1998 IEEE International Conference on Robotics and Automation
Leuven, Belgium, May 1998

# Predictive Vision Based Control of High Speed Industrial Robot Paths

Friedrich Lange Patrick Wunsch Gerhard Hirzinger

DLR Institute of Robotics and System Dynamics
P. O. Box 1116, D-82230 Wessling, Germany
e-mail: Friedrich.Lange@dlr.de

## Abstract

*A predictive architecture is presented to react on sensor data in the case of high speed motion and low bandwidth sensor data. This concept is used for the vision based control of an industrial robot (KUKA) to track a contour at a speed of 1.6 m/s. The vision task can be performed very fast since only 2 rows of the image are analyzed. In this way an accuracy of 0.3 mm is reached in spite of uncertainties in the robot's kinematic parameters. Vision and control work asynchronuously so that even delay times are tolerable during sensing as long as the time-instant of the exposure is known.*

## 1 Introduction

It is well known that sensor control of robots is limited by the bandwidth of both signal processing and robot dynamics. Therefore, regardless of the sensor system, tracking of unknown contours requires low speed. This may be one reason why in industrial applications online sensor control predominantly has been avoided.

However a camera can provide more information than the current deviation from a nominal position at the sample instant. It can also provide some look-ahead along the desired trajectory. This property can be exploited to overcome the bandwidth limitation of traditional vision based control to execute high speed sensor controlled movements. This paper presents such an approach.

In contrast to other visual servoing problems [HHC96] we do not track a target but we follow an edge with predetermined speed. This is motivated by industrial applications where the desired path is only inaccurately known because of tolerances of the workpiece dimensions or an uncertain positioning of the workpiece. A typical example is a robot which has to distribute the glue for sticking a sealing strip along an edge at the door of a car.

In addition we assume that the robot's tool center point (TCP) is only roughly known due to uncertain kinematic parameters.

In this paper, the task is to control a 6-axis robot with an endeffector integrated camera to follow a planar edge which is known coarsely (see figure 1). The prespecified path is only used to define the velocity profile. Information about the path geometry is derived from the sensor values. So the camera is used to detect displacements normal to the edge direction.

In general we see 3 steps to attain accurate tracking during high speed sensor controlled movements:

1. Localization of the desired position with respect to the actual position and the coordinate system (joint angles) of the robot.
2. Prediction of future steps of the desired path.
3. Control of the desired path in spite of disturbances due to the robot dynamics.

Step 1 means the evaluation of the sensor device to obtain the transformation between the actual and

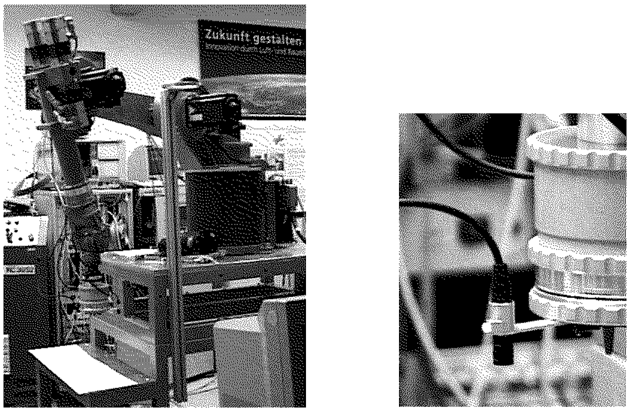


Figure 1: Experimental setup with endeffector mounted camera

the desired position. For vision based sensors features are extracted from an image. Then the corresponding positions are calculated using the known robot and sensor characteristics. Unknown kinematics require learning of the hand-eye-transformation (see e.g. [vdSG97, JFN97]). In industrial applications this is not a problem since the kinematic parameters are known and the deviations between the tool center point and the sensed desired position will be small. So step 1 is a standard problem which will not be outlined here (see e.g. [HHC96]).

Step 2 requires the localization of future desired positions as in step 1 for the current timestep. So the robot motion along the contour has to be mapped into the image. Then the sensed desired motion has to be transformed back to the robot coordinate system. In this case the transformation is not trivial since small uncertainties in the kinematic parameters of the robot or the camera produce orientational errors which yield significant positional differences. The problem arises from the fact that the position has to be determined with an accuracy of about .001 of the distance to the current TCP. So this step will be discussed in section 3.

Step 3 means the compensation of dynamical path errors. This problem has been solved for position controlled robots in [LH94]. The learned feedforward controller requires the desired joint values for the current and $n_d$ future controller timesteps. This corresponds to a prediction of the desired path over at least a period equivalent to the time constant of the robot, the motor-drives, and the feedback control loops. These future desired positions are obtained by step 2.

Known approaches for high speed tracking of a target [Cor95, CG96] use feedforward control as well, exploiting only estimations about the target speed in contrast to the approach taken here, where feedforward control will be based on learned dynamic characteristics of the robot.

## 2 Architecture for accurate control of robots with positional interface

Figure 2 shows the proposed architecture. Thin lines mean positions, e.g. the commanded and the actual position of the robot at timestep $k$. In contrast thick lines mean trajectories, e.g. the desired path from timestep $k$ to timestep $k + n_d$.

The vision system can not only measure the current control error, i.e. the difference between the actual position and the edge. In addition it can detect a spatially extended part of the desired path. So at

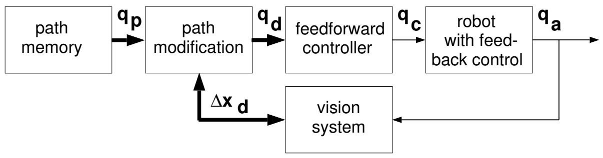


Figure 2: Indirect control architecture

timestep $k$ the following $n_d$ timesteps of the originally programmed path can be modified as well. This allows the feedforward controller with the learned parameters $r_i$ to output a commanded position

$$\begin{aligned} q_c(k) = q_d(k) + r_1 \cdot (q_d(k+1) - q_d(k)) \\ + \cdots + r_{n_d} \cdot (q_d(k+n_d) - q_d(k)) \end{aligned} \tag{1}$$

which in combination with the robot feedback control loops yields accurate following of the desired path $q_d$. The learning procedure is shown in [LH94]. It can be performed in advance without any sensor feedback.

Please note that all positions $q$ are expressed in the coordinate system of the robot, the joint space. Cartesian measurements $x$ have to be transformed therefore. Thus at timestep $k$ the path modification module calculates the modified desired path as

$$q_d(k+i) = inv\_kin[kin[q_p(k+i)] + \Delta x_d(k+i,k)] \tag{2}$$

or

$$q_d(k+i) = inv\_kin[x_p(k+i) + \Delta x_d(k+i,k)] \tag{3}$$

from the original desired path $q_p$ or $x_p$ and the cartesian difference $\Delta x_d(k+i,k)$ between the programmed and the sensed desired path at timestep $k+i$, measured at timestep $k$. $kin[.]$ and $inv\_kin[.]$ denote the forward and inverse kinematic transformation, respectively.

The path modifications $\Delta x_d$ are calculated from the actual position $x_a$ (which is transformed from $q_a$), the sensor values $\Delta x_s$, and the nominal edge positions $x_n$ (see figure 3). This approach allows the control of (curved) paths which do not coincide with an edge.

The positions $x$ are expressed in the cartesian sensor coordinate system which is chosen such that the robot moves along for instance the $y$-direction. So the $y$-direction is not sensor controlled. However deviations due to the robot dynamics are compensated in all components.

Hence,

$$\Delta x_d(k+i,k) = \begin{pmatrix} \Delta x_{dx}(k+i,k) \\ 0 \\ \Delta x_{dz}(k+i,k) \\ \vdots \end{pmatrix} \tag{4}$$

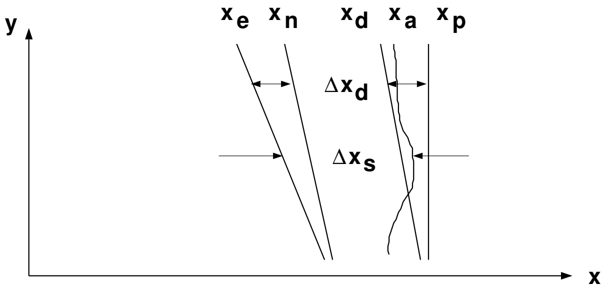


Figure 3: Notation of different paths ($x_p$ = original desired path, $x_d$ = sensed desired path, $x_a$ = actual path, $x_e$ = actual edge, $x_n$ = nominal edge, $\Delta x_s$ = sensor value, $\Delta x_d$ = modification of the desired path due to a difference between the nominal and the actual edge)

In this notation indices $x$, $y$ or $z$ as second indices represent the corresponding components of the vector.

At timestep $k$ the sensor values $\Delta x_s(k+i,k)$ are taken which look at the positions $x_p(k+i)$ by measurements from the current position $x_a(k)$. Then $\Delta x_d(k+i,k)$ can be revised by

$$\Delta x_d(k+i,k) = x_a(k) + \Delta x_s(k+i,k) - x_n(k+i) \quad (5)$$

The generation of sensor values $\Delta x_s(.)$ from the camera data will be explained in section 3. The nominal edge position $x_n(.)$ is given. For an edge parallel to the $y$-axis $x_n$ is constant.

In contrast to direct feedback of the sensor values (see [HHC96]) the separation of sensor evaluation and control is insensitive to delays during the signal processing or the execution of the positional commands. This superiority with respect to stability has already been discussed in [LH96], there concerning indirect force control. It allows high bandwidth of the desired motion in spite of low bandwidth sensing which is typical for vision based systems. The only requirement is that $x_a$ and $\Delta x_s$ are measured at the same time. This will be studied in subsection 3.3.

For very low vision bandwidth the sensed deviations may become so big that instantaneous return to the desired path exceeds the allowed accelerations of the robot. In that case it is not sufficient to modify some sampled values of the desired path. Instead, a new trajectory has to be generated which leads the robot from the actual state (position and velocity) to the desired path. Such a path planning module is required for large time delays as in [Hir93] or immeadiately after the sensor control loop has been started. The experiments of section 4 were chosen such that explicit path planning was not necessary.

# 3 Vision based prediction of future control errors

## 3.1 General setup

This section explains the determination of the $n_d$ sensor values $\Delta x_s(k+i,k)$ for the timesteps $k+i$ from the camera data of timestep $k$. It is assumed that the $x$-component of the sensor coordinate system is the only component to be controlled by the sensor. In this simple case, a single camera that tracks one single edge provides sufficient information. Additional sensor-controlled DOFs require an extended sensor system, e.g. a stereo camera system or a second edge which can be tracked. The $x$-component corresponds to the horizontal image coordinate $u$ whereas the $y$-component, i.e. the direction of the fast motion, affects the vertical image coordinate $v$.

The camera images are filtered by a $3 \times 3$ matrix to extract vertical edges (figure 4), which allows robust localization of the edge pixels.

Then two image rows $v_0$ and $v_1$ (orthogonal to the edge) are selected in which the positions $u_0$ and $u_1$ of the edge have to be determined. These rows correspond to the TCP (more specific, the center of the sensor coordinate system) at the current timestep $x_{ay}(k)$ (see "o" in figure 4) and to a future programmed position $x_{py}(k+n_d)$ (see "+" in figure 4) if the visual angle is big enough. This means that the choice of the rows is dependent on the programmed path. The rows are computed as

$$v_0(k) = f_y^{-1}(x_{ay}(k) - x_{ay}(k)) = f_y^{-1}(0), \qquad (6)$$

and

$$v_1(k) = f_y^{-1}(x_{py}(k+n_d) - x_{ay}(k)), \qquad (7)$$

with extra consideration for the periphery of the image. So the "+" in the left part of figure 4 corresponds to the maximal value whereas the "+" on the

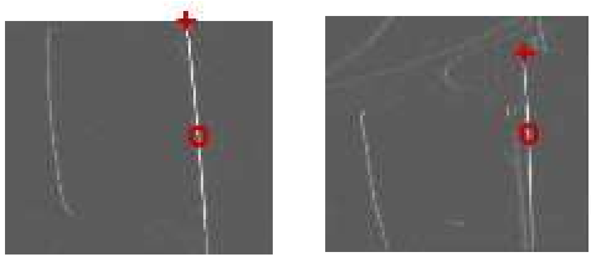

Figure 4: Image filtered for extraction of vertical edges (left shot: in the middle of the path, right shot: near the end of the path, **o** = edge at current TCP, + = edge at future desired TCP)

right hand side marks the end of the path which at the same time is the end of the vertical edge.

The image transformations $f_x(.)$ and $f_y(.)$ describe the mapping from pixels to positions which is defined by a calibration motion which is executed at the beginning of the path. This mapping is time-invariant if the distance between the camera and the edge is approximately constant in time. It is linear if the visual angle is small enough i.e. if the distance between the camera and the different edge positions is almost the same. For cameras with a large visual angle lens distortions may become significant if the edge is distant from the center of the rows. Then camera calibration methods as in [WCH92] have to be used to get at least $f_x(.)$. This is required when tracking different edges for sensor control of multiple degrees of freedom.

The limitation of the interpretation of the image with only two rows allows to compute the sensor values within 20 ms for each image (50 images per second) even with inexpensive standard vision hardware, as computation intensive edge filtering can be limited to the two selected rows.

The localization of the edge at the current and the future position are used to calculate a linear approximation of the edge between these two positions. This finally yields the $n_d$ sensor values $\Delta x_{sx}$ which are required in equation (5). Thus

$$\begin{aligned}\Delta x_{sx}(k+i,k) &= f_x(u_0(k)) \\ &+(x_{py}(k+i)-x_{ay}(k))\cdot\frac{f_x(u_1(k))-f_x(u_0(k))}{f_y(v_1(k))-f_y(v_0(k))} \\ &= a(k)+(x_{py}(k+i)-x_{ay}(k))\cdot b(k) \qquad (8)\end{aligned}$$

The number $n_d$ of calculated edge positions is between 10 and 20 for usual industrial robots. But the edge is computed by a linear approximation from two points in the image. So the number of points to be localized in the image might have to be increased for edges with high curvatures as long as this does not change the image sampling rate.

Originally, the focal length of the camera system should allow a prediction length of $n_d$ timesteps even at full speed. However this yields a very coarse resolution of the edge position. So a compromise has to be found. In this sense it is advantageous to tilt the camera towards the direction of motion for a fine resolution nearby and a large visual range.

### 3.2 Calibration

Equation (8) is sufficient if the kinematic parameters of the robot and the sensor are exactly known and the distance of the edge is time-invariant. Strictly speaking, the kinematic parameters of the TCP should not point to the actual robot's TCP but to the point where the optical axis of the camera systems hits the plane in which the edge lies. Otherwise a tilted end-effector would give an error in $\Delta x_d$.

If the kinematic parameters are not exactly known, calibration is required since at least orientational errors $\phi$ within the plane of the edge lead to big errors of the sensor values of $v_1$. In contrast to the detection of $f_x(.)$ and $f_y(.)$ (see sect. 3.1) the calibration of the orientation has to be repeated online because the effect of uncertain kinematic parameters or badly referenced joints is variable in space. This means that no extra motion is tolerable.

This calibration is done by comparing two positions from the real motion, measured by the joint values $q_a$, and from the observed motion which is tracked in the image.

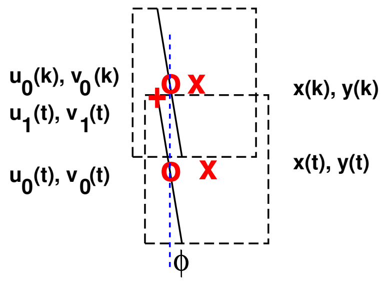


Figure 5: Calculation of the sensor orientation $\phi(k)$ from two images at time instants k and t (x = TCP, o = edge at TCP, + = edge at future position)

The first task for the calibration is to find the timestep $t$ in which the part of the edge at the future position $x_{ay}(t)+f_y(v_1(t))$ corresponds to the current position $x_{ay}(k)+f_y(v_0(k))$ at timestep k (see figure 5).

The orientational error of the image is calculated as the difference between the slope $b$ of the edge as it has been sensed at time-instant $t$ (lower left solid line) and the actual slope which is represented by the difference of the current edge positions in the two images (dotted line). This approach assumes that the measurements $u_0, v_0$ of the current edge positions are not affected by small orientational errors.

The rotation $\phi(k)$ between the real image and the nominal sensor coordinate system is then

$$\begin{aligned}sin(\phi(k)) &= \frac{x_{ax}(k)+f_x(u_0(k))-x_{ax}(t)-f_x(u_0(t))}{x_{ay}(k)+f_y(v_0(k))-x_{ay}(t)-f_y(v_0(t))} \\ &- \frac{f_x(u_1(t))-f_x(u_0(t))}{f_y(v_1(t))-f_y(v_0(t))} \qquad (9)\end{aligned}$$

Beyond that, if the kinematic transformation of $q_a$

calculates changes in the orientation of the camera, $x_{a\phi}(k) - x_{a\phi}(t)$ has to be added to $\phi(k)$.

So equation (8) is changed to

$$\Delta x_{sx}(k+i,k) \quad (10)$$
$$= a(k) + (x_{py}(k+i) - x_{ay}(k)) \cdot (b(k) + sin(\phi(k)))$$

This means that $\Delta x_{sx}$ is expressed in the nominal sensor coordinate system and not in the real sensor system.

### 3.3 Fusion of asynchronuous sensor data

Equation (5) requires measurements $x_a$ and $\Delta x_s$ to be simultaneous. In practice however, the robot and the vision system have their own time bases and different sampling rates. In addition, delays may occur during sending of vision data to the robot controller since in our experiments we use a non-dedicated ethernet-based network to communicate measurements to the robot.

Therefore the communication is organized in a bidirectional fashion. First, the robot controller sends a request which by the way includes the arguments 0 and $x_{py}(k+n_d) - x_{ay}(k)$ of equations (6) and (7). When this request reaches the vision system, the next image will be taken for computation. So the image is at most one frame of 20 ms plus 20 ms for the exposure older than the request. After the computation of the sensor values the data are sent to the robot controller which then examines if the data are valid. Two tests are proposed when the data are received by the robot controller in timestep $l$:

- The sensor values are not more accurate than the difference between the positions $x_{ax}(l)$ and $x_{ax}(k - 40ms)$. So for big positional changes the sensor values have to be discarded and a new request has to be sent. The choice of a suitable limit for this turns out to be difficult since high bandwidth sensor values may cause big changes in the $x$-component of the robot position. Strictly speaking not the component $x$ has to be constant but the component normal to the edge.
- The time difference $l - k$ between request and reception of data has to be limited to a maximum of about 40 ms to prevent accidentally small differences between $x_{ax}(l)$ and $x_{ax}(k - 40ms)$ (e.g. for paths with high curvature).

If a set of sensor data is valid, equation (10) is evaluated with $k-2$ instead of $k$ and $l+i$ instead of $k+i$, since the image which has been taken about 20 ms $\approx 2 \cdot 12$ ms before timestep $k$ is used to predict the edge for $i$ steps after the current timestep $l$. If no valid sensor data is available, the edge positions are predicted as well from equation (10) with $k$ according to the the last valid sensor data.

So a delayed or missing dataset is no problem for the stability of the servo loop since the control loop is not concerned. In the same way large delay times can be considered and, apart from the fact that the information arrives later, do not affect the performance.

## 4 Experiments

For the experiments a straight line of 80 cm is programmed. It decribes coarsely the edge (see figure 1). The path is executed by the robot with a maximal speed of 1.6 m/s. It is sensed by an endeffector mounted camera in a distance of about 30 cm from the edge.

Figure 6 shows the result using a straight edge. The programmed line (solid curve with an RMS deviation of 10 mm) turns out to be far away from the edge at the end of the path. Sensor control according to equations (1), (2) and (5), but without the prediction of equation (8) (dashed curve with an RMS deviation of 3.4 mm) can reduce the path error substantially but not totally. The errors come from fact that the actual edge position is sensed too late. Normal visual servoing with a specially tuned PD-type controller according to [HHC96] (dash-dotted curve with an RMS-value of 1.9 mm) reduces the path error somewhat better. Here, the accuracy is futher limited by the dynamical delays of the robot which affect the different joints differently. Both types of errors can be compensated by the predictive control approach with online calibration of the orientation (dotted curve with an RMS deviation of 0.3 mm). In contrast to the slope of the edge $b$

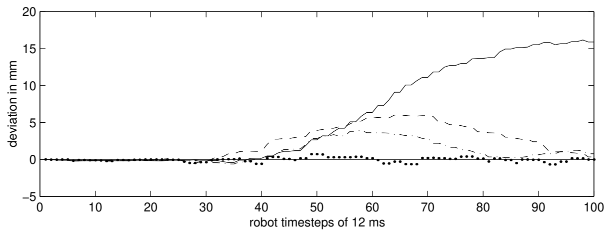


Figure 6: Control error at a straight edge (solid = without sensor control, dashed = with indirect sensor control but without prediction, dotted = with predictive sensor control, dash-dotted = normal visual servoing)

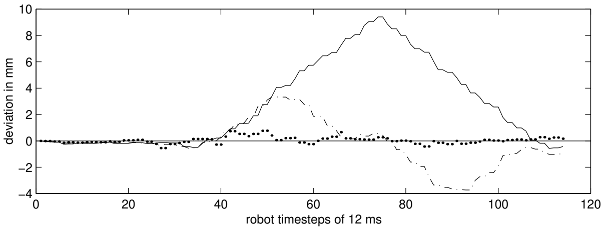


Figure 7: Control error at an edge with an unexpected corner (solid = without sensor control, dotted = with predictive sensor control, dash-dotted = normal visual servoing)

the orientation of the camera $\phi$ has been maintained after the previous experiement since calibration is not possible at the very beginning of the path.

The method is limited by joint elasticity which at least in some timesteps inhibits the correct measurement of $x_a$. So the method can be improved if the deflection of the joints can be estimated thus yielding the actual position and orientation of the camera.

The accuracy further depends on the ability of the feeedforward controller to follow a given path without dynamical deviations. By this, the reachable accuracy is restricted to about 0.3 mm as well.

On the other side, the resolution of the images seems adequate though 1 pixel corresponds to 0.45 mm. This means that the control error reached is less than 1 pixel.

The experiment is repeated in figure 7 with reduced speed. This time there is a non-modelled corner in the edge. Again, the proposed method (RMS = 0.25 mm) is superior to the standard visual servoing method (RMS = 1.7 mm).

## 5 Conclusion

Vision based control of an industrial robot at full speed highlights the performance of the proposed predictive control approach. In each image two points are localized to compute the next controller timesteps of the desired path and to calibrate the actual sensor coordinate system. This is used for a learned feedforward controller to follow the sensed path in spite of deviations due to the robot dynamics. Prediction and control is realized in a position based fashion. At the same time the vision system asynchronuously provides the controller with new sensory data.

Future experiments will focus on implementation aspects as filtering or the problem dependent selection of the camera's focal length or tilt, respectively. In addition, representations of the edge as higher order polynomials are tested with curved edges and more DOFs.

## References

[CG96] P. I. Corke and M. C. Good. Dynamic effects in visual closed-loop systems. *IEEE Trans. on Robotics and Automation*, 12(5):671–683, Oct. 1996.

[Cor95] P. I. Corke. Dynamic issues in robot visual-servo systems. In *Int. Symp. on Robotics Research ISRR'95*, pages 488–498, Herrsching, Germany, 1995. Springer.

[HHC96] S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, Oct. 1996.

[Hir93] G. Hirzinger. ROTEX - the first robot in space. In *6th International Conference on Advanced Robotics ICAR '93*, Tokyo, Japan, Nov. 1993.

[JFN97] M. Jägersand, O. Fuentes, and R. Nelson. Experimental evaluation of uncalibrated visual servoing for precision manipulation. In *Proc. IEEE Int. Conference on Robotics and Automation*, Albuquerque, New Mexico, April 1997.

[LH94] F. Lange and G. Hirzinger. Learning to improve the path accuracy of position controlled robots. In *IEEE/RSJ/GI Int. Conference on Intelligent Robots and Systems*, München, Germany, Sept. 1994.

[LH96] F. Lange and G. Hirzinger. Learning force control with position controlled robots. In *Proc. IEEE Int. Conference on Robotics and Automation*, pages 2282–2288, Minneapolis, Minnesota, April 1996.

[vdSG97] P. van der Smagt and F. Groen. *Visual feedback in motion*, pages 37–73. Academic Press, Boston, Massachusetts, 1997.

[WCH92] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(10):965–980, Oct. 1992.