Local Failure Detection in a Redundant Camera System for Visual Manipulator Guidance

Christian Scheering and Alois Knoll
University of Bielefeld, Faculty of Technology
Postfach 10 01 31, D-33501 Bielefeld, Germany
{pcscheer,knoll}@techfak.uni-bielefeld.de

Abstract

We describe a robust method for 3D visual manipulator control using a redundant uncalibrated camera system. Under the assumption of a simple linear camera model a fusion equation is derived for which only three parameters have to be estimated regardless of the number of cameras. Robustness is increased by detecting measurement-failures exploiting a simple heuristic symptom generation. In simulations as well as in real experiments the feasibility of our approach under various failure-situations for a 3D positioning task of a six degree of freedom Puma 260 is demonstrated.[1]

1 Introduction

It is a common task for visual-guidance systems to *track* at least the object to be guided. The problem that is tackled in this work is how to detect sensory failures in such systems.

In our recent work [6] we presented a method which exploits several arbitrary positioned uncalibrated cameras to guide a 6 degrees-of-freedom robot towards a 3D goal-position. A simple linear relationship between 3D displacements and their projection onto each image plane was derived and estimated with three initial linear independent robot movements. This directly measured hand-eye transformation formed the base of a dynamic-look-and-move [4] strategy.

Doing this we could abstain from an laborious and cumbersome explicit calibration of the cameras and their transformation w. r. t. the robot. In each step the image-displacement between the manipulator and the goal was measured locally in every image. The single measurements were fused together using a Kalman filter in order to estimate a 3D correction. This procedure was repeated until the goal has been reached. We showed that increasing the number of cameras the overall accuracy and tolerance w. r. t. simple failures (e. g. increased noise, no data anymore) could be increased. On the other hand the goal could not be reached if the noise in one camera was significantly higher or the tracking-process stuck on another object although other cameras still provided sufficient data.

Hence the present work describes the idea of *detecting* several possible measurement failures in order to prevent the system from relying on erroneous information. We used a few so called [2] heuristic symptoms to generate residuals and evaluated them before a 3D correction is calculated. If an incorrect measurement is detected its contribution to the overall result is neglected.

2 Visual Guidance Strategy

The task is to position a manipulator holding an object over a desired target (Fig. 1) by utilising several uncalibrated cameras. The key idea of our visual control approach is to assume a parallel camera model for the image forming process. We than define an image-based position error in j different views and construct a simple linear equation from the parallel camera-model. The latter is called the *fusion equation* and is used for a resulting Cartesian correction movement. The parameters of this equation are estimated with

[1]The work presented in this paper has been funded by the German Research Foundation (DFG) in the collaborative research afford SFB 360, project D4.

a linear Kalman filter (KF) using measurements obtained by the cameras in different locations.

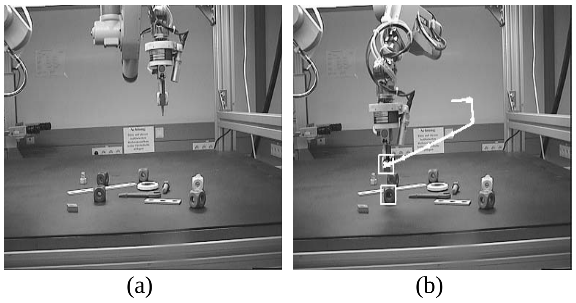


Fig. 1: Initial (a) and final position (b) of the manipulator and its projected trajectory as seen from one camera.

Using a parallel projection [3] $\boldsymbol{P}^j$ to generate the feature $\boldsymbol{f}^{\boldsymbol{j}}$ of a 3D point $\boldsymbol{m}$ given in *homogeneous* coordinates $\boldsymbol{m}^w = (m_x, m_y, m_z, 1)^T = (\boldsymbol{m}, 1)^T$ onto the j^{th} camera plane results in

$$\begin{aligned}\boldsymbol{f}^{\boldsymbol{j}} &= \begin{pmatrix} r^j_{11} & r^j_{12} & r^j_{13} & t^j_1 \\ r^j_{21} & r^j_{22} & r^j_{23} & t^j_2 \end{pmatrix} \cdot \boldsymbol{m}^w \\ &= (\boldsymbol{R}^j\, \boldsymbol{t}^j) \cdot \boldsymbol{m}^w \\ &= \boldsymbol{P}^j \cdot \boldsymbol{m}^w \end{aligned} \qquad (1)$$

Assuming that both the target and the manipulator can be represented as points in Cartesian 3D space, a simple error function for a linear point-to-point movement of a manipulator at $\boldsymbol{m}$ to a goal $\boldsymbol{g}$ results from defining an appropriate error-displacement vector $\Delta \boldsymbol{d_e}$ which is to be minimised:

$$\Delta \boldsymbol{d_e} = \boldsymbol{m} - \boldsymbol{g} \rightarrow 0 \qquad (2)$$

For the corresponding displacement feature $\Delta \boldsymbol{f}^j_e$ in the j^{th} camera using eq. (1) a simple linear relationship follows:

$$\begin{aligned}\Delta \boldsymbol{f}^j_e &= \boldsymbol{f}^j_m - \boldsymbol{f}^j_g \\ &= \boldsymbol{P}^j \cdot \boldsymbol{m}^w - \boldsymbol{P}^j \cdot \boldsymbol{g}^w \\ &= \boldsymbol{R}^j \cdot \boldsymbol{m} + \boldsymbol{t}^j - \boldsymbol{R}^j \cdot \boldsymbol{g} - \boldsymbol{t}^j \\ &= \boldsymbol{R}^j \cdot \Delta \boldsymbol{d_e} \end{aligned} \qquad (3)$$

Given a base of three orthogonal displacement vectors[2] $\{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3\}$ the error-displacement vector $\boldsymbol{d_e}$ can be calculated by their linear combination:

$$\boldsymbol{d}_e = \sum_{i=1}^{3} \xi_i \boldsymbol{d}_i, \text{ with } \xi_{1,2,3} \in R \qquad (4)$$

Using $\boldsymbol{R}^j$, the projected version of eq. (4) is:

$$\boldsymbol{f}^j_e = \boldsymbol{R}^j \sum_{i=1}^{3} \xi_i \boldsymbol{d}_i = \sum_{i=1}^{3} \xi_i \boldsymbol{R}^j \boldsymbol{d}_i = \sum_{i=1}^{3} \xi_i \boldsymbol{f}^j_i \qquad (5)$$

Hence under this assumption $\boldsymbol{f}_e$ is a linear-combination of the projected base using the *same* $\boldsymbol{\xi}$ as in 3D space. Calculating an appropriate set of scalars ξ_1, ξ_2, ξ_3 in the image space and inserting them into eq. (4) leads directly to the desired displacement-vector in the Cartesian 3D space.

Unfortunately, eq. (5) is under determined. Therefore at least two views are necessary yielding an over-determined system. Assuming a redundant multi-camera system with j different cameras, all views can be integrated simply by solving the following over-determined system:

$$\underbrace{\begin{pmatrix} \boldsymbol{f}^1_e \\ \boldsymbol{f}^2_e \\ \vdots \\ \boldsymbol{f}^j_e \end{pmatrix}}_{\boldsymbol{z}} = \underbrace{\begin{pmatrix} \boldsymbol{f}^1_1 & \boldsymbol{f}^1_2 & \boldsymbol{f}^1_3 \\ \boldsymbol{f}^2_1 & \boldsymbol{f}^2_2 & \boldsymbol{f}^2_3 \\ \vdots & \vdots & \vdots \\ \boldsymbol{f}^j_1 & \boldsymbol{f}^j_2 & \boldsymbol{f}^j_3 \end{pmatrix}}_{\boldsymbol{H}} \cdot \underbrace{\begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix}}_{\boldsymbol{\xi}} \qquad (6)$$

Eq. (6) is the aforementioned *fusion equation*, which plays the central role in our approach. Only three parameters need to be estimated (independent of the number of cameras) and only three initial test movements are necessary for this purpose. $\boldsymbol{H}$ is determined simply by measuring the projection of each test move in all the images.

We use a linear discrete Kalman filter to solve for the parameters of eq. (6). Assuming zero-mean, white-noise $\mathbf{v}$ and $\mathbf{w}$, the plant and measurement equation are:

$$\begin{aligned}\xi(k+1) &= \xi(k) + \mathbf{v} \\ \mathbf{z}(k) &= H(k) \cdot \xi(k) + \mathbf{w} \\ \mathbf{v} \sim N(0, \mathbf{Q}), &\ \mathbf{w} \sim N(0, \mathbf{R}) \end{aligned} \qquad (7)$$

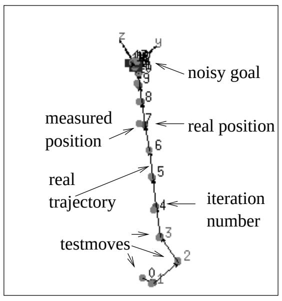
The incremental prediction and update solutions can be found in the literature [1]. In our approach the whole system dynamic is included in its noise

[2]Since only displacements are considered, the Δ is omitted in the sequel.

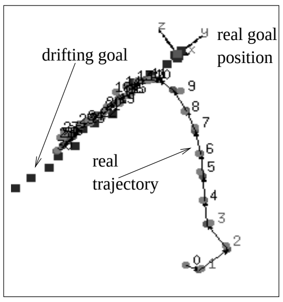
$\mathbf{v}$. We have chosen pure diagonal matrices for $\mathbf{Q}$, $\mathbf{R}$ and the initial state covariance $\mathbf{P}_{(\mathbf{0}|\mathbf{0})}$ with the diagonal elements $\sigma^2_{P_{(0|0)}} = 0.1, \sigma^2_Q = 0.01$ and $\sigma^2_R = 5.0$. The initial state-estimate is set to $\boldsymbol{\xi}_{(0|0)} = (1,1,1)^T$. For a point-to-point movement to a selected target the projection $\boldsymbol{f}_i^j$ of the manipulator during the three Cartesian test moves are obtained first. With the measured position residuals an initial down-scaled correction movement $\boldsymbol{d}_c = s \cdot \boldsymbol{d}_e, \in (0,1]$ is calculated. After each movement a new $\boldsymbol{\xi}$ is estimated. This is iterated until the target is reached (dynamic look-and-move). Results from simulations and real experiments showing the accuracy of the proposed method can be found in [6].

3 Residual Generation

To realise the above described visual-guidance scheme it is necessary to track the robots position in each image. When using several redundant cameras one should expect to benefit from the redundancy not only by an increased accuracy but also from an increased robustness. This is true (up to a certain amount) for noisy measurements (s. Fig. 2(a)) but in case of serious errors (e.g. the robot is occluded or the algorithm tracks something else) the target is not reached anymore.



(a) Faultless trajectory.



(b) Undetected fault.

Fig. 2: Trajectory from a faultless camera and a failing camera without detection (stopped after 30 iterations).

Fig. 2(b) shows the (simulated) case of a tracking-failure. After a few iterations the goal starts to drift away from its initial position. Without a failure detection the faulty measurements result in a trajectory far from the desired goal.

Under the constraint of a non moving target we defined five heuristic symptoms to detect several failures. All of them are measured and evaluated locally for each image:

- d_T
  The distance from the projection of the desired trajectory. This projection is a straight line from the manipulators position after the test moves towards the desired goal. A failure occurs if the manipulator moves to far away from the desired trajectory.
- φ_T
  The actual angle between the desired trajectory and the last piece of the manipulators actual trajectory (calculated from the actual and the last manipulator-position).
- d_m
  The projection of the last piece of the manipulators actual trajectory onto the desired trajectory between two successive images.
- d_g
  The moved distance of the goal between the first and the actual detected position.
- d_{mg}
  The remaining distance between manipulator and goal.

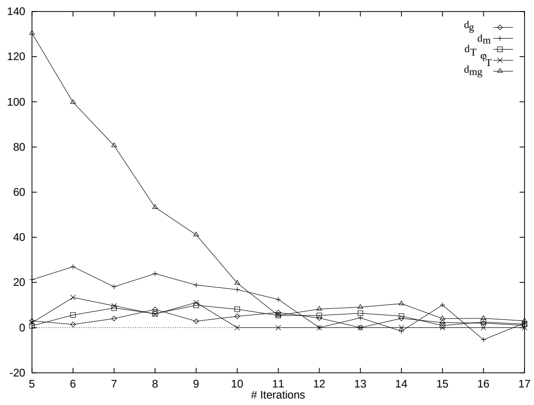


Fig. 3: Symptoms of a faultless camera.

Fig. 3 shows an example symptoms-plot of a faultless camera. With these symptoms we have defined three failure-indicators:

1. Trajectory is (not) ok.
   $traj\_ok = (d_T \leq 20 \text{ and } \varphi_T \leq 25.0)$. If the manipulator distance is less than 20

pixels and the angle between the actual trajectory-piece and the desired trajectory is less than 25 degrees then the manipulators is still on its desired path otherwise not. Unfortunately φ_T tends to oscillate very strong the nearer the manipulators comes to the goal. Therefore φ_T is set to zero if $d_{mg} \leq 20.0$.

2. The manipulator is not moving.
   $m\_static = d_m < 5.0$. This is the case if the manipulator-tracker stucks to another object. In order to prevent false alarms near the desired goal this indicator is evaluated only in a certain distance from the goal (if $d_{mg} > 20.0$).

3. The goal is not static.
   $g\_static = d_g \leq 10.0$. Due to noise the goal is allowed to move only a small distance from its initial detected position.

A failure is generated if any of the above indicators is on. The contribution of an erroneous measurement to the overall result is neglected simply by zeroing the corresponding innovation in the Kalman filter.

4 Simulation

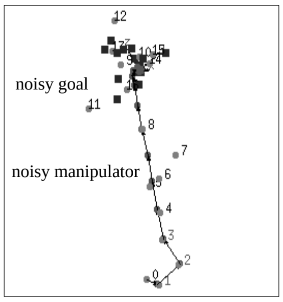
We first show examples of detected failures generated within a simulation. For each modelled failure we present an image of the situation observed by the failing camera. We used four cameras from which one produces a failure after the fifth iteration. The cameras are placed at distances between two and three meters from the goal. The initial distance between the goal and the manipulator is approx. 40cm. Each measurement is overlayed with simulated Gaussian noise with a deviation of two pixels. With the aforementioned heuristic rules we are able to detect the following failure situations:

- Heavy noise.
  Due to very high noise no reliable displacements are detectable. This results in a moving target (g_static is not true) and/or a non straight manipulator trajectory ($traj\_ok$ is not true) (s. Fig. 4(a)).

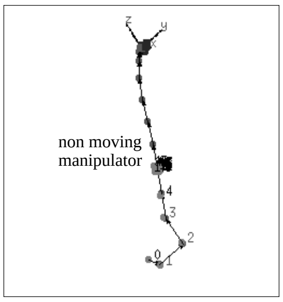
- Non moving manipulator.
  This includes the case of a stationary difference (always the same measurement). Detected with m_static (s. Fig. 4(b)).

- Manipulator drift and/or goal drift.
  E. g. if the tracker looses contact to one or both of them and drifts away gradually. The $traj\_ok$ flag indicates the drifting of the manipulator while g_static signals a non stationary goal (s. Fig. 4(c)).

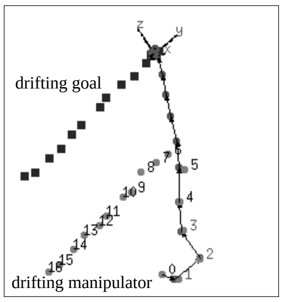
- Manipulator and/or goal walk.
  Due to a tracking failure something else is tracked instead. This failure is simulated by an incremental random-walk (s. Fig. 4(d)). As in the case of a drifting goal g_static signals a moving goal. A walking manipulator is detected either with $traj\_ok$ or m_static (if it is still on the trajectory but makes no progress).



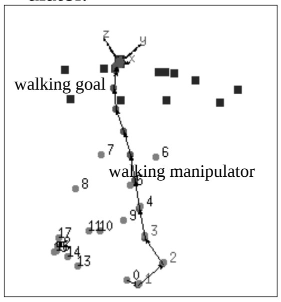
(a) Heavy noise.



(b) Non moving manipulator.



(c) Drifting goal and manipulator.



(d) Walking goal and manipulator.

Fig. 4: Trajectories in the presence of detected failures.

5 Real Experiment

In this section we present two common failure-situations detectable with our approach. The first is the tracking of another robot resulting either in

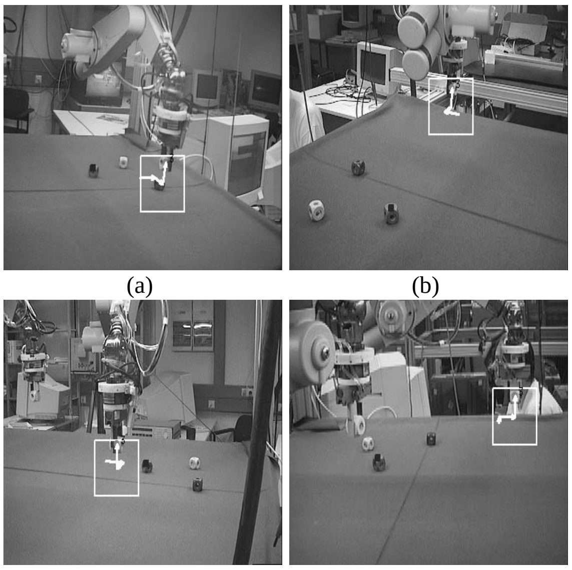


Fig. 5: Initial conditions of all cameras after the test moves.

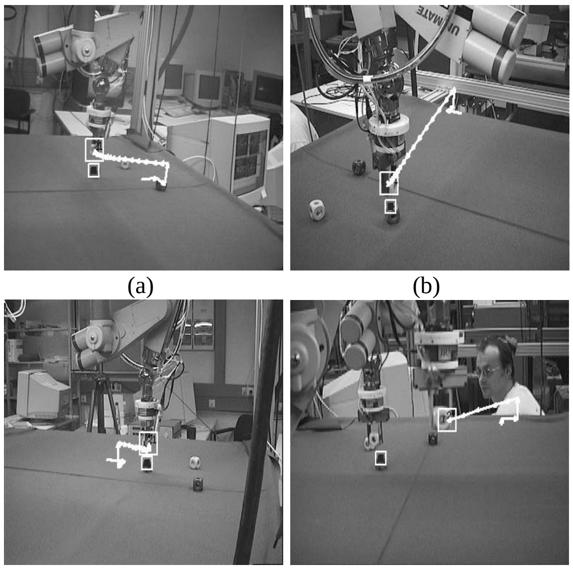


Fig. 7: Fault induced by a non moving second manipulator. The failing camera is (d).

a faulty trajectory or a non moving manipulator. The second failure is the occlusion of the robot and the goal. Similar to the simulation we used four cameras to guide the manipulator. The tracking was performed using a template-match. The task in the experiment was to position the robot carrying a yellow wooden cube above a desired target (also a cube). Fig. 5 shows the starting conditions of all cameras for all experiments after having performed the initial test moves. In Fig. 6 the corresponding last images for a faultless run

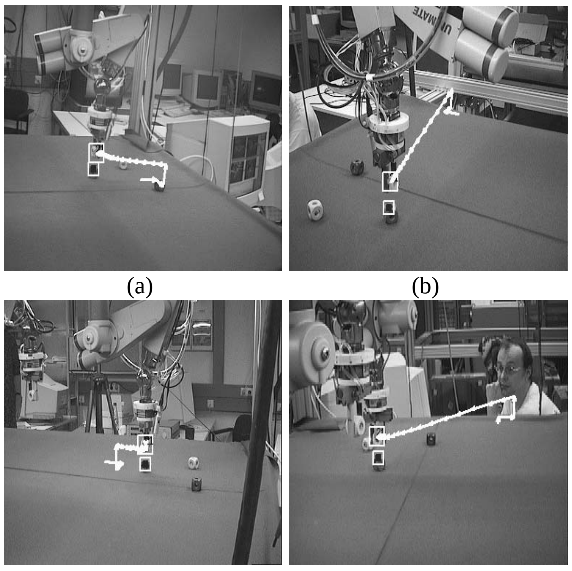


Fig. 6: Result of a faultless run.

are displayed showing the projected trajectory after task-completion within 22 iterations.

In the first experiment we moved manually an additional manipulator into the scene. Fig. 7 shows the last images of the corresponding sequence. Although the tracking-algorithm got caught after 8 iterations (which was detected in the 9th iteration) to the non moving manipulator (Fig. 7(d)) the goal-position above the desired target was reached roughly after 22 iterations (Fig. 7(a) to (c)).

Fig. 8 shows the result when the second manipulator is moving instead of holding its position. Fig. 8(d) shows the tracked trajectory of the manipulator in the foreground which was moved manually. In this case the target was successfully reached again due to the three faultless working cameras (Fig. 8(a) to (c)).

In the second experiment the failure was induced by occlusion of the manipulator in the sight of another camera. After 8 iterations the manipulator was occluded by a newspaper and became invisible. However, the tracking-algorithm detected the best match resulting in a randomly walking manipulator (Fig. 9(c)). As within the other experiments the desired goal-position could still be reached but took 24 iterations this time ((Fig. 9(a), (b) and (d)).

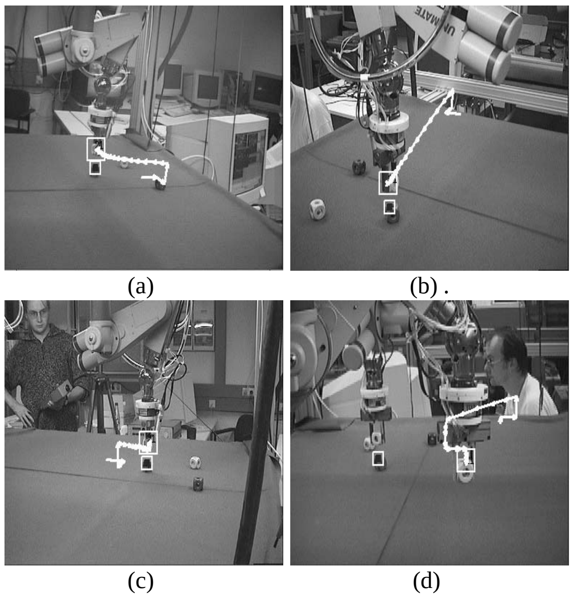


Fig. 8: Fault induced by a moving additional manipulator. The failing camera again is (d).

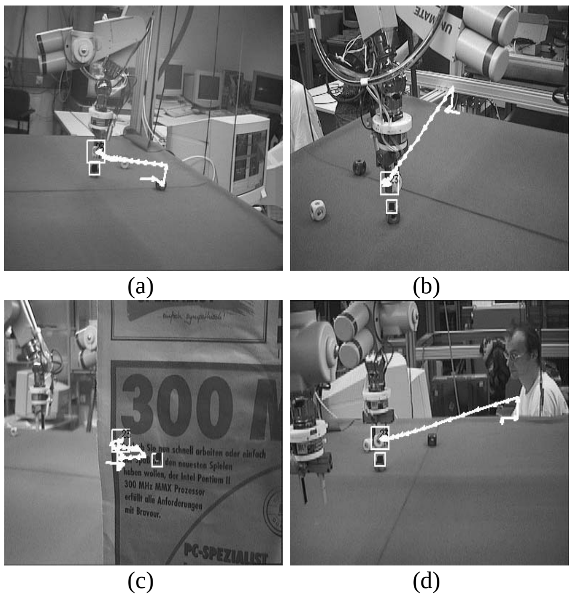


Fig. 9: Fault due to occlusion. The failing camera is now (c).

6 Conclusions

In this work we presented an uncalibrated visual manipulator control using redundant cameras. Independently of the number of cameras only three parameters have to be estimated. Although one should expect that using a redundant camera-system should increase the system-robustness we found situations in which relying on erroneous measurements would result in a complete failure. Therefore we introduced a heuristic-symptom based fault-detection. We demonstrated the appropriateness of this scheme both in simulations and real-world experiments.

As every model-based fault-detection our scheme suffers from the fact that unmodelled faults won't be detected and/or that the defined fixed thresholds in the residual evaluation are not usable in every situation.

Therefore our future work will tend to go into two directions which we would like to evaluate and to compare to each other. First we think of exploiting the Kalman filter itself to detect faults with ideas presented in [5] for detection of faults in navigation systems. On the other hand we would like to extend the residual evaluation with learning methods as described in [2] but using a neuro-fuzzy approach developed in our group [7] in order to overcome the fixed-threshold problem.

References

[1] Y. Bar-Shalom and X. Li. *Estimation and Tracking*. Artech House, 1993.

[2] P. M. Frank and B. Köppen-Seliger. New developments using ai in fault diagnosis. *Engineering Applications of Artificial Intelligence*, 10(1):3–14, 1997.

[3] D. Harris. *Computer graphics and applications*. Chapman and Hall, 1984.

[4] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Trans. Robot. Automat.*, 12(5), 1996.

[5] S. Scheding, E. Nebot, and H. Durrant-White. The detection of faults in navigation systems: A frequency domain approach. In *Proc. Int. Conf. on Robotics and Automation (ICRA-98)*, pages 2217–2222, 1998.

[6] C. Scheering and B. Kersting. Uncalibrated hand-eye coordination with a redundant camera system. In *Proc. Int. Conf. on Robotics and Automation (ICRA-98)*, 1998.

[7] J. Zhang and A. Knoll. Constructing fuzzy controllers with b-spline models - principles and applications. *International Journal of Intelligent Systems*, 13:257–286, 1998.