International Symposium on Robotics, ISR 2004, Paris, France, March 2004

Spatial Control of High Speed Robot Arms using a Tilted Camera

Friedrich Lange, Gerd Hirzinger

Institute of Robotics and Mechatronics
Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)
Oberpfaffenhofen, D-82234 Weßling, Germany
e-mail: Friedrich.Lange@dlr.de
http://www.robotic.dlr.de/Friedrich.Lange/

Abstract

The paper demonstrates that when mounted on a robot endeffector a camera can be used to control the robot arm to move at high speed along a given workpiece. We use a predictive architecture that considers both the robot's dynamics and the workpiece's uncertainties of the pose or shape. To comply with applications such as glue spraying or laser cutting, the camera is mounted laterally with respect to the tool and, for reasons of visibility, tilted. This complicates the control equations. Nevertheless we manage to compute them in real time and to follow a non planar tube at a speed of 0.7 m/s. In spite of a cost effective hardware setup the mean path deviations are less than 1 mm.

1 Introduction

Usual programs of industrial robot arms require an accurate knowledge of the world. But there are potential applications in which the position and orientation – and sometimes even the shape – of the workpiece are subject to big uncertainties. Then a universal sensor is essential, which on the one hand supports high accuracy requests typically less than a millimeter, and on the other hand allows control at high speed e.g. 1 m/s. In addition, for industrial use a cost-effective hardware setup is desired.

In this paper we use a standard CCIR camera mounted on a robot with an appropriate control architecture. The camera measures the workpiece pose with respect to the tool center point (tcp) of the robot. We show two sample tasks (Fig. 1) of a robot with endeffector-mounted camera. They are used to sense (curved) lines or edges which are parallel to the desired motion. This allows to *refine* a coarsely programmed path at full speed.

According to the notation of this paper the programmed path is called *reference path* and the corresponding lines of the nominal scenario are *nominal lines*. In reality the *sensed lines* will differ from the nominal lines, thus defining the *desired path*, where a path is given by trajectories of positions and orientations.

In our experiment we program a horizontal circular path which is online modified in radial and in vertical direction using image data. A possible application is the spraying of glue to flexible or unprecisely positioned workpieces. Other applications are laser cutting, or soldering. In these cases the desired path is determined during motion, e.g. by surveying of the boundary lines of the workpiece. For all these tasks high accuracy at high speed is required.

The camera is mounted laterally to provide enough space for a tool. So with constant moving sense we have a predictive sensor as in [1]. With alternating moving sense the camera has to be tilted so that the corresponding nominal line point $\mathbf{p}_n$ comes approximately to the center of the image (see Fig. 1d). In this case segments of the lines might be occluded by the tool.

A tilted mounting yields a priori unknown distances of the different line points, more than ever as camera calibration yields small rotations too. As well, the task frame defined by the tcp and the camera frame are different.

Similar approaches have been proposed predominantly with simple scenarios [2, 1]. Other methods handle with multiple lines with pointwise given nominal location, thus allowing both translational and orientational sensor induced path corrections. However they require complex computations [3], or work at lower speed [4, 5]. The complexity comes from rotations between nominal and real world which in practice are not significant. Therefore we present a method with denies time consuming iterations to solve systems of trigonometric equations.

The paper is organized as follows: In section 2 we present a control concept that allows different sampling rates for the robot controller and the sensor. This

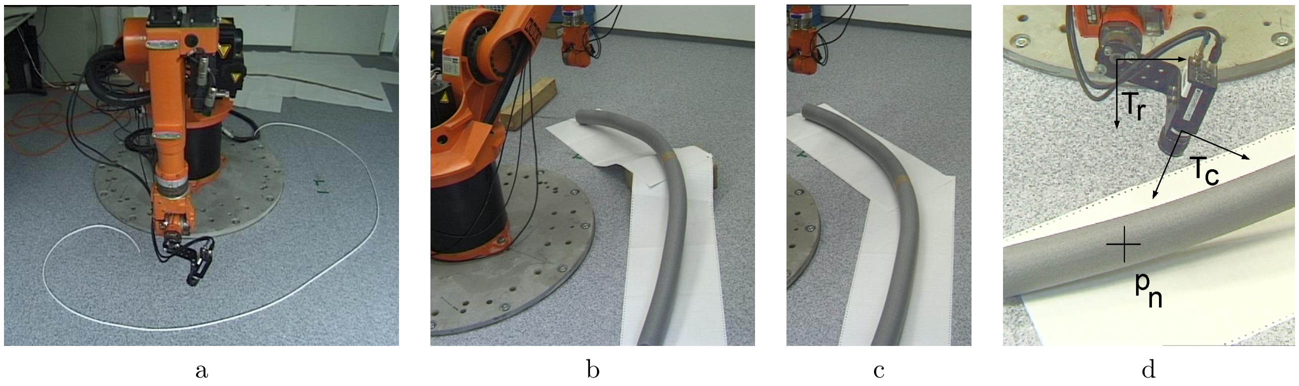

Figure 1: Sample tasks with natural and artificial background and tilted mounted camera

leads to a universal positional controller (section 3) and a task-dependent path planning algorithm (section 4). Experimental results are then demonstrated in section 5.

2 Control concept

Instead of directly using sensor data to control the robot, we use a predictive architecture which separates positional control from the sensing of the desired path (see Fig. 2). Positional control provides what we call a cartesian ideal robot. This means that for all sampling steps its arm position $\mathbf{x}_a$ is identical to the desired pose $\mathbf{x}_d$. The realization of such an ideal robot will be explained in the sequel. Sensor data affect the system by a path planning module that computes the desired poses at the sampling steps. This is presented in section 4. The fusion of both parts, positional control and path planning, yields control from image data.

This concept allows different sampling rates of the camera and the robot controller. Our approach is to integrate image data in each case in the next positional sampling step. As in [6] time delays owing to processor load are tolerable as long as the instant of exposure is known.

3 Positional control

The core of this paper is not restricted to a special positional control method. A standard industrial positional controller will do if the industrial interface allows online modified references. Such an interface, usually called sensor interface, is required since the desired positions $\mathbf{x}_d$ are not assumed to be available long before the motion.

A standard sensor interface allows to send online desired values and to receive present positional values. For the purpose of this paper we refer to the desired values of the sensor interface as a *command* vector $\mathbf{q}_c$, to distinguish between *desired positions* $\mathbf{q}_d$ (of the interface of the ideal robot) and the input to the industrial controller. The actual joint values are called *arm position* $\mathbf{q}_a$.

For curved paths to be executed at high speed we recommend to increase accuracy using advanced control methods which process not only the current desired pose but also the desired speed and acceleration or - what we use - the desired poses of multiple future sampling steps as in [7, 8].

Fig. 2 shows an adaptive feedforward controller [9] as an add-on to the standard industrial feedback controller. The proposed controller works as well in joint space since there are substantial couplings when considering robot dynamics in cartesian space. In joint space the estimation of couplings can be restricted to some additional adaptive parameters.

This feedforward controller uses a special predictive interface [9] defined either in joint space or in cartesian coordinates: In every sampling step the desired poses of the next n_d sampling steps are required, with n_d corresponding to twice the time constant of the controlled robot. This is a key aspect for the realization of an ideal robot.

Bold face lines in Fig. 2 represent data for current and future time steps, i.e. predictions of sensor data are required.

4 Path planning

The task of path planning is to determine the desired poses of next n_d sampling steps. It is advantageous to use a reference path with respect to which nominal lines are defined. Then, the task is to find the path which lies with respect to the real (sensed) lines, in the same way as the reference path to the nominal lines. This path is called *desired path* and is to be transferred to the ideal robot.

For each timestep we compute the transformation matrix ${}^{r}\mathbf{T}_d$ which describes the frame of a point of the

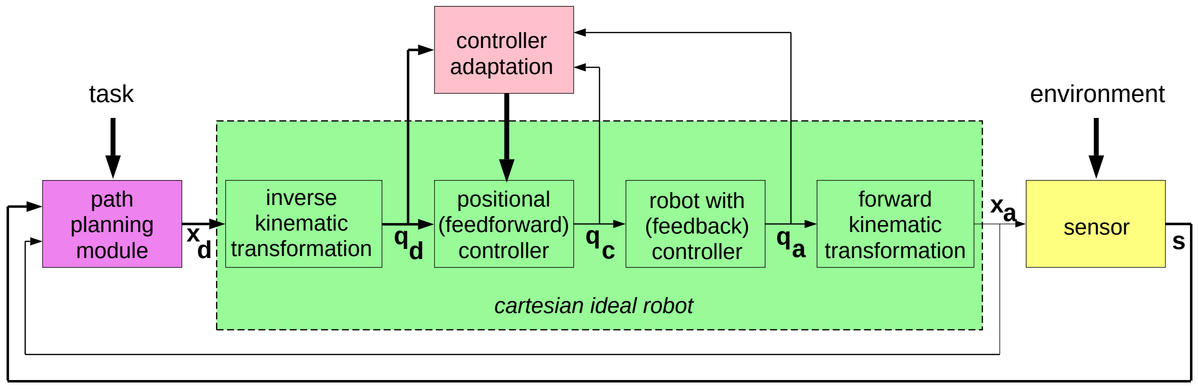
task
environment
controller adaptation
path planning module
x_d
inverse kinematic transformation
q_d
positional (feedforward) controller
q_c
robot with (feedback) controller
q_a
forward kinematic transformation
x_a
sensor
s
cartesian ideal robot

Figure 2: Architecture of control

desired path $\mathbf{T}_d$ with respect to the frame of the corresponding point of the *reference path* $\mathbf{T}_r$. The so called *sensor correction* ${}^r\mathbf{T}_d$ is computed from the postulation that the *desired path* is defined by the actually *sensed line* $\mathbf{p}_s$ in the same way as the *reference path* is given by the *nominal line* points $\mathbf{p}_n$. $\mathbf{p}$ is a point vector and, as $\mathbf{T}$, expressed in homogeneous coordinates. All these variables depend on time. The index is omitted, however, in order to simplify the notation.

This gives the fundamental equation

$$ {}^d\mathbf{p}_s = {}^r\mathbf{p}_n \tag{1} $$

and then

$$ \begin{aligned} {}^r\mathbf{p}_s &= {}^r\mathbf{T}_d \cdot {}^d\mathbf{p}_s \\ &= {}^r\mathbf{T}_d \cdot {}^r\mathbf{p}_n. \end{aligned} \tag{2} $$

Neglecting orientational sensor corrections yields

$$ {}^r\mathbf{p}_s = {}^r\mathbf{p}_d + {}^r\mathbf{p}_n. \tag{3} $$

If we have at least two lines and their nominal line positions ${}^r\mathbf{p}_{n_i}$ we can compute the components of ${}^r\mathbf{p}_d$, namely ${}^rx_d$ and ${}^rz_d$ if y is understood as the direction of motion. ${}^ry_d$ is defined to be zero since we evaluate lines points which are in the x-z-plane of the reference system. Because of the neglected orientational sensor corrections this means also

$$ {}^ry_s = 0 = {}^dy_s. \tag{4} $$

In the image the lines are detected by single points. Here we assume that the camera is oriented such that lines are approximately vertical in the image (see Fig. 3, an animated view from the camera can be started by clicking at video 1 or small video 2 [1]). We

[1]We provide some of our video clips on this CD in two sizes - the smaller ones may be preferred if motion is not smooth. Besides we recommend to use the repeat option of the player since the clips are very short.

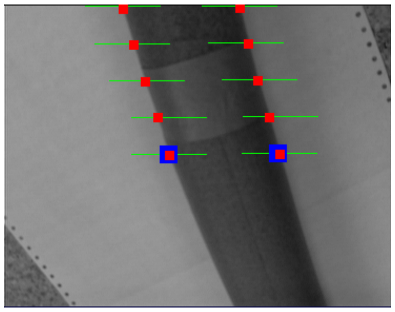

Figure 3: View from the robot mounted camera with 2 lines and 5 sensed points each. The desired positions of the lines[2] in the image are marked with big blue blocks. The windows for line search are shown by horizontal lines.

use a simple edge detection algorithm which evaluates a single image row. Horizontal line searching gives the best accuracy since it allows processing of single image fields. The detected line points are corrected by a rectification algorithm using previously identified camera and lens distortion parameters.

At least for curved paths the sensed line points will not share image row with the corresponding nominal image points. So horizontal search starting at the expected image position of a time step will give image points corresponding to the reference of different time instants. Therefore we represent the lines in the image by polynomials. All future computations are then based on these *line polynomials*

$$ \xi = f(\eta) \tag{5} $$

[2]The desired position of a line in the image is the image point of the current nominal line point when the tcp pose of the exposure is the reference pose. If these desired line positions coincide in the image with the sensed lines, the actual pose complies with the desired path.

with ξ und η as horizontal and vertical normalized image coordinates of a line point.

Using the projection equation

$$^{c}\mathbf{p}_s = \begin{pmatrix} \xi \cdot {}^{c}z_s & \eta \cdot {}^{c}z_s & {}^{c}z_s & 1 \end{pmatrix}^T , \tag{6}$$

where ${}^{c}z_s$ is the distance between the camera and the line point, we get the pose vector ${}^{r}\mathbf{p}_s$ from the reference pose of the tcp to a sensed line point

$$^{r}\mathbf{p}_s = \begin{pmatrix} {}^{r}x_s \\ {}^{r}y_s \\ {}^{r}z_s \\ 1 \end{pmatrix} = {}^{r}\mathbf{T}_c \begin{pmatrix} \xi \cdot {}^{c}z_s \\ \eta \cdot {}^{c}z_s \\ {}^{c}z_s \\ 1 \end{pmatrix} . \tag{7}$$

With equation (4) we get

$$\begin{aligned} 0 = {}^{r}y_s &= \begin{pmatrix} 0 & 1 & 0 & 0 \end{pmatrix} \cdot {}^{r}\mathbf{T}_c \cdot {}^{c}\mathbf{p}_s \qquad (8) \\ &= ({}^{r}\mathbf{T}_{c10} \cdot \xi + {}^{r}\mathbf{T}_{c11} \cdot \eta + {}^{r}\mathbf{T}_{c12}) \cdot {}^{c}z_s + {}^{r}\mathbf{T}_{c13}, \end{aligned}$$

where for example ${}^{r}\mathbf{T}_{c10}$ is the component (1,0) of the transformation matrix ${}^{r}\mathbf{T}_c$. This transformation matrix expresses the actual pose of the camera with respect to the reference pose of the tcp. At least for curved reference paths or a camera which is tilted with respect to the tool or the lines (see Fig. 1d) this transformation differs from identity.

The equations for ${}^{r}x_s$ and ${}^{r}z_s$ are a little bit more complicated to evaluate. With $\mathbf{T}_r$ and $\mathbf{T}_d$ having the same orientation, using equation (3) we can write

$$^{r}x_s = {}^{r}x_d + {}^{d}x_s = {}^{r}x_d + {}^{r}x_n, \tag{9}$$

where ${}^{r}x_n$ is the nominal distance of the line with respect to the reference trajectory of the tcp. This distance is given, e.g. as the space between the desired cutting edge and a visible edge on the workpiece. ${}^{r}x_d$ is the wanted path correction which is the same for different lines:

$$^{r}x_d = ({}^{r}\mathbf{T}_{c00} \cdot \xi + {}^{r}\mathbf{T}_{c01} \cdot \eta + {}^{r}\mathbf{T}_{c02}) \cdot {}^{c}z_s + {}^{r}\mathbf{T}_{c03} - {}^{r}x_n. \tag{10}$$

By comparing ${}^{r}x_d$ of two lines we get

$$\begin{gathered} ({}^{r}\mathbf{T}_{c00} \cdot \xi_0 + {}^{r}\mathbf{T}_{c01} \cdot \eta_0 + {}^{r}\mathbf{T}_{c02}) \cdot {}^{c}z_{s_0} + {}^{r}\mathbf{T}_{c03} - {}^{r}x_{n_0} \\ = \\ ({}^{r}\mathbf{T}_{c00} \cdot \xi_1 + {}^{r}\mathbf{T}_{c01} \cdot \eta_1 + {}^{r}\mathbf{T}_{c02}) \cdot {}^{c}z_{s_1} + {}^{r}\mathbf{T}_{c03} - {}^{r}x_{n_1}. \end{gathered} \tag{11}$$

Likewise we compute

$$^{r}z_d = ({}^{r}\mathbf{T}_{c20} \cdot \xi + {}^{r}\mathbf{T}_{c21} \cdot \eta + {}^{r}\mathbf{T}_{c22}) \cdot {}^{c}z_s + {}^{r}\mathbf{T}_{c23} - {}^{r}z_n \tag{12}$$

and thus

$$\begin{gathered} ({}^{r}\mathbf{T}_{c20} \cdot \xi_0 + {}^{r}\mathbf{T}_{c21} \cdot \eta_0 + {}^{r}\mathbf{T}_{c22}) \cdot {}^{c}z_{s_0} + {}^{r}\mathbf{T}_{c23} - {}^{r}z_{n_0} \\ = \\ ({}^{r}\mathbf{T}_{c20} \cdot \xi_1 + {}^{r}\mathbf{T}_{c21} \cdot \eta_1 + {}^{r}\mathbf{T}_{c22}) \cdot {}^{c}z_{s_1} + {}^{r}\mathbf{T}_{c23} - {}^{r}z_{n_1}. \end{gathered} \tag{13}$$

where ${}^{r}z_{n_0}$ and ${}^{r}z_{n_1}$ are the nominal distances to the lines, as e.g. the distance between the laser and the workpiece. Usually the distances to the two lines are the same.

In the case of two lines, with two equations (8), equation (11), equation (13), and two equations (5) we have a total of six equations to determine $\xi_0, \xi_1, \eta_0, \eta_1, {}^{c}z_{s_0}$, and ${}^{c}z_{s_1}$. Because of the nonlinearity, equation (5), a numerical solution is required which usually converges within 2 iterations. The wanted components ${}^{r}x_d$ and ${}^{r}z_d$ of the path correction are calculated by inserting the computed variables into the equations (10) and (12), using any of the lines.

If only one line is visible, we need a priori information, e.g. the distance of the line or, more precisely, the plane in which the line lies. If this plane is parallel to the x-y-plane of the tool frame we can use ${}^{r}z_d = 0$. In this case the system of equations is limited to equations (5), (8), and (12) to determine ξ, η, and ${}^{c}z_s$ which are inserted into equation (10).

Strictly speaking we use a priori information as well with two lines. It is the assumption that the plane of the lines is parallel to the x-y-plane of the tool. A varying distance can be interpreted as a non zero roll angle. So the specification is restricted to the pitch angle of the plane, i.e. the rotation around the y-axis of the reference frame of the tcp. The computation of this value needs at least three lines [3].

For every capture we compute path corrections $\mathbf{T}_r$ for multiple reference poses but fixed camera pose $\mathbf{T}_c$ which is the pose at the time instant of the exposure. To avoid the computation for all n_d sampling steps (see section 3), we represent the path corrections also as polynomials, using again parameter estimation methods. The resulting polynomials allow to readout the wanted path modifications with minimal effort. Therefore the method is still suitable for $n_d \approx 20$. With an appropriate feedforward control it yields minimal path errors.

The indirect computation of the desired poses by using path modifications with respect to a reference path is advantageous since curved paths with varying orientation are allowed. Solely the path correction itself is assumed to be done without rotations.

5 Experiments

As first experiment we consider a bent tube that has to be followed at a speed of 0.7 m/s by a KUKA KR6/1 robot (see Fig. 1b and c, and video 3 or small video 4) using the industrial controller KRC1. Image processing, i.e. detection of the boundary lines of the tube, runs in field mode of the camera at a rate of 50 Hz. This is asynchrounous to the control rate of 83 Hz.

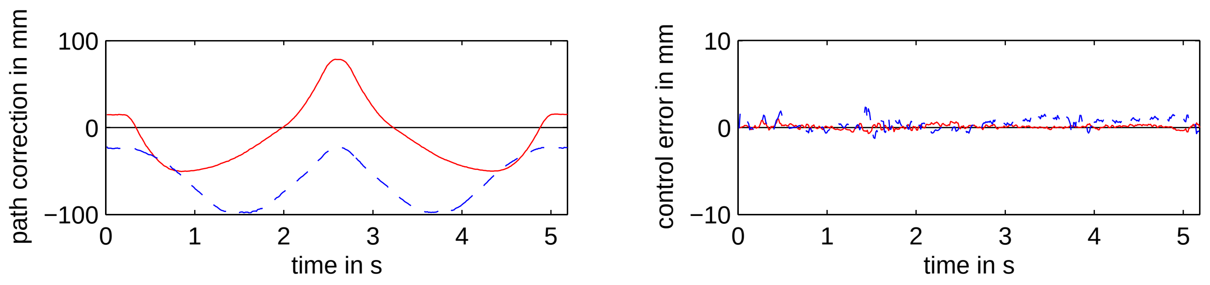

Figure 4: Sensor induced path correction and control error when following a tube in space (solid = horizontal, dashed = vertical, note the different scales)

Previously identified camera reconstruction errors are compensated, lens distortion among others [10]. All tasks for robot control and image processing run in parallel on a 400 MHz processor of the industrial controller KRC1. Control uses the operating system VxWorks and vision runs under Windows95 which is one of the VxWorks tasks. Thus the additionally required hardware is limited to a camera and a standard frame grabber.

	1 line	2 lines
mean pixel error	1.2 pixel	1.4 pixel
max. pixel error	2.9 pixel	4.4 pixel
mean. hor. path error	0.3 mm	0.3 mm
max. hor. path error	1.0 mm	1.0 mm
mean vert. path error	-	0.9 mm
max. vert. path error	-	2.4 mm
	small video 7	video 1 or small video 2

Table 1: Path error when following a bent tube with 0.7 m/s evaluating in the horizontal plane (1 boundary line) or in space (2 boundary lines) respectively

Table 1 displays the reached accuracy. When tracking only one boundary line (Fig. 1c), the tube lies horizontally on the floor. In this case, in spite of big differences between nominal and actual tube, a mean pixel error of about 1 pixel is reached even without filtering of the edge data.

With a non planar layout (Fig. 1b) we evaluate both boundary lines to determine a spatial path. The vertical accuracy is inferior because on the one hand the measuring of the distance is ill-conditioned due to the geometrical setup, and on the other hand because elastic oscillations of the robot are excited. Compliance in the robot joints prevent accurate measurements of the camera pose, which influences the detected line poses. Nevertheless the mean control error in both degrees of freedom (dof) falls below one millimeter (Fig. 4). The reached path accuracy equals thus the complex method of [3].

As a second experiment the robot has to follow, as well at 0.7 m/s, a light cable which lies on the ground (see Fig. 1a, and video 5 or small video 6). This demonstrates that, in spite of the structured floor, there is no artificial background required. Path corrections are horizontal only, and computed by different methods. Because of bigger deviations from the reference path, path errors (see table 2) are somewhat bigger than in the previous experiment. This is caused by approximation errors between the polynomials and the real shape of the line. Besides, exposure instant uncertainties will cause additional errors.

The errors of the method of this paper are listed in the right-hand column. They are compared, first, with a method which, using the same control architecture, works without the predictive interface of the positional controller, and therefore without feedforward control. So with this method in the inner control loop, only the standard feedback controller is used. In contrast, the path planning module is unchanged. Depending on the shape of the line, only a small increase of the path errors appears with this setup. This is because although the positional control is not predictive, the vision system still uses the upper and lower regions of the image, thus allowing predictions of the desired path.

A further alternative is a classical visual servoing algorithm which only evaluates the location of the line in the center of the image, i.e. without any prediction. Control is implemented using a coarsely optimized PD algorithm for the cartesian signal in x direction of the reference system. This controller ignores the camera positions.

$$\begin{aligned} {}^r x_{cx}(k) &= {}^r x_{cx}(k-1) \\ &\quad + K_P \cdot {}^r e_x(k) + K_D \cdot ({}^r e_x(k) - {}^r e_x(k-1)) \end{aligned} \tag{14}$$

	without use of sensor data	visual servoing with PD controller without prediction	positional controller without feedforward with prediction	positional controller with feedforward with prediction
mean pixel error	-	35 pixel	1.9 pixel	1.3 pixel
maximum pixel error	-	63 pixel	4.7 pixel	3.6 pixel
mean path error	95 mm	9.7 mm	0.8 mm	0.6 mm
maximum path error	157 mm	19.9 mm	2.4 mm	1.5 mm
		small video 8		small video 9

Table 2: Path error when following a cable using different methods

Because of the tilted mounted camera (see Fig. 1d), a control error of

$$\begin{aligned} {}^r e_x &= ({}^r z_n - {}^a\mathbf{T}_{c23}) \cdot \frac{{}^a\mathbf{T}_{c00}\cdot\xi + {}^a\mathbf{T}_{c01}\cdot\eta + {}^a\mathbf{T}_{c02}}{{}^a\mathbf{T}_{c20}\cdot\xi + {}^a\mathbf{T}_{c21}\cdot\eta + {}^a\mathbf{T}_{c22}} \\ &+ ({}^a\mathbf{T}_{c03} - {}^r x_n) \end{aligned} \tag{15}$$

is evaluated. ${}^a\mathbf{T}_{cij}$ are the elements of the (constant) tranformation matrix between tcp and camera. In this experiment the performance is poor, and without limitation the robot would leave the allowed range of accelerations.

6 Conclusion

The article shows that vision systems are also applicable for tracking tasks with high robot speed. Even accurate control of the robot is possible. The crucial fact is the predictive image evaluation, maybe in combination with an adaptive positional feedforward control.

The results are reached by means of a cost effective method for which additional hardware is limited to standard components. In order to guarantee low costs, image processing, computation of the desired path, and dynamical control are executed using only the standard processor of an industrial robot controller, in our case the KUKA KRC1.

The method is demonstrated in a task where the location and the shape of a curved robot path is modified in 2 dof during motion. In addition to such applications with coarsely known workpieces, online measuring of a target position can accelerate pick-and-place tasks or basic assembling.

Future work will improve the measurements of the camera pose e.g. using an observer, thus avoiding oscillations caused by joint elasticities.

References

[1] Dr. Barthel Sensorsysteme. SCOUT joint tracking system. http://www.scout-sensor.com/.

[2] F. Lange and G. Hirzinger. Is vision the appropriate sensor for cost oriented automation? In R. Bernhardt and H.-H. Erbe, editors, *Cost Oriented Automation (Low Cost Automation 2001)*, Berlin, Germany, October 2001. Published in IFAC Proceedings, Elsevier Science, 2002.

[3] F. Lange and G. Hirzinger. Predictive visual tracking of lines by industrial robots. *The International Journal on Robotics Research*, 22(10-11):889–903, Oct-Nov 2003.

[4] J. A. Gangloff and M. F. de Mathelin. Visual servoing of a 6-dof manipulator for unknown 3-d profile following. *IEEE Trans. on Robotics and Automation*, 18(4):511–520, August 2002.

[5] P. Rives and J.-J. Borrelly. Real-time image processing for image-based visual servoing. In M. Vincze and G. D. Hager, editors, *Robust vision for vision-based control of motion*, pages 99–107. IEEE Press, 2000.

[6] J. Zhang, R. Lumia, J. Wood, and G. Starr. Delay dependent stability-limits in high performance real-time visual servoing systems. In *Proc. IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pages 485–491, Las Vegas, Nevada, Oct. 2003.

[7] D. W. Clarke, C. Mohtadi, and P. S. Tuff. Generalized predictive control - part I. the basic algorithm. *Automatica*, 23(2):137–148, 1987.

[8] M. Grotjahn and B. Heimann. Model-based feedforward control in industrial robotics. *The International Journal on Robotics Research*, 21(1):45–60, January 2002.

[9] F. Lange and G. Hirzinger. Learning of a controller for non-recurring fast movements. *Advanced Robotics*, 10(2):229–244, April 1996.

[10] http://www.robotic.dlr.de/VISION/Projects/Calibration/CalLab.html, 1999.