ROBUST OBJECT TRACKING FOR ROBOT MANIPULATION AND NAVIGATION

Michael ZILLICH, Dietmar LEGENSTEIN, Minu AYROMLOU and Markus VINCZE
Institute of Flexible Automation, Vienna University of Technology, Gusshausstr. 27-29/361, 1040 Wien, Austria
phone: +43/1/5041446, email: mz,dl,ma,vm@infa.tuwien.ac.at

Working Group TC V-12

KEY WORDS: Computer vision, Feature extraction, Object tracking, Real-time, Robots, Vision systems

ABSTRACT

Applications for visual control of a robot manipulator often require the pose (position and orientation) of an object as input for the control loop of the robot. On the other hand robot navigation usually needs the pose of the robot with respect to a world co-ordinate system. In both cases the same task of determining the pose of an object with respect to the camera has to be carried out. This object can be a tool to grasp or the environment to navigate in. We present a system that is able to robustly determine object pose in real-time from features in the image such as points, straight edges and ellipses using a 3D model of the object. As an example of a robot navigation task we present a walking robot which is to be used for inspection and welding of sections of large ships at a shipyard.*

1 INTRODUCTION

With the advance of computer and camera technology, vision is becoming a powerful sensing mode to control the motion of robots and other machines. Examples are industrial, service, and personal robots that need functions to find, track, and locate objects. A large number of systems for Visual Servoing have been built in the last decades (for extensive review see (Vincze et.al.,1999)). However, the number of commercial applications is small. One major roadblock to widespread use of computer vision for control tasks especially in complex environments is the lack of robustness inherent in many vision algorithms. To be applicable in unstructured industrial or everyday environments such a system must overcome the major problem of robustness against varying lighting conditions, reflections (specular highlights) or a changing background. Another important issue is speed. Modern computers get faster and faster, but the processing of large quantities of data involved in acquisition and processing of images still poses problems for real-time applications. The employed methods must be fast enough not to impede the speed of the steered robot.

We present V4R (Vision For Robotics) as a system that aims to robustly track and determine pose of one or several objects in real-time with no constraints on the environment. It uses a model of an object and extracted image features to calculate 6D pose.

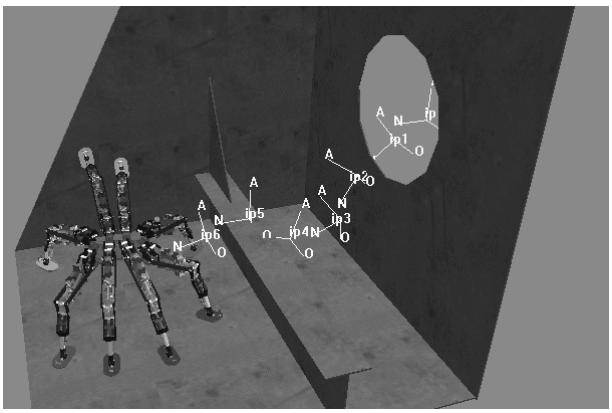

Figure 1: Application example: Robug IV inside a mockup of a ship section

Figure 2 shows an example where V4R is part of a robotic system. The RobVision project (RobVision) has the goal to steer the walking mobile robot Robug IV (provided by project partner PORTECH) into sections of large ship bodies at the

*This work is partly supported by the Austrian Science Foundation (FWF) under grant P13167-MAT, Esprit Project E28867 RobVision and Project GRD1-1999-10693 FlexPaint.

Odense Steel Shipyard in Odense, Denmark. Here the robot could be used for tasks such as inspection and welding. The robot is provided with its current pose relative to the ship by the V4R system. V4R and a parallel stereo vision system obtain a sparse object model from a CAD system (provided by Aalborg University, Denmark), which extracts relevant model features from the rather complex model of the whole ship. A stereo head with two cameras (constructed at DIST, University Genova) is mounted on the robot. The stereo head compensates jerky robot motions and delivers stabilised images to the vision systems. V4R integrates 2D features extracted from the images and 3D features provided from the stereo system with the model data to estimate the pose of the robot relative to the ship. Figure 1 shows a simulation of the walking robot inside a ship section.

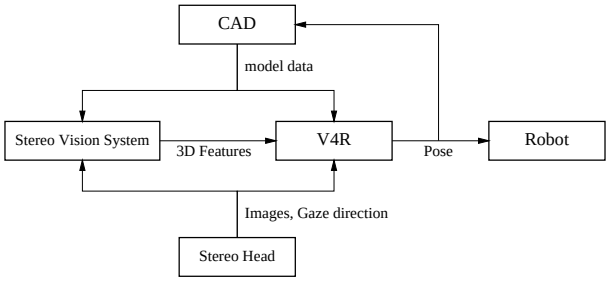

Figure 2: Application example: V4R within the RobVision project

1.1 Key elements of V4R

To meet the real-time constraint features are tracked and only searched locally within small tracking windows. The tracking window sizes, which determine the processing time for each feature, can be adjusted in order to limit the total processing time of the vision system.

Various methods are employed to enhance robustness. Model information is used to predict feature appearance in the image. Image features are grouped and their information combined to render the tracking of each feature more robust. For robustness of feature search, multiple image properties like colour, intensity and gradient are integrated. Robustness of the pose estimation is achieved using redundancy (as many as possible image features are used for pose calculation) and outlier detection.

Object pose is calculated using the 2D information extracted from the image and the matched 3D information provided by the object model. The pose determination algorithm employed accepts any number and various types of features: points in 2D and 3D, lines, surface normals of regions, circles (ellipses). The object model is basically a boundary representation consisting of edges (straight edges, circle arcs) and surfaces, with co-ordinates given in the object co-ordinate system.

Successful tests were conducted tracking simple-shaped objects in front of cluttered backgrounds with no constraints on lighting.

1.2 Related Work

There have been several specific attempts to build general integrating software systems for visual-based control. Commercial vision packages provide algorithms, but interfacing and structure do not allow visual servoing or tracking. The first system that integrates the components for Visual Servoing is the dynamic vision approach (Dickmanns,1994). It exploits the temporal evolution of geometric features to build a model of the perceived world. Object behaviour is used to predict image positions. Tracking is then used to confirm or update motion behaviours. Tracking itself uses simple detectors to locate lines at several orthogonal search areas. Extensive use of models is exploited to predict behaviours and to obtain confidence in the actions of objects in the world. The approach was first used to steer cars and is now extended to 3D for air vehicle control.

The “Vision as Process” project developed a software system to integrate vision and an active head and the interactions between them (Crowley et.al.,1995). Integration united 2-D image with 3-D structure data to control the head motions. To enable three dimensional control of robots a set of basic tasks for image-based visual servoing has been extended to a software environment for eye-in-hand visual servoing (Marchand,1999). It has been used to follow pipe systems using one camera. A milestone for tracking tasks has been the XVision system that can be used for stereo visual servoing (Hager et.al.,1998). Servoing uses points and lines on the robot gripper and on the object for alignment. The last two approaches need manual initialisation of the visual features. The presentation of the objects in the image is purely 2D and provides no means to build topologies or reuse features for groupings of features.

We will now briefly outline the remaining part of this paper. Section 2 explains the general idea of the vision system proposed and an overview of its components. Section 3 describes the way image features like lines and ellipses are

tracked. Section 4 goes into details of robustness. The main task of pose determination is explained in section 5. Section 6 presents the results of tests along with example images. And section 7 finally gives an outlook on tasks that still lie ahead and possible extensions of the system.

2 SYSTEM OVERVIEW

Figure 3 shows the main components and function of V4R. A CAD database provides the system with model data (model features). Features are extracted from the images of one or several cameras (image features). Pose is calculated using image and model features.

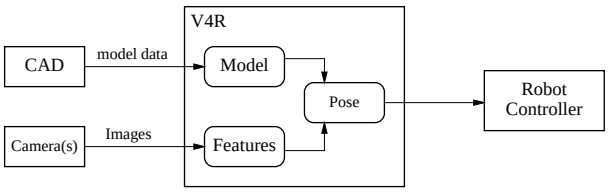

Figure 3: V4R system overview

As outlined in the introduction, the vision system should be able to track one or several objects. In the case of robot navigation one can think of one large object which is the whole environment. For picking up or assembling parts the system will have to track several objects at once. Furthermore it should be possible to utilise several cameras. For example several fixed cameras can be used to observe the workspace of a robot manipulator or a stereo camera head can be mounted on a mobile robot as in the case of RobVision.

2.1 Scene

For handling multiple objects the system uses the concept of a scene (see figure 4). The scene contains one or several objects. For each object there are several views, one for each camera. A view contains the features extracted from the images of a certain camera, such as lines and ellipses. Image features are mostly 2D but given a range image or stereo images also 3D features could be extracted. Current work however concentrates on 2D features. The object also contains a model which consists of a list of 3D model features like 3D model lines and circles.

So an object consists of the following components:

- A model. The model is a list of model features. Basic model features are edge features (straight lines, circle arcs), junctions (the intersections of edges) and regions (surface patches bounded by edges). Currently implemented model features are lines, ellipses and junctions. A model junction is defined as a 3D point and a list of connected edges. The 3D co-ordinates are given relative to the object co-ordinate system. A model line is defined by its two end junctions.
- A list of object views, one object view for each camera. Each object view contains 2D image features. There are corresponding types of image features for the model features: lines, elliptic arcs (since generally circles appear as ellipses in the image), junctions and regions. These image features are linked to the model features. This link is required later for the pose calculation algorithm. Each view has its own view pose, which is the pose of the object relative to the camera associated with that view.
- And finally the object pose. This is the pose of the object relative to a reference co-ordinate system. In the RobVision case of a mobile robot the reference co-ordinate system is the robot co-ordinate system. For a fixed robot manipulator the reference system could be the base of the manipulator.

2.2 Models

As outlined in the introduction the object models are simple boundary representations. There is a vast number of possible model representations in computer vision. The reason we choose this form of representation results from the requirements we have for our models. Since the basic image processing technique in V4R is tracking of single image features and for pose determination we need a link from each image feature to a corresponding model feature, the object model basically consists of such model features, which have attributes such as colour, texture or type of machining. For the models we choose model features that

- are likely to appear in man-made environments: straight edges, circular arcs, flat surfaces.

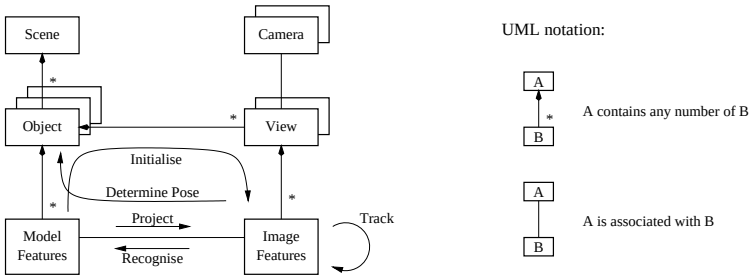

Figure 4: Scene overview

- have an image feature equivalent which is fast to find and track
- are easy to import from existing CAD systems, since CAD models are commonly available in industry.

A further advantage is the explicit 3D representation that allows to link the visual process to grasp or assembly planning techniques.

2.3 Vision cycle

The vision system operates continuously. It starts with a list of model features (in case of RobVision provided by an external CAD system), a list of corresponding image features and an initial estimate for the object pose. When the system starts the image features are obviously not found yet. The only information at hand are the model features and the initial pose estimate. So the model features are projected into the camera image(s) to obtain initial estimates for the image feature locations. Assuming that the initial pose is close enough to the actual pose the image features will eventually be found (see section 3 for details on image feature extraction). Once features are found they are tracked, i.e. in subsequent images they are only searched locally in a tracking window which follows the feature. With all found image features and their corresponding model features the system then estimates the pose of the object (see section 5).

So one cycle of the vision loop consists of the following steps (which are performed for each object):

1. - If a feature is not yet found in the image project it using the model feature and the current pose estimate and try to find it at this estimated location.
   - If the feature was found in the last step then track it. Search for the features locally in the tracking window.
2. Determine pose from image features found and model features for each view, i.e. the pose of the object relative to each camera. Then integrate the view poses to one object pose, i.e. the pose of the object with respect to a common reference co-ordinate system.

New model features and corresponding image features can be added or old ones removed during operation. This is useful because different features become visible as the robot moves through the environment.

2.4 System integration

To be of use for actual applications V4R needs to be integrated with other components of a robotic system. V4R therefore offers a simple ASCII messaging protocol over TCP/IP. From a CAD database V4R accepts commands like `AddObject`, `AddModel` or `AddFeature`. To its clients, e.g. a robot, it sends out `ObjectPose` messages. V4R enables monitoring of its status (how many features were found, how reliable is current pose estimation) via `SystemStatus` messages. Furthermore it offers for experienced users commands to tune system parameters at runtime as well as commands to enable display of features found and logging of system performance.

3 IMAGE FEATURE TRACKING

We will now shortly outline how we achieve real-time operation of the vision system by using image feature tracking. Please refer to the cited papers for more detailed descriptions.

3.1 Lines

A tracking algorithm initially proposed by XVision (Hager et.al.,1998), and further developed by INFA, is applied to determine straight lines. First a sub-image of the whole camera image is grabbed from the full image using a tracking window (region of interest). The tracking window follows the position and orientation of the feature. Then the edge is searched locally in the sub-image. Note that the tracking window is oriented in such a way that in the sub-image the edge appears (almost) vertical. This significantly simplifies the edge finding. The edge finding methods use a scheme of Edge-Projected Integration of Cues (EPIC) that combines gradient, intensity, object side, and possibly colour information for robust detection (Vincze et.al.,1999). The robust detection of the geometry of a line, i.e. the fitting of a line into the detected edge pixels (edgels), is achieved with a probabilistic scheme (RANSAC) initially proposed by (Fischler et.al.,1981). Once the line is found the position and orientation of the tracking window is updated.

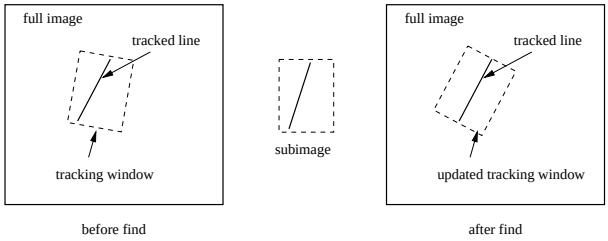

Figure 5: Line tracking

3.2 Elliptical Arcs

The general idea of ellipse tracking is to place many small tracking windows of height one pixel, i.e. actually tracking lines, on the ellipse. Then search for the edge step in these one-dimensional sub-images. This gives a number of edge points of the ellipse. The ellipse is then robustly fitted into these points using again the RANSAC scheme, this time with a different geometric constraint. For details on ellipse tracking see (Vincze,2000). Again the positions and orientations of the tracking lines are updated after the ellipse has been successfully found.

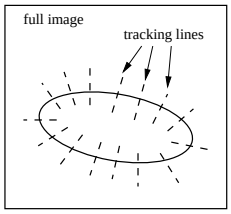

Figure 6: Ellipse tracking

3.3 Junctions

By junctions we have defined the point where two or more edges intersect. Therefore the determination of junctions relies simply on the determination of the lines and/or arcs that intersect on each junction. However, not all junctions can be generated by intersecting the edges. Straight lines at corners are intersected using the method of (Förstner et.al.,1987). The basic rule is that the angle between the cutting edges must be greater than a certain minimum, otherwise the intersection is unreliable. The angle depends strongly on the view. Furthermore there are junctions that join a line and arc with one common tangent. In this case the junction can never be calculated as intersection, but the closeness of the edges is validated.

3.4 Real-time

In order to guarantee that processing time of the whole vision cycle stays within a given limit, e.g. field rate (20 ms), the processing time for each feature must be limited. Therefore the sizes of the sub-images that are grabbed from the full image are fixed. If we approach an object then the tracked edges and their surrounding tracking windows will become longer. But as the size of the sub-image is fixed, we have to subsample the image along edge direction. It shows that the subsampling rate does not affect the edge finding algorithm.

4 ROBUSTNESS

Robustness is a key issue for the successful application of a vision system for practical use. In an unconstrained real world environment the system has to cope with variations in lighting, partial occlusions of objects, unknown and changing backgrounds, reflections (specular highlights) and so on. V4R achieves robustness at different levels.

4.1 Robustness at image processing level

Image processing is in our case mostly edge detection. The simplest way of detecting edge pixels (edgels) is to use some edge filter. We are currently using a 1-D Prewitt mask (note that since the line to detect is vertical in the sub-image we only need gradient in horizontal direction). Applying the edge filter will lead to many edgels. Thresholding is the usual way to discard edgels that are considered too weak. The problem is of course that quite strong edgels could result from the background. So throwing away weak edgels might actually throw away the edge of the object we are interested in. The solution to this problem is to use more properties than just gradient for edge detection.

Since an edge is the boundary of an object we know that on one side of the edge we have the object (or on both sides if the edge is no contour edge). Therefore any edgel that is part of the object edge should have to the right (or left, depending on edge direction) pixel values (intensity values, colour values, texture values) which are the same as those of the object (or to be more precise: of the adjacent object surface). Using these additional information we can discard any background edgels as they have arbitrary values on their sides.

4.2 Robustness at feature level

At the level of features we enhance robustness by connecting features. Tracking of lines along their direction is difficult. The line tracker can only follow rotations and motions perpendicular to the line. Using junctions to intersect lines however solves the problem. Each line can update its orientation and position perpendicular to its direction. The position along line direction (and also line length if we have junctions at both ends) is calculated from the junctions.

Robustness at feature level is also enhanced using model information. Lost features are projected onto the image using their model information and the current estimated pose of the object. As it is not likely that all features of an object are lost at the same time there should still be enough features to calculate pose. Then it is easy to project one or a few lost features.

4.3 Robustness at object level

At the object or pose calculation level V4R is robust by using as many as possible features for pose determination. Therefore it does not affect pose calculation too much if some features are lost. As outlined (and explained in more detail in section 5) the pose determination algorithm needs the attribution of an image feature to a model feature. Now if this attribution is incorrect (for example the image feature search fails to find the correct feature but still finds something it believes to be correct) then this incorrect feature – model matching will affect the pose calculation. Therefore V4R performs outlier detection and removes such wrong observations.

5 POSE DETERMINATION

The main task of V4R is to calculate 6D object pose from image features, three positional and three rotational parameters. The algorithm employed is based on an algorithm proposed by (Wunsch,1997) and was modified for providing accuracy estimation of the calculated pose as well as outlier detection. The Wunsch-algorithm was chosen because it is fast and therefore well suited for object tracking.

The key idea of the Wunsch-algorithm is to calculate 3D discrepancies between model features transformed into the camera co-ordinate system and observed image features. For example the discrepancy for an image point would be the 3D distance of the model point from the line-of-sight going through the focal point of the camera and the image point. For each type of feature (2D point, 3D point, 2D line, ellipse, surface normal) equations can be written that describe these discrepancies.

These discrepancies in 3D space are treated as observations. The algorithm now minimises the quadratic errors of these observations. Most of these observations are non-linear in the unknown pose parameters. After linearisation we can combine all equations to one matrix equation. This matrix equation is then solved using least-squares adjustment.

Due to the linearisations the algorithm theoretically works only for small angular deviations in the pose. Experiments however show that the algorithm also converges for larger angular deviations.

A wrong attribution of an image feature to a model feature will result in a gross observation error. Outlier detection (Kraus,1997) is used to remove such observations. This is possible as there are typically more observations than unknowns. The observation with the largest error is removed and pose calculation repeated. This process is iterated until no more gross errors are detected.

6 RESULTS

The feasibility and robustness of the approach is demonstrated with a robot grasping a green household sponge as example of a service task. The robot has a colour camera attached to the gripper. The workpiece is on the plate. There is no specific finishing, the plate is particularly rough and has stains of rust. Lighting comes from two overhead neon lights and through the window behind the viewer. Figure 7 shows snapshots from a descent motion. The sponge is found by searching first for green regions in the image at low resolution. Searching for the outline of this region, the top surface of the sponge is found in a few steps. With the top surface four corners are available, which is sufficient information to determine the 6D pose. Using an extended Kalman filter the signal is smoothed and the control signal to steer the motion of the robot calculated.

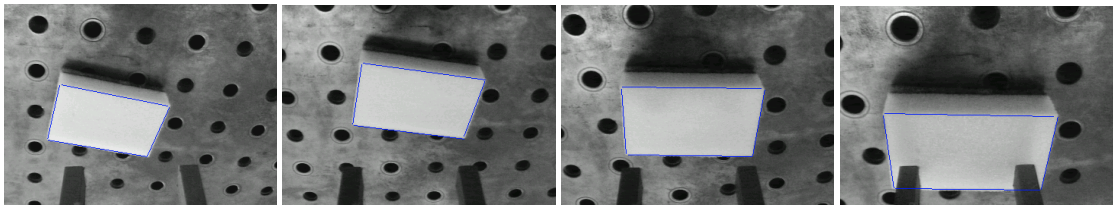

Figure 7: Snapshots during the descent to grasp the sponge. Note the changing light intensity, due to the shadow from the approaching robot.

Figure 8 shows images of a welded steel mockup which resembles in colour and texture the type of surfaces that are present inside a ship body. The left picture shows detected lines. Some of them, especially the shorter ones at the bottom are detected rather poorly due to shadows. Pose calculation is however still possible. The picture on the right shows the CAD model of the object projected onto the image using the pose calculated from the detected lines.

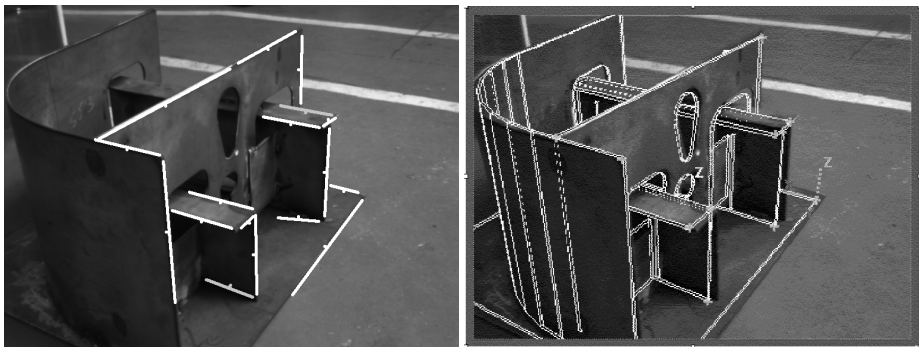

Figure 8: Mockup: tracked lines and projected CAD model

7 OUTLOOK AND CONCLUSION

We have presented a system for robust real-time object tracking and pose determination. Robustness is achieved at all levels of the system from low level image processing to image feature tracking to object tracking. The real-time constraint is met by choosing tracking window sizes so that the processing time stays within a given limit. V4R integrates with other components of a robotic system by providing a means of communication over TCP/IP using messages.

Open tasks are the integration of more image properties such as texture, optical flow or disparity from stereo images. Another important issue that will be addressed in the near future is automated search for image features. Only if a sufficiently good initial pose estimate is available image features can be initialised by projecting model features into the image. If no pose estimate is available features have to be searched in the whole image and if found have to be attributed to model features (object recognition).

It is possible to indicate levels of detail for the features and feature attributes in the model. For example, typical area features such as coloured or textured regions specify search level one (coarse) and edges search level two (fine). Then it is possible to run initialisation fully automated, as level one features are searched first and, if successful, the search is extended to level two. Such an implementation is an automated extension to the Incremental Focus of Attention methodology (Toyama et.al.,1996). This knowledge could for example be stored in more intelligent object models.

REFERENCES

Crowley, J.L., Christensen, H.I.: *Vision as Process*; Springer Verlag, 1995.

Dickmanns, D.E., *The 4D-Approach to Dynamic Machine Vision*, Conf. Decision & Control, pp. 3770-3775, 1994.

Fischler M.A., Bolles R.C., *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*, Communications of the ACM **24**(6), 381-395, (1981).

Förstner W., Gülch E.: *A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centers of Circular Features*, ISPRS Intercommission Workshop, Interlaken, 1987.

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.

Kraus, K.: *Photogrammetry Volume 2, Advanced Methods and Applications*, Dümmler/Bonn, pp.205-215, 1997.

Marchand, E., ViSP: *A Software Environment for Eye-in-Hand Visual Servoing*, IEEE ICRA, pp.3224-3229, 1999.

RobVision project home page: http://robvision.infa.tuwien.ac.at

Toyama, K., Hager, G.D.: *Incremental Focus of attention for Robust visual Tracking*, CVPR'96, pp.189-195, 1996.

Vincze, M.: *Robust Tracking of Ellipses in Real-Time*, Pattern Recognition, accepted for publication, 2000.

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.

Vincze, M., Hager, G.D.: *Robust Vision for Vision-based Control of Motion;* IEEE Press, 1999.

Wunsch, P., *Modellbasierte 3-D Objektlageschätzung für visuell geregelte Greifvorgänge in der Robotik*, PhD Thesis, Munich University of Technology, pp.37-47, 145-149, 1997.