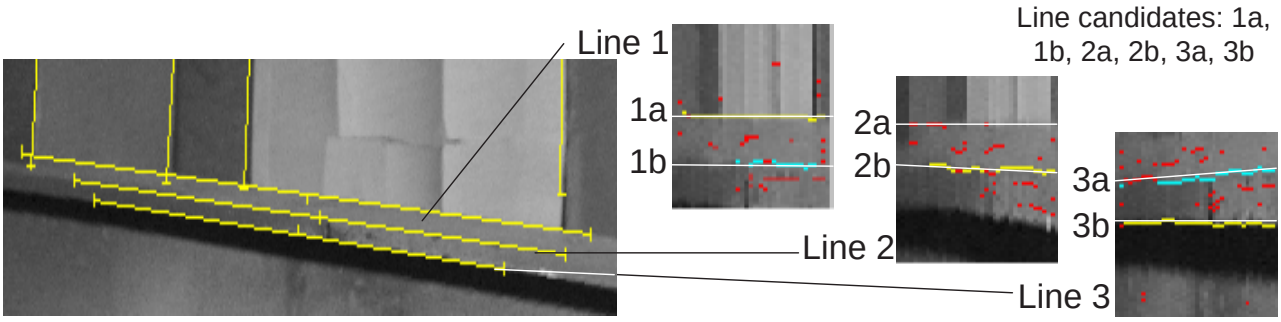


Figure 11: Validating parallel line features. The left image shows an example of three parallel lines as part of the experiment in Fig. 16. Three parallelism constraints are established between Line 1 and Line 2, between Line 2 and Line 3, and between Line 1 and Line 3. Validation successfully disambiguates the line candidates given in the small tracking windows to the right. Line 2 in the image is found as candidate 1b, 2b, and 3a in all images, but finally it is only matched correctly to Line 2. The tracking windows have been turned 90 degrees to correspond to the line orientation in the original image.

evaluated and votes for the involved edge candidates are determined. To handle these votes coming from different constraints, a simple majority voting scheme is established, which leads to the selection of the topologically best fitting edge candidates. The experiment in Section 5.2 will show the value of validation in a realistic experiment.

5 Experiments

The capability of V4R and the EPIC and validation methods are shown in three experiments. First, the capability of EPIC is highlighted in ellipse tracking. The use of EPIC is the main factor to enable tracking of real-world objects over cluttered background and with lighting and reflection changes. Second, EPIC and validation are used for navigating a walking robot into a

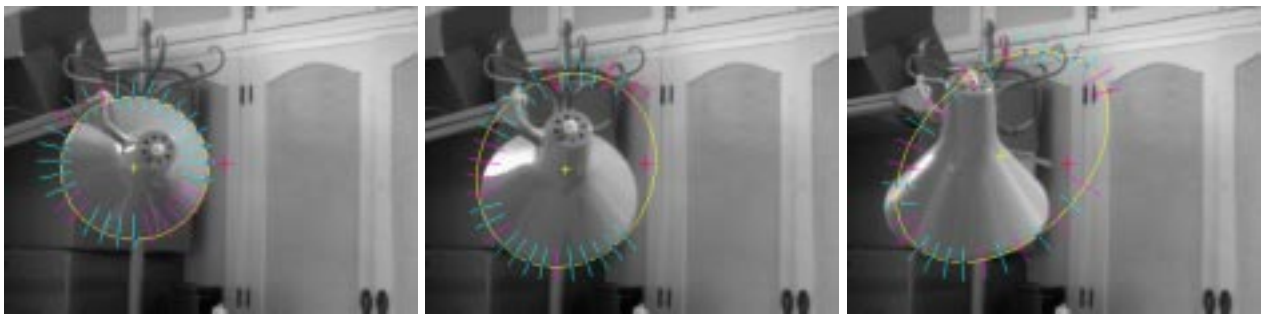


Figure 12: Tracking a lamp shade using only gradient information.

ship section and, finally, to navigate a mobile robot in an indoor environment. The validation step adds additional robustness and uses features such as parallel lines as significant object indicators instead of eliminating double lines [NaMu00].

The following section presents sample images of the tracking sequences. The multimedia Extensions show results as video sequences which are color coded to show intermediate tracking results.

5.1 Tracking Ellipses

The experiments are conducted using the intensity component of color images processed at frame rate. To demonstrate the capability of the EPIC approach, a lamp shade is tracked using only gradient information and then using EPIC to find the edgels for ellipse fitting (Extension 1). Fig. 12 shows examples from this sequence using only gradient information. The background clutter soon produces strong edges that distract the line trackers along the circumference and tracking fails.

Fig. 13 and Extension 2 demonstrate the tracking of the same lamp shade using EPIC to find the edgels on the tracker lines. During the sequence the slim head of the lamp occludes part of the rim of the lamp shade. The occluded segment is not tracked. If the ellipse tracker

detects several neighboring tracker lines that cannot find the ellipse detected by the other line trackers, the ellipse is broken up. The tracker lines are equally distributed along the remaining circumference of the ellipse. At each end of the ellipse segment one tracker line is added to detect the recovery from the occlusion [BiVi01]. Ellipse arcs with a subtended angle less than 180 degrees are difficult to track, since ellipse fitting becomes unstable [FiPi99].

Fig. 14 and Extension 3 show the tracking of an ellipse segment over cluttered background and with significant changes of the reflectivity of the tracked object, which are both handled by EPIC.

The ellipse tracker is an integral part of the model-based tracking tool V4R. Using the ellipse information in the image, the 3D center point of the model circle and the surface normal of the ellipse plane are reconstructed. This information is then used to estimate the pose of the

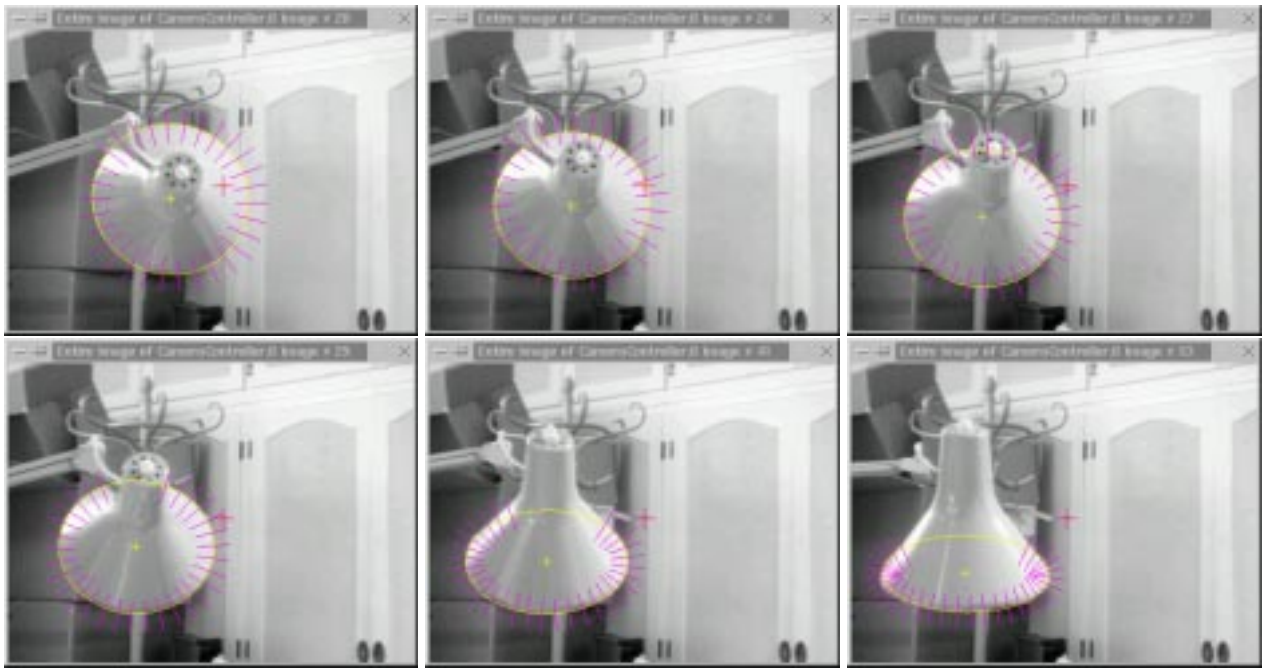


Figure 13: Tracking of the lamp shade using EPIC. The tracker lines are placed along the visible contour detected in the previous tracking cycle. At both ends one tracker line is placed beyond the end to enable adaptation to changes in occlusion or orientation of the ellipse arc.

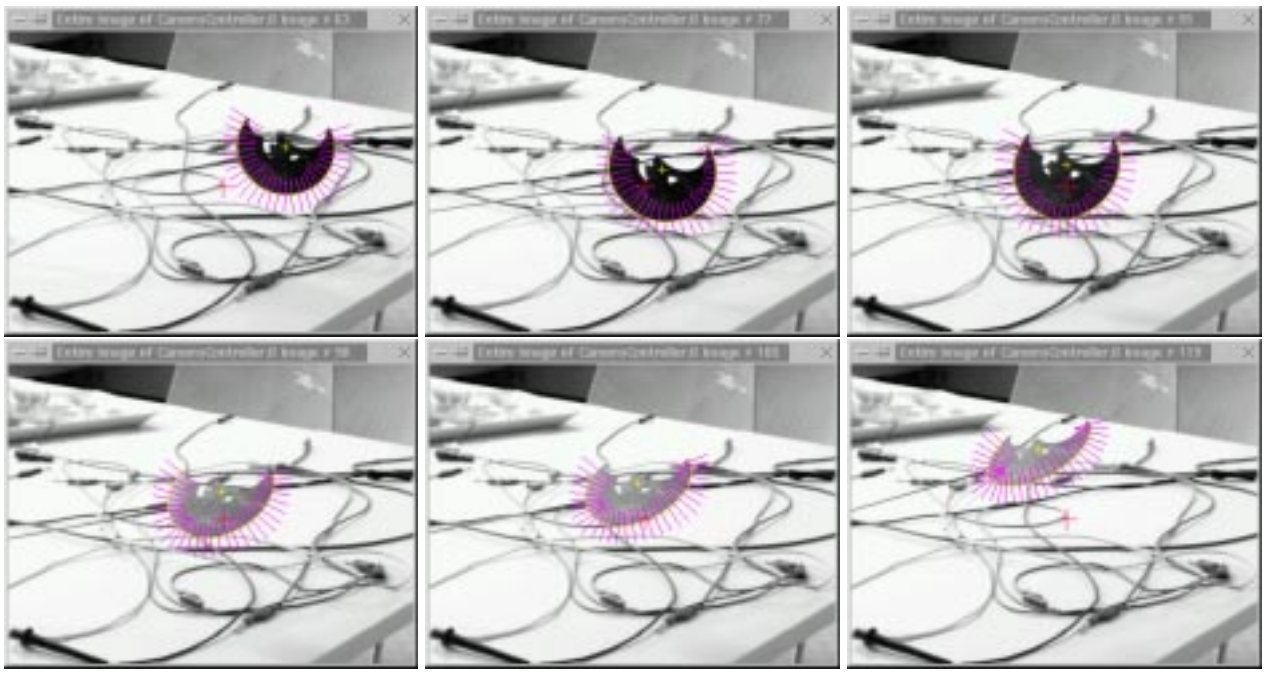


Figure 14: Samples of tracking an ellipse arc over cluttered background and with substantial changes of reflectivity.

object together with the information of line and junction features (see Extension 4). Figure 15 gives samples from the sequence of tracking a monitor base. The estimated pose is re-projected into the image for better evaluation of the result obtained.

5.2 RobVision - Navigating a Walking Robot into a Ship Body

As part of the Esprit project RobVision (**ROB**ust **VI**sion for **S**ensing in **I**ndustrial **O**perations and **N**eeds) a system has been developed to navigate a walking robot through a ship section for inspection and welding tasks. In this project robust behavior has been improved by integrating the features of two different vision methods, one measuring 3D junctions with the stereo head (Pronto), the second tracking edge and junction features in a single image (V4R). Pronto was developed by the Lab for Integrated Advanced Robotics of the University of Genoa. The real-time capability is important to reach an acceptable performance of the overall system. In this

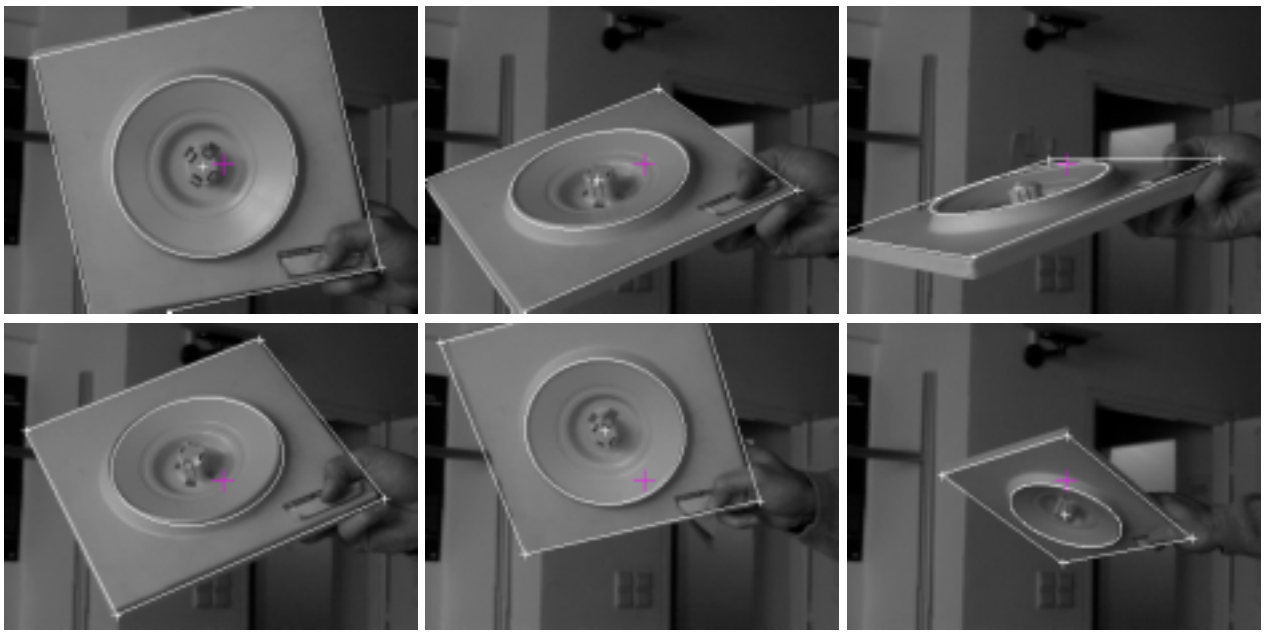


Figure 15: Tracking the base of a monitor using ellipse and lines. The pose estimated from the lines and the ellipse is re-projected into the image.

project both vision systems need to work on the same image. This is solved by transmitting the image from Pronto to V4R. With the additional communication overhead to exchange the features and estimated poses, a pose update cycle time of 120ms is achieved. The communication strategy followed Figure 2 with the additional vision system Pronto working in parallel to V4R, where V4R finally estimates the pose using the image features found by both vision systems.

As seen in Figure 16 a mock-up of a ship section — a typical test environment — was built by Odense Steel Shipyard Ltd. (OSS), the end user of this project, with structures that can be found in a big ship environment. Several trajectories were examined to test the robustness of the system and its accuracy. The mock-up is made of large metal plates welded together. Figure 17 shows an example of a camera view. The CAD system developed by the Department of Production of the Aalborg University (AAU) provides the features seen in Figure 17(a) and

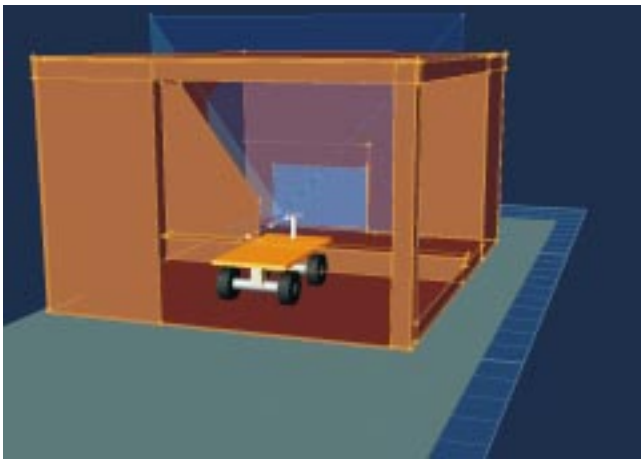
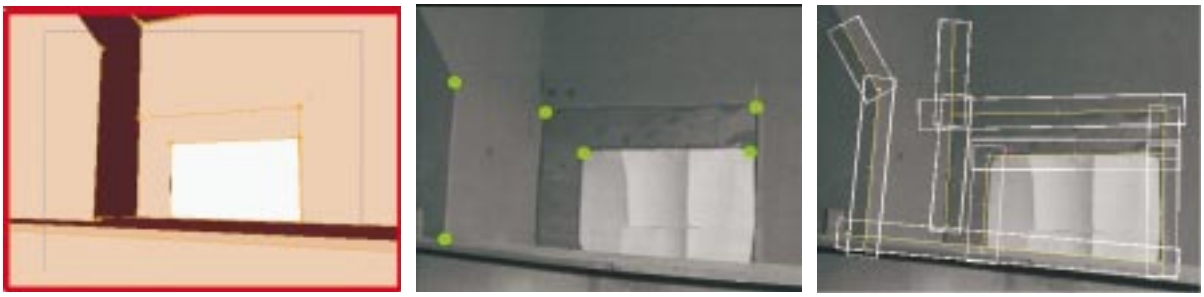


Figure 16: Model of the mockup and the trolley with the stereo head looking at one view point.



(a) (b) (c)

Figure 17: (a) View with features generated by the CAD subsystem, (b) 3D junctions detected by the stereo vision subsystem, (c) lines in their searching windows tracked by V4R.

delivers them to the vision systems Pronto and V4R. The Pronto stereo vision system is able to extract the 3D coordinates of the junctions (Figure 17(b)) and V4R extracts the 2D position of edges and junctions in the image (Figure 17(c)).

The difficulty of the ship environment is that some welding causes irregular edge features, contrast is generally poor and additional features on the plate surfaces can adversely affect feature finding and pose calculation. Within RobVision the capacity of two vision systems (Pronto and V4R) is used to extract redundant features from the images. All features extracted by these two vision systems are used to estimate the robot pose in V4R. See [ViBe01] for more details about this project.

Extension 5 shows the tracking performance from the walking robot. Fig. 18 shows samples of the images captured along a path inside the mockup. The robot starts in front of the T-truss and moves about 1 meter ahead. To visualize the final tracking result, the model is re-projected into the image displaying the new calculated pose.

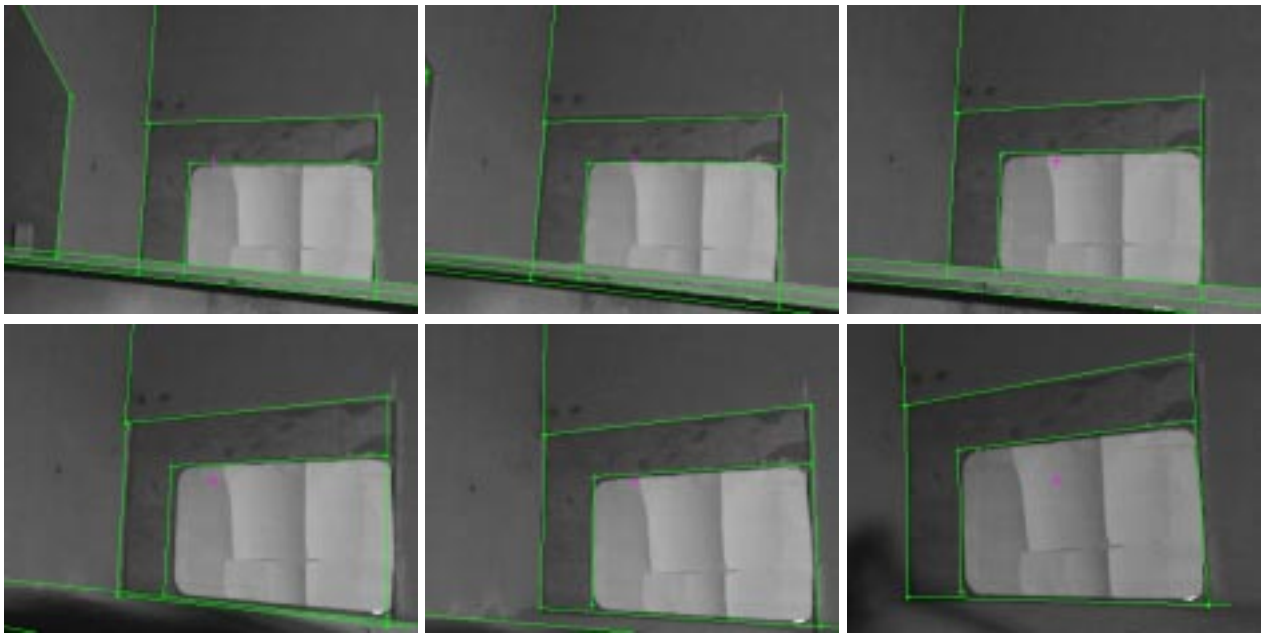


Figure 18: Tracking the mockup from a walking robot using EPIC and validation. The three parallel lines in the bottom front can be easily distinguished.

The value of validating lines is seen by comparing the tracking performance in Fig. 18 shows samples of the images captured along a path inside the mockup. The robot starts in front of the T-truss and moves about 1 meter ahead. To visualize the final tracking result, the model is re-projected into the image displaying the new calculated pose. with Fig. 19. While the parallel lines in the bottom front are nicely disambiguated in Fig. 18, tracking is confused and lines are placed on top of each other in Fig. 19.

Fig. 20 gives the pose recorded along the path of the sequence in Extension 5 (Fig. 18). It can be observed that the pneumatic walking robot produces many jerks. Especially fast changes in

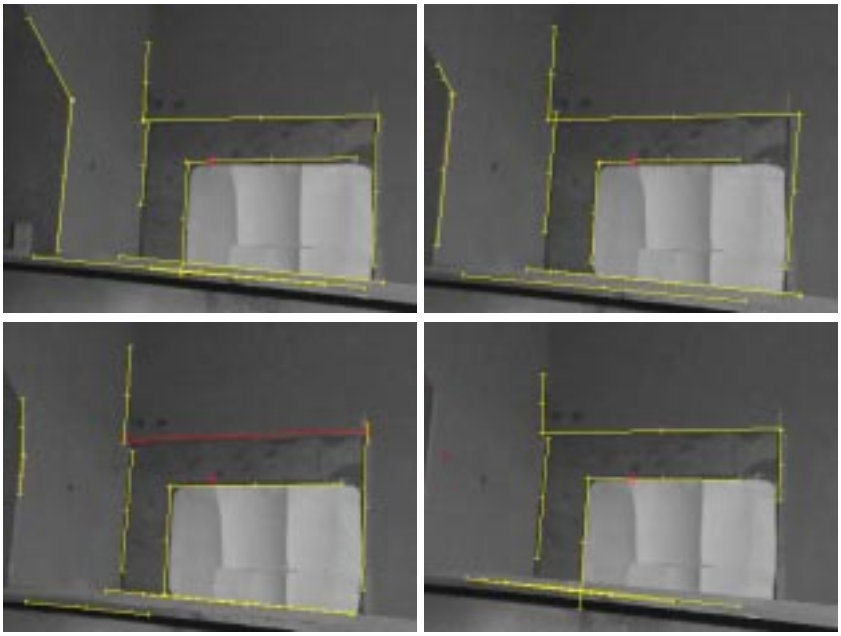
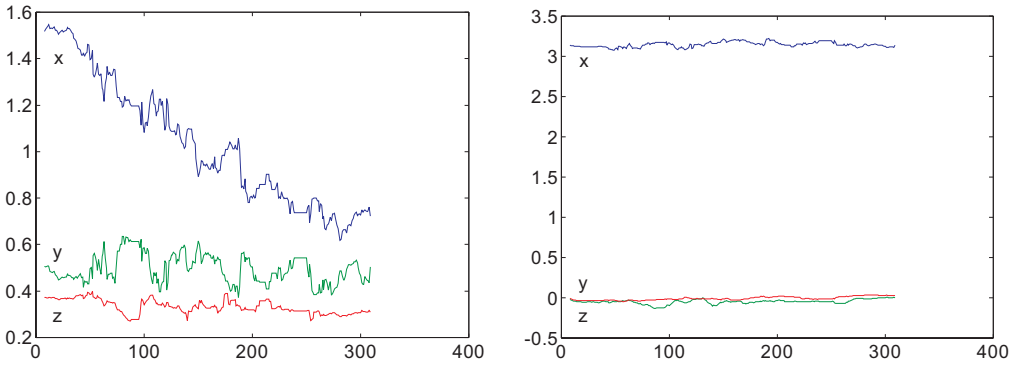


Figure 19: Tracking the mockup from a walking robot using EPIC without validation. The three parallel lines in the bottom front appear as only one or two tracked lines and correspondence changes between frames.

the orientation of the robot cause big deviations of the feature positions in the image. Because of the spatially restricted window size, these rapid deviations cannot be tracked and the features are lost. Larger windows would need longer calculation time and therefore not eliminate this problem [ViWe97]. This loss of features reduces the reliability of the system. Therefore a head stabilization has been implemented, which detects the body motion using accelerometers and which can be taken into account when placing the windows, and the robustness of feature finding is regained.

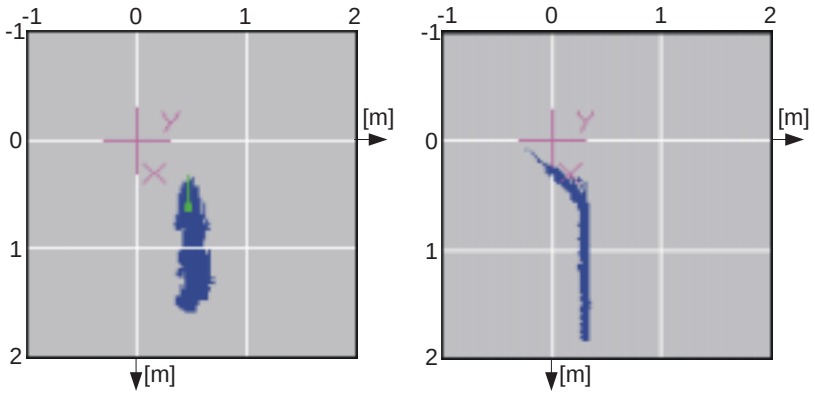
In Figure 21 the motion of a trolley is compared to the motion of the robot. The smoother motion of the trolley results in a smoother robot trajectory.

The tests executed show that the system benefits from the redundancies implemented. As long as sufficient features can be tracked, the system is able to re-find lost features. Additionally,



(a) (b)

Figure 20: Robot pose vs. the frame number along a trajectory: (a) 3D position in [m] and (b) 3D orientation using RPY angles in [rad].



(a) (b)

Figure 21: Projection of (a) walking robot poses (b) trolley poses along a trajectory into the x/y plane.

incorrectly detected features are filtered out in the pose calculation step. An example of tracking sparse line features (five instead of the minimally required three line features) gives Figure 22 and Extension 6. Tracking is not as robust as compared to using many redundant features. The stabilization of the pose is further increased by using 3D features and by the continuous update of new image features from the CAD system, when moving over longer trajectories. The case of sparse features also profited from using accelerometers to stabilize the camera.

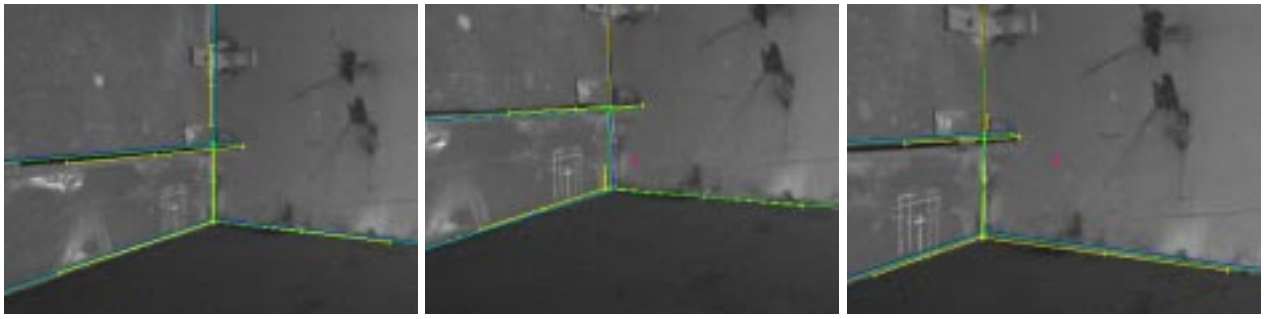


Figure 22: Examples of tracking from a walking robot in a part of the mockup with only 5 visible edges.

To summarize, in the experiments the walking robot could be controlled at its top speed of 3 cm/s. Tracking using EPIC and validation proved to be robust in an environment with all walls of the same color. The integration of redundant features proved feasible. Within the mockup the robot was located with an accuracy of better than 5 cm, which is sufficient to fulfill the task and place the welding equipment.

5.3 Mobile Robot Navigation

To obtain experience in the prospective scenario of a service robot, the camera is mounted on a mobile robot which should navigate in an office environment. A CAD model of the office is generated and some primitive tasks are defined. Figure 23 and Extension 7 show the result of

the vision process where the robot is moving about two meters. The model is re-projected as in Figure 18.

Figure 23: Tracking in an office environment from a walking robot. The pose estimated from the lines is re-projected into the image.

Figure 24 shows the pose estimates for the path of the robot in the x/y plane. The robot used is a Pioneer 2 DX with three wheels. Therefore the motion on even ground is smooth, the image motion is smooth and the system performs reliably.

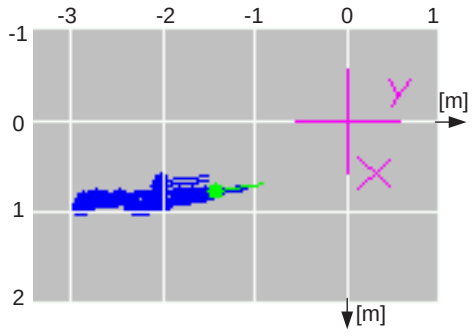


Figure 24: Projection of the robot path moved into the ground plane.

The tests indicate the same conclusions as the RobVision project (see section 5.2). Large

image deviations caused by high angular velocities are difficult to track. To increase the reliability of the whole sensor-robot system the camera will be mounted on a camera head, which is able to compensate most of the turning motion [CrVi00].

6 Summary and Future Work

This paper has presented the V4R framework for model-based visual servoing for robotic applications. The system is simple to use and performs robustly as demonstrated in three sets of experiments. A particular feature of V4R is the technique of Edge Projected Integration of Cues (EPIC), which provides significant robustness in extracting edge features such as lines and ellipse arcs. This robustness is obtained by making extensive use of model knowledge. EPIC uses knowledge of edge-background adjacency to separate the object from the background. In a final feature validation step, the topology of the features in the object model is exploited to select the best set of features among the possible candidates. The result is a framework which can be used in realistic environments. For interested researchers, the authors can provide the source code of V4R.

One limitation of visual tracking is the maximum bound on target velocity (and acceleration). The work in the RobVision project showed that the integration of accelerometers (gyros) improves tracking by stabilizing the camera and by improving the pose estimate of the camera. The rationale is that accelerometers can track fast motions, while slow motion is tracked more accurately using vision. It will be future work to integrate a full 6D accelerometer head with the visual tracking system. Due to the modularity of the system, the integration with other sensors will be easy to implement. Another advantage of this approach is in handling cases where only a few features are visible, for example close to a wall.

Future work is needed to realize the full potential of the validation technique. Presently only part of the model knowledge is exploited. Other Gestalt properties, for example, the region and bounding polygon information provided by a wire-frame model, will be incorporated to further improve validation and robustness.

Finally, the initialization requires that the start pose is set or features are selected manually. Object recognition methods can solve this problem, however they often require substantial computing time and perform poorly with clutter and in uncontrolled lighting situations. It is again proposed to utilize the model knowledge to focus the recognition processes so as to enable the rapid finding of the target object and therefore the fast initialization for the tracking process. Such a technique is also necessary to enhance error recovery and to enable fault tolerant behaviors.

Acknowledgements

The authors would like to express their deep gratitude to Peter Corke for his invaluable help to improve the presentation of our work. This work has been mainly supported by the RobVision project Esprit 28867 and is partly supported by the Austrian Science Foundation (FWF) under grant P13167-MAT and Project GRD1-1999-10693 FlexPaint.

References

[AlSh89] J. Aloimonos and D. Shulman, "Integration of Visual Modules - An Extension of the Marr Paradigm", Academic Press, Boston, 1989.

[BiVi01] J. Biber, M. Vincze, M. Ayromlou, W. Ponweiser: "Robust Real-time Tracking of Ellipse Arcs", OAGM Austrian Pattern Recognition Workshop, Berchtesgaden, 2001.

[BoWe89] M. Boldt, R. Weiss, E. Riseman, "Token-Based Extraction of Straight Lines", *IEEE SMC*, Vol.19, No.6, 1989, pp. 1581-1594.

[Chri99] Christensen, H.I., Ed.: "Computer Vision Systems", Proc. of the first Int. Conf. ICVS, Lecture Notes in Comp. Science, Springer, 1999.

[Cork96] Corke, Peter I.: *Visual Control of Robots: High Performance Visual Servoing*, Research Studies Press (John Wiley), 1996.

[CrVi00] Chroust S., Vincze m.,Traxl R., Krautgartner P., Ecaluation of Processing Architecture and Control Law on the Performance of Vision-Based Control Systems, IEEE AMC, March 2000.

[CrSt92] J.L. Crowley, P. Stelmaszyk, T. Skordas, P. Puget, "Measurement and Integration of 3-D Structures by Tracking Edge Lines", *Int. J. of Computer Vision*, Vol.8, No.1, 1992, pp. 29-52.

[DiGr88] Dickmanns, E.D., Graefe, V.: *Dynamic Monocular Machine Vision* and *Applications of Dynamic Monocular Machine Vision;* Machine Vision and Applications, pp.220-261, 1988.

[EbBa00] C. Eberst, M. Barth, et.al., "Robust Vision-based Object Recognition Integrating Highly Redundant Cues for Indexing and Verification", *IEEE ICRA 2000*, pp. 3757-3764.

[FeMa98] W. Feiten, B. Magnussen, J. Bauer, G.D. Hager, K. Toyama, "Modeling and Control for Mobile Manipulation in Everyday Environments", 8^th *Int. Symp. Robotics Research*, 1998.

[FiBo81] Fischler, M.A., Bolles, R.C.: *Random Sample Consensus: A Paradigm for Model Fitting;* Communications of the ACM Vol.24(6), pp.381-395, 1981.

[FiPi99] Fitzgibbon, A., Pilu, M., Fisher, R.B.: "Direct Least Square Fitting of Ellipses", IEEE PAMI 21(5), 476-480, 1999.

[GaGe89] E.B. Gamble, D. Geiger, T. Poggio, D., Weinshall, "Integration of Vision Modules and Labeling of Surface Discontinuities", *IEEE Trans. PAMI*, Vol.19, No.6, 1989, pp. 1576-1581.

[Genn92] Gennery, D.B.: *Visual Tracking of Known Three-Dimensional Objects;* Int. J. of Computer Vision Vol.7(3), pp.243-270, 1992.

[HaTo98] Hager, G.D., Toyama, K.: "The XVision-System: A Portable Substrate for Real-Time Vision Applications," *Computer Vision and Image Understanding* 69(1), pp. 23-37, 1998.

[Harr92] Harris, C,: *Tracking with rigid models*, In A. Blake and A. Yuille, editors, Active Vision, pages 59-73, MIT Press, 1992.

[HoAh89] W. Hoff, N. Ahuja, "Surfaces from Stereo: Integrating Feature Matching, Disparity Estimation and Colour Detection", *IEEE Trans. PAMI*, Vol.11, No.2, 1989, pp. 121-136.

[Hond01] ASIOM The Honda Humanoid Robot, http://world.honda.com/robot/, 2001.

[HuHa96] Hutchinson, S., Hager, G.D., Corke, P.: "Visual Servoing: A Tutorial," *IEEE Trans. RA* Vol.12(5), 1996.

[InYo00] K. Inoue, H. Yoshida, T. Arai, Y. Mae: "Mobile Manipulation of Humanoids - Real Time Contorl Based on Manipulability and Stability", *IEEE ICRA 2000*, pp. 2217-2222.

[JoLa00] S. Jörg, J. Langwald, J. Stelter, G. Hirzinger, C. Natale, "Flexible Robot-Assembly using a Multi-Sensor Approach", *IEEE ICRA 2000*, pp. 3687-3694.

[KiNe98] Kim, D., Nevitia, R.: *Recognition und localization of generic objects for indoor navigation unsing functionality;* Image and Vision Computing Vol.16, pp.729-743, 1998.

[KoKa92] Kosaka, A., Kak, A.C.: *Fast Vision-Guided Mobile Robot Navigation Using Model-Based Reasoning and Prediction of Uncertainties;* CVGIP: Image Understanding Vol.56(3), pp.271-329, 1992.

[KrCh00] D. Kragic, H.I. Christensen, "Cue Integration for Manipulation", in [ViHa00], pp. 1 - 16.

[Kr97] Karl Kraus, *Photogrammetry*, Vol.2, Advanced Methods and Applications, Dümmler Verlag, Bonn, pp. 205-215, 1997.

[LeMe99] M.-S. Lee, G. Medioni, "Grouping into Regions, Curves, amd Junctions", *Computer Vision and Image Understanding*, Vol.76, No.1, 1999, pp. 54-69.

[Lowe92] Lowe, D.G.: *Robust Model-Based Motion Tracking Through the Integration of Search and Estimation;* Int. J. of Computer Vision Vol.8(2), pp.113-122, 1992.

[NaMu00] Nagel, H.H., Müller, T., Gengenbach, V., Gehrke, A.: *Spatially-Adaptive Filtering in a Model-based Machine Vision Approach to robust Workpiece Tracking;* in [ViHa00], 1999.

[OhKo98] Ohya, A., Kosaka, A., Kak, A.: *Vision-Based Navigation by a Mobile Robot with Obstacle Avoidance Using Single-Camera Vision and Ultrasonic Sensing;* IEEE Transaction on Robotics and Automation Vol.14(6), pp.969 - 978, 1998.

[1] Rosin, P.L.: *Further Five-Point Fit Ellipse Fitting*, Graphical Models and Image Processing 61, 245-259, 1999.

[Schi00] B. Schiele, "Towards Automatic Extraction and Modelling of Objects from Image Sequences", 8th International Symposium on Intelligent Robotic Systems 2000, Reading, UK, July 2000.

[ShOk00] Shirai, Y., Okada, R., Yamane, T.: *Robust Visual Tracking by Integrating various Cues;* in [ViHa00], 2000.

[ThRe01] Thompson, R.L., Reid, I.D., Munoz, L.A., Murray, D.W.: *Providing synthetic views for teleoperation using visual pose tracking in multiple cameras*, IEEE SMC Part A 31(1), 43 -54, 2001.

[ToSc97] Tonko, M., Schäfer, K., Heimes, F., Nagel, H.H.: *Towards Visually Servoed Manipulation of Car Engine Parts;* IEEE ICRA, pp. 3166-3171, 1997.

[ToHa99] Toyama, Kentaro; Hager, Gregory D.: *Incremental focus of attention for robust vision-based tracking*, International Journal of Computer Vision 35(1), Pages 45-63, 1999.

[ViWe97] Vincze, M., Weiman, C.: *On Optimising Window Size for Visual Servoing;* IEEE ICRA, pp. 2856-2861, Albuquerque, April 22-24, 1997.

[ViHa00] M. Vincze, G.D. Hager, Eds., *Robust Vision for Vision-Based Control of Motion*, IEEE Press, 2000.

[ViAy99] Vincze, M., Ayromlou, M., Kubinger, W., "An Integrating Framework for Robust Real-Time 3D Object Tracking,' ' *Int. Conf. on Vision Systems*, Gran Canaria, pp. 135-150, 1999.

[Vinc00] Vincze, M.: *Real-Time Vision, Tracking and Control - Dynamics of Visual Servoing;* IEEE ICRA, invited presentation, San Francisco, April 20-25, 2000.

[Vinc01] Vincze, M.: *Robust Tracking of Ellipses at Frame Rate;* Pattern Recognition 34(2), 487-498, Elsevier Publishers, 2001.

[ViBe01] M. Vincze, C. Beltran, A. Gasteratos, S. Hoffgaard, O. Madsen, W. Ponweiser, M. Zillich: A System to Navigate a Robot into a Ship Structure, ICVS 2001.

[WrAz97] Wren, C.R., Azarbayejani, A., Darrell, T., Pentland, A.P.: *Pfinder: Real-Time Tracking of the Human Body;* IEEE Transactions on Pattern Analysis and Machine Intelligence Vol.19(7), pp.780-785, 1997.

[WuHi97] Wunsch, P., Hirzinger, G.: *Real-Time Visual Tracking of 3D-Objects with Dynamic Handling of Occlusion;* IEEE ICRA, 1997.

[Zele98] Zelenka, R.: *Robuste Objektverfolgung in Echtzeit mittels einer hierarchischen Schichtenarchitektur*, Diploma thesis, in German, Technical Univeristy Vienna, 1998.

7 Appendix 1: Index to Multi-Media Extensions

The multi-media extensions to this article can be found online by following the hyperlinks from www.ijrr.org.

Extension	Media type	Description
1	Video	Tracking a lamp shade (a full ellipse) using only gradient information.
2	Video	Tracking a lamp shade using EPIC. Cluttered background, reflectivity changes and partial occlusion are handled.
3	Video	Tracking an ellipse arc over cluttered background and with substantial changes of reflectivity.
4	Video	Tracking the base of a monitor using ellipse and lines for pose estimation.
5	Video	Tracking a ship mockup from a pneumatically actuated walking robot.
6	Video	Difficulty of tracking a section of the mockup with only 5 visible edges.
7	Video	Tracking in an office environment from a mobile robot.

Table 1: Index table of Extensions.