VISION BASED LEARNING OF GRIPPER TRAJECTORIES FOR A ROBOT ARM

Marco Matthias Päschke, Dr. Josef Pauli
Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität zu Kiel
Preusserstr.1-9 D-24105 Kiel Germany

97RO048

Abstract

As opposed to the usual approach of programming robot arms by giving a complete description of the trajectory (e.g. *direct programming*) this work is based on the idea of *Programming by Demonstration (PbD)*. Only few intermediate positions, which approximate the trajectory, are given and recorded by a stereo-camera-system. The idea is to extract the path of the manipulator from the images by geometrical connecting this positions to form a smooth trajectory. This is done by tracking the appearance of the manipulator in the sequence of stereo images. The manipulator position in previous images and the manipulator gripper appearance are used for locating it in the next images. Based on this sequence of positions a *trajectory-structure* is formed, which allows to execute a vision based movement. The trajectory is general in the sense that the starting position and the orientation can be specified variable. The approach can be used in automotive industry for vision based learning of trajectories to handle working tools.

1 Introduction

Instead of programming a robot by giving a complete description of the trajectory (e.g. *direct programming*) this approach is based on the idea of *Programming by Demonstration (PbD)*, which means to indirect program a system by giving examples. Basic ideas and related work on this topic can be found in [Heise 92] and [Friedrich95].
The operator uses the control panel to move the gripper in discrete steps to intermediate positions of the desired trajectory. A stereo-camera-system records this sequence and a computer-vision-system reconstructs a smooth 3D-trajectory. This trajectory only serves as an example for a whole class of trajectories having variable starting position and orientation. In the application phase this two unknowns have to be determined (e.g. manually by the operator or, perhaps, automatically with the help of a vision system). Furthermore, the information of the gripper trajectory can be used to anticipate collisions with unexpected objects in order to early interrupt the course.
The gripper is tracked in the images by using its appearance in the previous image as decribed in section 2. Section 3 briefly explains the calibration used. In section 4 a *trajectory structure* is defined, which allows to execute a vision based movement. Figure 1 shows the whole system and the way the components depend on each other.

2 Tracking the gripper

To extract a series of gripper positions from a sequence of images, the gripper has to be located in every single image. To be more precise, in every single image a certain *reference point* of

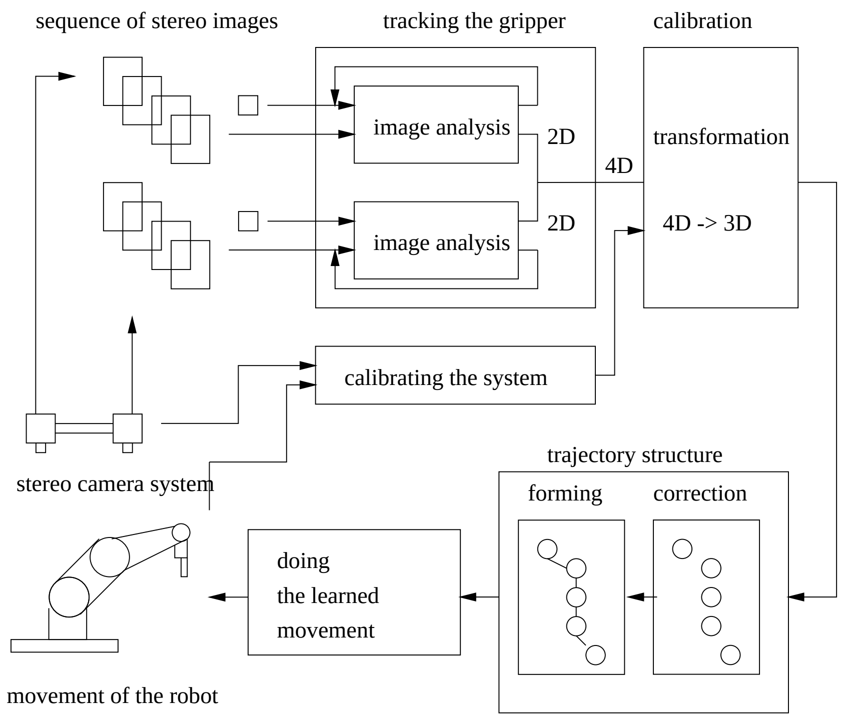

Figure 1: The components and their interaction

the gripper has to be found. Based on this precondition it is possible to consider the extracted positions as intermediate points of the movement. A two step procedure is applied to extract the reference point from the images:

(i) roughly locate final segment of the gripper by using its appearence (greylevels) from previous image

(ii) exactly determine the reference point by geometrical analysis in the selected *gripper image region* (in the following named *patch*).

In figure 2 the procedure is illustrated graphically.

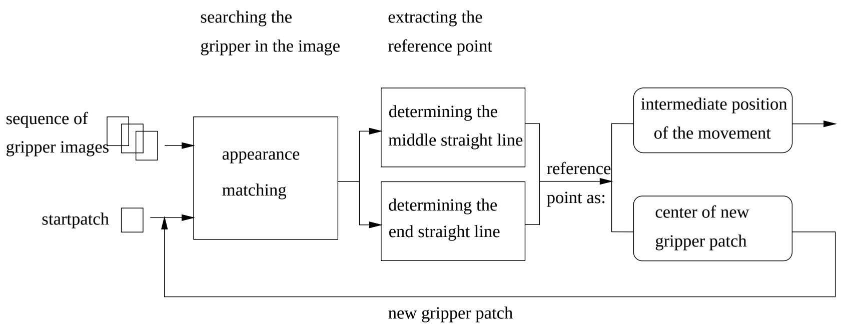

Figure 2: Tracking the gripper - a two step procedure

The input of the procedure is a sequence of stereo images representing the movement of the gripper in discrete steps. Furthermore, two image patches are supplied - one for each image sequence - which depict the appearance pattern of the gripper in the starting position of the movement. Both image sequences of the stereo cameras are analysed the same way but independently.

2.1 Locating the gripper region

The gripper image region is located by doing correlation matching using the expected gripper appearance (instead of using a model of the gripper). An $m \times m$-image B depicts the whole scene in which the robot arm is working and an $n \times n$-patch P contains the final segment of the gripper. Now a correlation image C is computed, by defining $C(k,l)$ as sum of squared distances
$C(k,l) = \sum_{\substack{i\in\left\{k-\frac{n}{2},\dots,k+\frac{n}{2}\right\} \\ j\in\left\{l-\frac{n}{2},\dots,l+\frac{n}{2}\right\}}} \left(B(i,j) - P(i-(k-\frac{n}{2}), j-(l-\frac{n}{2})\right)^2$ for each image position (k,l).
Figure 3 shows a scene, the relevant gripper image region and the resulting correlation image.

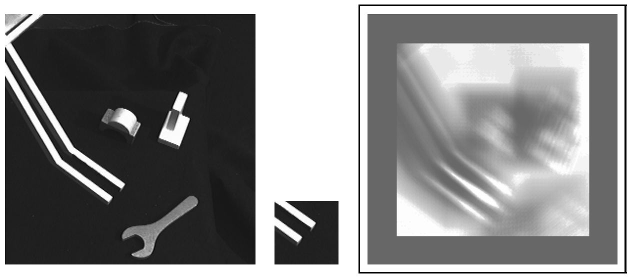

Figure 3: Scene, gripper image region and resulting correlation image (higher greylevel indicates better correlation)

The position of maximal correlation is looked for by starting the search in the position of the patch located in the previous image and expanding the catchment area (for reasons of efficiency). The position with the least sum of squared distance is used as center of the region to be cut out and used to compute the reference point in the second step.

2.2 Locating the reference point

After the $n \times n$ region of the gripper is found, the reference point has to be located as exact as possible. It will be used both for an intermediate position of the whole movement and as the center of the gripper image region for locating the gripper in the following image of the sequence. Figure 4 shows graphically the reference point of the gripper defined for this purpose.

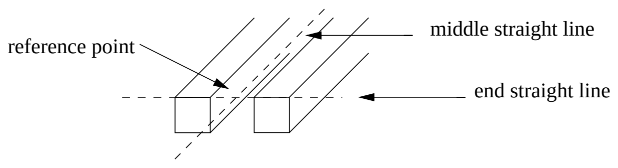

Figure 4: Position of reference point

It is the virtual point of intersection between the *middle straight line* and the *end straight line* of the robot gripper. To extract these straight lines it is necessary to first recognize the top faces of the gripper.

2.2.1 Extracting the top faces of the gripper

This is done in two different ways using simple heuristics, depending on whether an object is grasped or not. In the case of the free gripper it is assumed that the image patch can be segmented into regions as follows. The background area of the gripper is approximately homogenous and therefore can be segmented in one region, which is expected to have the largest area of all regions. Furthermore, the gripper fingers are spacely disconnected, the top faces of the two gripper fingers are homogenous and can be segmented into one region for each, and they are the second and third largest areas. Figure 5 shows exemplarily the validity of these heuristics in the case of a free gripper. However, if an object is grasped these preconditions are no longer valid and that is why another heuristic is needed. It has to be mentioned, that the scene is lighted quite well and the gripper is made of reflecting material. This guarantees the top faces to be those parts of the image with the highest greylevel. Figure 6 illustrates the validity of these heuristics in the case of a grasped object.

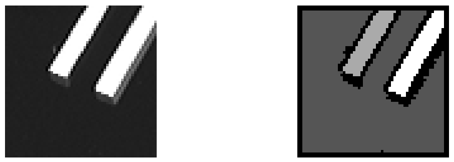
Figure 5: Patch without object and its segmentation image

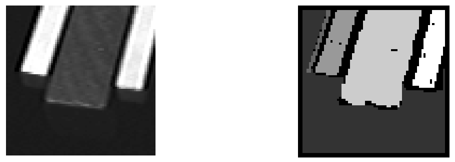
Figure 6: Patch with grasped object and its segmentation image

In other robotic enviroments specific criteria have to be found depending on the technical equipment used (e.g. a specific color of the gripper or an identity tag). Based on the segmentation result the reference point of the gripper must be computed.

2.2.2 Detecting the middle straight line

First a middle straight line is determined exactly between the extracted regions of the two gripper fingers, which come from the top faces of the gripper. Each point on this middle straight line is characterized such that the euclidean distance to both regions is equal. Alternatively, a city block metric, which computes distances only in x- respectively y-direction, works as well (see figure 7). If the distance in positiveand negative x- (y-) direction is equal this point is added to a set M of points near or on the middle straight line. The middle straight line is obtained by approximating a straight line through the points of M.

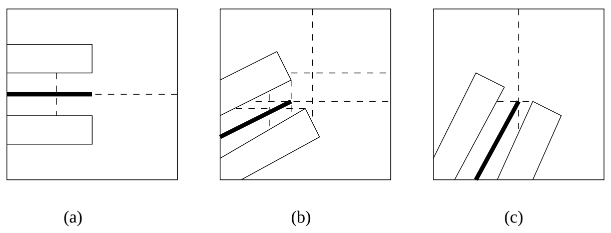

Figure 7: Determining the distance to the top faces

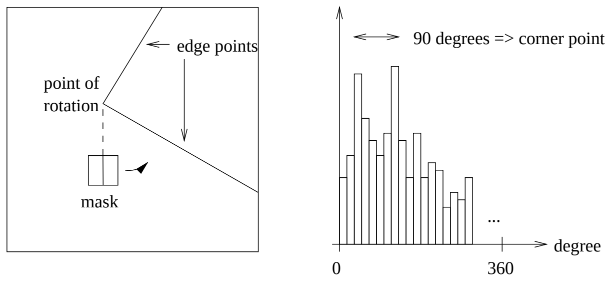

Figure 8: Rotating a squared mask to detect corners

2.2.3 Detecting the end straight line

First a set of relevant edges is extracted from the gripper image region by computing the gradient magnitudes and setting a threshold. Care must be taken with this threshold to guarantee that at least two corner points of the final segment of the two gripper fingers are in the set of edges. Thus, in a second step the relevant corners have to be detected in the binary image of edges. A binary correlation mask (see figure 8) is used having edge points at the middle vertical axis. The mask is applied at every edge of the binary gripper image and rotated in discrete steps of angles in the interval of $[0, 360]$ degrees.

For each rotation step a value is retained, which describes how many edges of the middle axis of the mask correspond to edges of the gripper. If two maximum peaks are found, being about 90 degrees apart from each other, then the rotation point belonging to that situation is taken as corner point. Only those points are considered, which are close to the top faces. The end straight line is obtained by approximating a straight line through these corner points.

3 Calibrating the camera system

Calibrating a stereo camera system means to find the relation between *image coordinates* ($2 \times$ 2D) and *world coordinates* respectively *robot coordinates*. Figure 9 shows the dependencies.

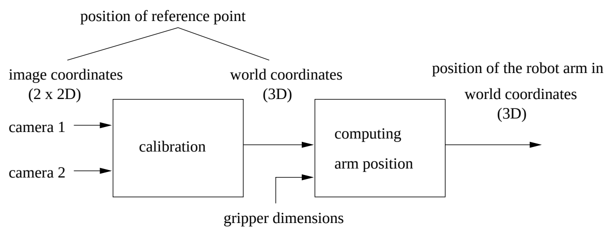

Figure 9: Transformation from 4D-image coordinates to 3D robot coordinates

To obtain the function $calib : 4D \rightarrow 3D$ we use the technique proposed in [Faugeras93]. For each camera a matrix

$$P_i = \begin{pmatrix} p^i_{11} & p^i_{12} & p^i_{13} & p^i_{14} \\ p^i_{21} & p^i_{22} & p^i_{23} & p^i_{24} \\ p^i_{31} & p^i_{32} & p^i_{33} & p^i_{34} \end{pmatrix}$$

($i = 1, 2$) is computed by using pairwise combinations of 3D world points and the 4D stereo points. The matrix is defined within the following context. Given a point in world coordinates (x_w, y_w, z_w) the position in image coordinates can be obtained by resolving

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \cdot \delta_i = P_i \cdot \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix},$$

where δ_i is an arbitrary scale factor.
However, starting with a stereo image point (x_1, y_1, x_2, y_2) a transformation is needed for computing the world coordinates (x_w, y_w, z_w). By combining P_1 and P_2 an overdetermined linear equation system ($\vec{b} = P\vec{a}$) is obtained, which is solved by using the pseudo inverse matrix

$$\vec{a} = (P^T P)^{-1} \cdot P^T \cdot \vec{b}.$$

The vector $\vec{a}$ contains the unknown 3D world coordinates and two scale factors. For the purpose of this work it is possible to compute the relation between image and robot coordinates directly without an intermediate world coordinate system. Only one certain point, namely the reference point of the gripper is used as the 3D point (w.r.t. the robot base coordinate system), and the gripper is moved in discrete steps through the working space in order to get a set of 7D-coordinates.

4 Trajectory structure

Using the calibration result the reference point of the gripper can be reconstructed into 3D robot coordinates for arbitrary positions of the gripper in the working space. Accordingly, when the

operator steers the gripper through the working space and stops at certain intermediate places, the system can compute a sequence of 3D positions $(x_1, y_1, z_1), ..., (x_n, y_n, z_n)$ extracted from the series of stereo images. Alternatively, a computer vision system could be used to find an appropriate path in the working space on the basis of stereo images, and in that case there is no need for the operator to manually steer the robot for demonstration.
This section describes how to define and use a smooth trajectory.

4.1 Definition

A trajectory structure will be defined by having in mind, that the operator demonstrated just an example of a class of congruent trajectories. That is, the starting position and orientation are a priori unknown, and it must be possible to easily include them when they become known. Thus, during the demonstration phase a trajectory structure will be constructed, which represents only the geometric relationship between intermediate points, and not the absolute positions and orientations.
The trajectory structure is a sorted sequence $(k_1, ..., k_m)$ of nodes. A node is defined by three components, which are:

- vector $(v_x, v_y, v_z) \in I\!R^3$,written as $k_i.(v_x, v_y, v_z)$,
- increment to the following node $(dx, dy, dz) \in [-1, 1]^3$, written as $k_i.(dx, dy, dz)$
- orientation $\varphi \in [0, 2\pi]$, written as $k_i.\varphi$.

Vector $k_i.(v_x, v_y, v_z)$ decribes the relation between point (x_i, y_i, z_i) and (x_1, y_1, z_1),defined as

$$k_i.(v_x, v_y, v_z) = (x_i, y_i, z_i) - (x_1, y_1, z_1) \qquad \text{for all} \;\; j \in \{1, ..., n\}.$$

The increments are given by

$$k_i.(dx, dy, dz) = k_{i+1}.(v_x, v_y, v_z) - k_i.(v_x, v_y, v_z)$$

for a node k_i $(i \in \{1, ..., n-1\})$.
The orientation φ of the gripper can be choosen free. In the implementation reported here it should keep an orthogonal orientation at every point of the trajectory.

4.2 Using the structure

To do a movement the trajectory has to be defined absolutely by incorporating the starting point (x_s, y_s, z_s) and three rotation angles $\varphi_{xy}, \varphi_{xz}, \varphi_{yz}$ (describing the rotation in the xy-, xz- and yz-coordinate-plane). The structure is moved into the starting position, which means every node is translated with (x_s, y_s, z_s), and then it is rotated as defined by $\varphi_{xy}, \varphi_{xz}, \varphi_{yz}$. Figure 10 shows a simple 2D-example.

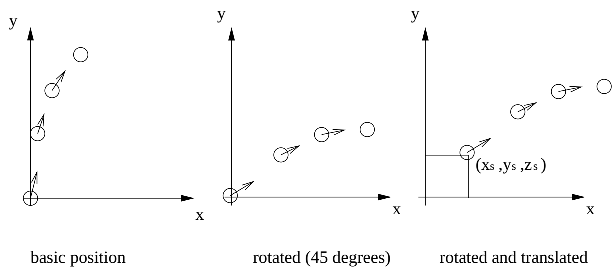

Figure 10: Moving the structure in the starting position

Starting in (x_s, y_s, z_s) the second node works as an *attractor* until it is reached. Now the next node comes into play and the procedure is repeated for all nodes of the trajectory structure. Importantly, a criterion is needed to test, whether a node is passed or not. For this a plane defined by the position of the node $k_i.(v_x, v_y, v_z)$ and the stored increment $k_{i-1}.(dx, dy, dz)$ is used. For all nodes it is tested whether this plane is passed. For a position (x_p, y_p, z_p) the difference vector $(x_d, y_d, z_d) = (x_p, y_p, z_p) - k_i.(v_x, v_y, v_z)$ and the scalar product s between (x_d, y_d, z_d) and $k_{i-1}.(dx, dy, dz)$ is $s = x_d dx + y_d dy + z_d dz$ is calculated. There are three cases resulting:

- $s > 0 \Longrightarrow -90 < |\angle((x_d, y_d, z_d), k_{i-1}.(dx, dy, dz))| < 90$
- $s = 0 \Longrightarrow |\angle((x_d, y_d, z_d), k_{i-1}.(dx, dy, dz))| = 90$
- $s < 0 \Longrightarrow 90 < |\angle((x_d, y_d, z_d), k_{i-1}.(dx, dy, dz))| < 270.$

If $s = 0$ the plane is reached and $s > 0$ means that it is passed. Figure 11 illustrates this criterion. Finally the orientation of the gripper at a certain point on the trajectory has to be found. This is done by first calculating the distances to attracting node k_i and preceding node k_{i-1} as

$$dist_v = \sqrt{(x_p - k_{i-1}.v_x)^2 + (y_p - k_{i-1}.v_y)^2 + (z_p - k_{i-1}.v_z)^2}$$

and

$$dist_a = \sqrt{(x_p - k_i.v_x)^2 + (y_p - k_i.v_y)^2 + (z_p - k_i.v_z)^2}.$$

and then getting the orientation of the gripper as weighted mean of $k_{i-1}.\varphi$ and $k_i.\varphi$:

$$\varphi' = \frac{dist_a \cdot k_{i-1}.\varphi + dist_v \cdot k_i.\varphi}{(dist_a + dist_v)}.$$

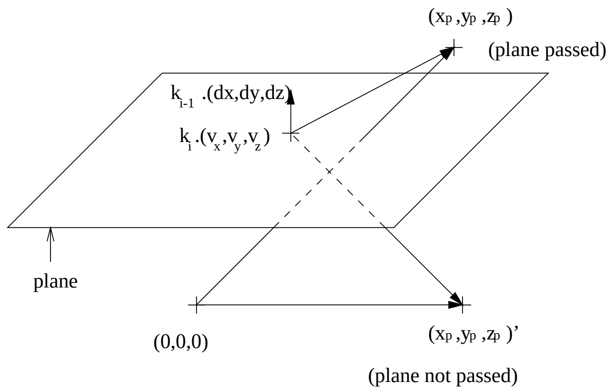

Figure 11: Testing whether a plane is passed or not

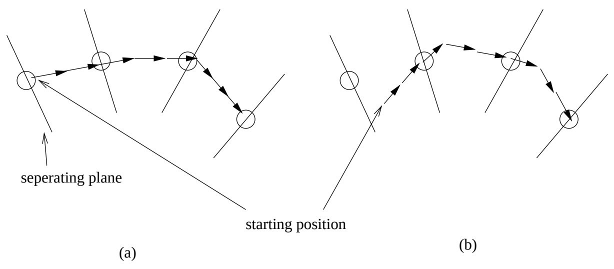

Figure 12: Example showing the resulting movements beginning in the starting point (a) or somewhere beside of it (b)

Figure 12(a) shows a simple example of the path from node to node. Further, figure 12(b) shows an interesting behaviour if a certain intermediate point can not be reached exactly, maybe due to an obstacle at that place. In this case the subsequent node attracts the gripper and thus it comes back to the original trajectory

4.3 Smoothing

Usually the positions extracted from the images are unprecise, because of resolution limit of the image or errors in calibration. To get a smooth trajectory a simple method can be used by considering the position of the neighbouring nodes. For a node k_i the position is adjusted by first calculating the distance between k_{i-1} and k_{i+1} as

$$d_1 = \sqrt{(k_{i+1}.v_x - k_{i-1}.v_x)^2 + (k_{i+1}.v_y - k_{i-1}.v_y)^2 + (k_{i+1}.v_z - k_{i-1}.v_z)^2}$$

Then

$$(x_m, y_m, z_m) = k_{i-1}.(v_x, v_y, v_z) + \frac{d_1}{2} \cdot k_{i-1}.(dx, dy, dz)$$

is the point in the middle position between k_{i-1} and k_{i+1}. The distance between $k_i.(v_x, v_y, v_z)$ and (x_m, y_m, z_m) is

$$d_2 = \sqrt{(k_i.v_x - x_m)^2 + (k_i.v_y - y_m)^2 + (k_i.v_z - z_m)^2}.$$

To do the adjustment two parameters $\varepsilon_1, \varepsilon_2 \in [0, 1]$ are defined. ε_1 defines whether a correction has to be done (e.g. in the case of large distances between points) and ε_1 says *how strong* it should be done.
With $(x_d, y_d, z_d) = (k_i.v_x - x_m, k_i.v_y - y_m, k_i.v_z - z_m)$ the new position is

$$k_i.(v_x, v_y, v_z)' = \begin{cases} k_i.(v_x, v_y, v_z) + \varepsilon_2 \cdot (x_d, y_d, z_d) & , \text{ if } \quad d_2 \geq \varepsilon_1 \cdot d_1 \\ k_i.(v_x, v_y, v_z) & , \text{ otherwise} \end{cases} .$$

This is done for every single node starting with k_2.

5 Conclusions

The proposed framework enables an operator to program a robot by just giving intermediate positions recorded by a stereo camera system. For the operator there is no need for exact knowledge of how to program the robot. Instead, it is possible to use the example trajectory in various starting position and orientation by simple translation and rotation of it. Furthermore, it is possible to involve a computer vision system, which analyses the sequence of stereo positions for an early avoidance of collisions with new objects in the scene. The ability to react on unexpected appearance of obstacles and return to the original trajectory is one of the most important features of robot systems to be used in praxis. Especially in industries like automotive productions it may help to establish a vision driven system to program and - being in use - observe mobil (e.g. see [Lin95]) and stationary robot manufactoring systems. A practical example for use may be the dismantlement of automobiles. Especially to automatically dismantle the tires certain complicated trajectories have to be learned and then translated and rotated versions of them cab be used in case of an automobile being deformed or simply moved, so that other orientation and direction is necessary to approach the tires.

References

[Faugeras93] O.Faugeras. *3-Dimensional Computer Vision.* MIT Press, Massachusetts 1993.

[Friedrich95] H.Friedrich, R.Dillmann. *Robot Programming Based On A Single Demonstration And User Intentions.* 3rd European Workshop on Learning Robots at ECML'95.

[Heise 92] R.Heise. *Programming Robots by Example.* Research Report No. 92/476/14, Department of Computer Science, University of Calgary, 1992.

[Lin95] I.S.Lin, J.Perret, F.Wallner, R.Dillmann. *Interactive Environment Modeling and Vision-Based Task Specification for a Teleoperated Mobile Robot.* 7th International Conference on Advanced Robotics (ICAR '95), Sant Feliu de Guixols, Spain.