

Predictive Visual Tracking of Lines by Industrial Robots

Friedrich Lange
Institute of Robotics and Mechatronics
Deutsches Zentrum für Luft- und Raumfahrt (DLR)
Oberpfaffenhofen, D-82234 Wessling, Germany
E-mail: Friedrich.Lange@dlr.de

Gerd Hirzinger

Abstract

Many tasks for industrial robots can be described by high precision line following at high speed. This can be executed accurately if the lines are sensed by a camera since then not only the desired pose at the current time step is sensible. In addition a segment of the desired path can be predicted. We propose polynomials to represent the progression of the elements of the desired pose. This allows to realize a dynamical sensor control architecture that considers the two main problems: low sampling rate and delays in image processing, and deviations from commanded paths due to the robot dynamics. In contrast to previous publications we now present the complete formulae to control translation and orientation of the robot by tracking (curved) lines that are visible for a single eye-in-hand camera. Experiments using off-the-shelf hardware show that the robot can be precisely controlled at high speed.

1 Introduction

Tracking of lines is one of the standard tasks for vision in industrial robotics. Usual applications are laser welding, soldering, sealing, or glueing. In these cases the robot has to keep close tolerances of less than 1 mm across the path. The problem is that the desired path is not precisely given a priori. Imagine e.g. a sealing strip that has to

be glued parallel to an edge, or two objects that have to be spliced by soldering. In both cases the positions of edges or lines have to be detected to define the desired path (see Fig. 1).

The problem of following a (curved) line is similar to the standard problem in visual servoing where a camera has to follow a moving object. The difference is that an object can only be sensed at its current position. Future positions may be predicted from estimations of speed and acceleration as e.g. in [1, 2, 3]. In contrast, a line can be seen in some region around the current center of the image. So there is enough information to define a substantial part of a desired path with respect to the current camera pose. This is essential for accurate motion at high speed. Besides, it allows to do without specialized hardware that is required for high performance object tracking (as e.g. in [4]). For line following we can restrict to standard

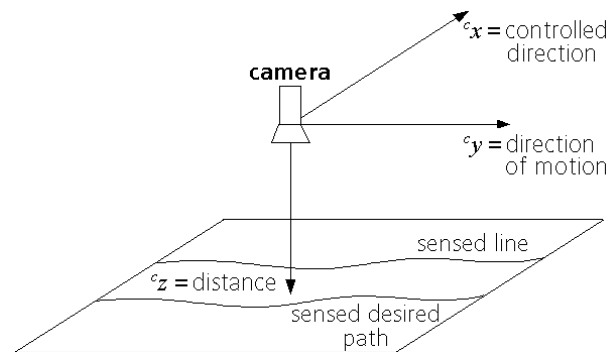


Figure 1: General setup

CCIR systems.

Another critical point is the accurate execution of a desired path, in spite of disadvantageous robot dynamics. Especially for advanced robot use this is a challenge for the control architecture.

Andreff et al. [5] derive a special representation of straight lines in 3D. A scene of different lines is evaluated for controlling the robot's motion to its (static) nominal pose. First, the orientational error of the lines in the image is minimized, then the location is tracked. In singular configurations a third step may be required to localize an additional point.

Mahony and Hamel [6] control the pose of unmanned aerial vehicles (e.g. helicopters) with respect to two or more parallel lines as e.g. motorways or power lines. Their approach uses force and torque inputs to control the dynamics in an image-based fashion.

Rives and Borrelly [7] track an underwater pipe. They evaluate position and slope of two lines which limit the cylindrical shape of the pipe. The translation along the pipe and the rotation around the pipe cannot be controlled by image data. The remaining 4 degrees of freedom (dof's) are controlled. Image-based servoing means that the control input is directly computed from image data. This implies that the dynamics of the vehicles are part of the feedback loop.

Baeten et al. [8, 9] track a contour of unknown shape using an industrial robot with a force controller. In addition to the force signal they process the image data of the contour taken by a robot mounted camera. The camera is mounted ahead of the force sensor to be able to predict contact forces for feedforward. In case of corners in the contour a special algorithm is started to rotate the robot hand in such a way that the camera remains over the contour. In addition, speed is reduced when passing the corner.

Several lines to be followed by an industrial robot have been discussed by Gangloff et al. [10]. Using a positional interface they control up to 5 dof's of the robot pose when following 3 parallel lines with

known spacing. The visual system processes the locations and the tangents of the lines at the current position. The position-based approach considers the robot dynamics by predictive control which results to be superior to usual PID controllers.

These approaches use lines to detect the pose of the camera while following the lines. The further progression of the lines in the images is considered by tangents, not by curvature. A reason may be that in many applications curvature is too small to be detected accurately. Besides, the use of points and tangents in the image center is quite robust since disturbances as lens distortion or misalignment of the camera will affect mainly the outer image regions.

Robot dynamics have been considered by Conticelli and Allotta [11] in form of a two-level architecture. An inner loop is designed to linearize the robot dynamics whereas the image-based controller is realized in an outer loop. So the latter assumes a decoupled controlled system.

This is similar to a usual approach of force control for robots with positional interface. Generally, in sensor control applications with industrial robots an inner positional control loop is widely accepted. In most cases this inner loop is implemented by the standard industrial control architecture.

In this paper we merge control of robot dynamics with the interpretation of sensor data. This is done using three main elements. First of all a new control architecture is created. By defining a special interface we achieve the separation of task specific considerations from control issues. So we can apply a general purpose controller.

The second element of our approach is to use sensor data to specify a desired path which is then executed using the selected controller. In other words we do not feedback current sensor signals but we use geometrical information to compute the desired path. With respect to usual control loops our approach is advantageous concerning stability and precision in case of asynchronous sensors.

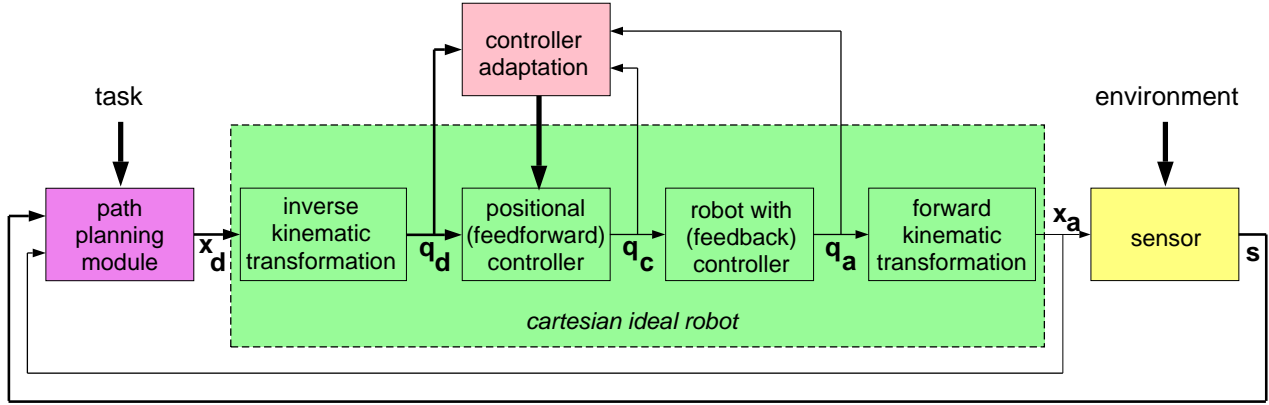


Figure 2: Architecture with cartesian ideal robot when no external forces are present (\mathbf{x}_d = desired cartesian pose, \mathbf{q}_d = desired joint angles, \mathbf{q}_c = commanded joint angles, \mathbf{q}_a = actual joint angles, \mathbf{x}_a = actual cartesian pose, \mathbf{s} = sensed distances, bold face lines denote several time steps of the vectors (i.e. a path segment) instead of a single time step (i.e. a pose vector)

And third, according to the above mentioned special interface, sensing is not restricted to determine a current deviation of the camera pose. Instead, a substantial *segment* of the desired path is computed from every image. This theoretically allows tracking of a desired path without any tracking error.

In our application the desired motion is defined by lines whose nominal progression is given in a model of the task. We represent such a path by a sequence of sampled poses in up to 6 dof’s. For simplicity we restrict to a single camera that is mounted in an eye-in-hand configuration near the tool center point (tcp) of the robot. The camera parameters and the kinematic transformations are known.

The paper is organized as follows: In Sect. 2 we introduce a hierarchical control architecture. An adaptive control scheme is presented for the lower level. We then formulate the task and introduce polynomials to represent the desired trajectory of the robot pose (Sect. 3). The polynomial parameters can be computed (Sect. 4) if at least three lines can be seen by the camera. We also show, how a reduced number of dof’s can be calculated by not more than one or two lines. Finally, the method is demonstrated in simulations and experiments (Sect. 5).

2 Dynamical aspects

2.1 Sensor control architecture

In most robotic sensing tasks the robot is treated as an ideal positional device which is able to follow a target if its trajectory can be sensed correctly. For high speed motion this assumption is not permitted since then the robot dynamics interfere.

But we can define a hierarchical architecture with an *ideal robot* as lower level. Thus our architecture (Fig. 2) distinguishes between control of robot dynamics and sensory feedback. The ideal robot executes the desired trajectory exactly and without time-delay. The outer loop interpretes the image data to move the tcp in the desired way. The controller of the robot dynamics is designed independently from the tracking task. As well the visual servoing task can be formulated without knowing the robot characteristics.

As interface we assume that the ideal robot online accepts sampled desired poses \mathbf{x}_d of the tcp. As output we define the actual pose \mathbf{x}_a of the tcp. This is common for a usual interface to integrate sensed setpoints. Beyond it, here we assume that in each time step not only the current desired pose is given but a whole path segment of n_d time steps of the desired path. Signals representing such path segments are illustrated by bold

face lines in Fig. 2. Especially the desired poses of future time steps enable the controller to realize an ideal robot ($\mathbf{x}_a(k) = \mathbf{x}_d(k)$).

Desired poses of future timesteps have already been proposed in form of predictive sensors [12, 8]. Such sensors are mounted aside the tcp in such a way that they sense before the tcp reaches the respective position.

The compensation of robot dynamics by computing future timesteps of the *robot trajectory* has been presented as “predictive functional control” [13] or “predictive control” [14, 15]. Consideration of latencies by prediction has further been studied in [16]. But none of these approaches has been used in combination with predictive sensors. However, to the authors’ assessment, solely the specification of future time steps of the *desired pose* allows the system to operate precisely. This setup has been called *dynamical sensor control architecture* in [17].

With respect to vision, the special property of our method is to evaluate the image not only at the current position, computing the current desired pose. In addition in every sampling step we compute several future desired poses from the progression of the lines, like [10]. This allows not only to compute the required n_d sampling steps of the interface of the ideal robot. In addition, the required path segment can be calculated for future time steps of the controller. This is advantageous since the control loop may sample faster than the video rate.

2.2 Implementation of the ideal robot

There are several possibilities to realize an ideal robot, e.g. a model based approach. In this case the interface would describe derivatives of the desired pose instead of future values. It would go beyond the scope of this paper to survey all feasible control strategies and their related interfaces. Instead, we only describe our approach [18].

We decided to use an *adaptive feedforward* controller. So we leave the highly optimized cascaded

feedback control loop unchanged. Its main advantages are the high sampling rate of the inner loops and the sophisticated tuning to the robot characteristics. The disadvantage of the industrial control is that it is not able to guarantee ideal tracking of an arbitrary desired trajectory since the desired path is present for the controller only in form of a snapshot. Other time instants of the desired path are not known.

We set up control of the robot in joint space since then the system is almost decoupled. Control in cartesian space is more expensive. So we have to transform the desired path \mathbf{x}_d to vectors of desired joint angles \mathbf{q}_d and to compute the actual poses \mathbf{x}_a in cartesian space from the joint values \mathbf{q}_a .

The interface of the industrial feedback controller accepts joint commands \mathbf{q}_c as input. So the task of the adaptive feedforward controller is to provide the joint commands $\mathbf{q}_c(k)$ by calculations from future time steps of the desired path $\mathbf{q}_d(k+i)$ with $i = 0, \dots, n_d - 1$.

The number of sampling steps n_d of desired poses within the interface of the ideal robot is about twice the number of sampling steps that fit into the robot time constant. As long as no contact forces are present, this allows that the ideal robot can be realized properly. Further details of our implementation can be found in [18]. Therefore,

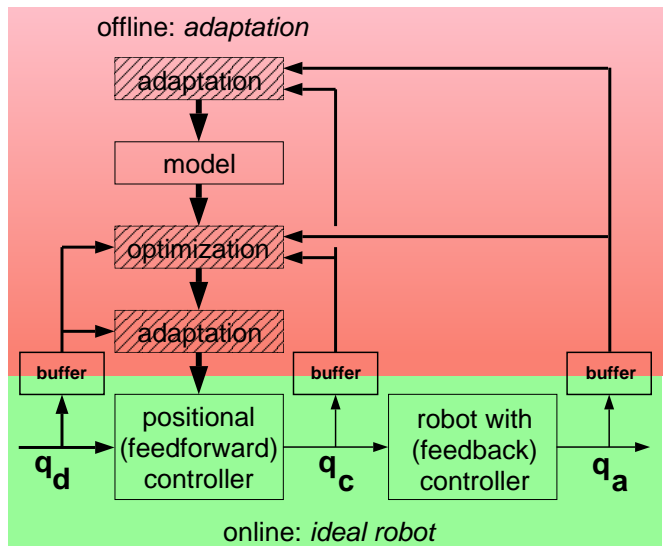


Figure 3: Structure of adaptation

here we restrict to a short overview.

The adaptation is executed in three steps (hatched blocks in Fig. 3) using selected trajectories. After the first run the robot with its feedback control system is identified (1st step). Then the positional controller is adapted iteratively. This is done by a posteriori optimization of the sample trajectory (2nd step), always adapting the parameters of the feedforward controller to fulfill the optimal trajectory (3rd step). The resulting controller will be able to control arbitrary paths, at least in the neighbourhood of the current working point.

The iterative approach tolerates a coarsely identified model since remaining errors are incrementally reduced. In contrast we need a controller which can represent nonlinear mappings with high precision. A global neural net which directly computes the commands \mathbf{q}_c is not realizable with sufficient accuracy. So we use that the robot is almost linear because the invariant motor inertia is dominant compared to the position dependent arm inertia. Therefore nonlinear parts may be scaled finer. So we propose an extended setup with linear approach and parallel neural nets to compensate for couplings. As neural nets we use multilayer perceptrons with well suited training algorithms [19].

Additional feedback of measured values is not required because of the good repeatability of modern industrial robots. So what we need is pure feedforward control. Compared with model-based methods this corresponds to the *computed torque method*, since feedback (in the industrial control system) is independent from the desired motion.

2.3 Discussion

The dynamical sensor control architecture has no effect on stability since it does not feedback the robot position. Sensor and pose feedback in Fig. 2 seem to affect stability, but actually the signals are only used to determine the desired motion (see Sects. 3 and 4). So stability is only depending on the internal feedback control loop which is represented by a single block in Fig. 2.

Summarizing, the desired cartesian poses are updated in video rate of e.g. 50 Hz. Then the joint commands are computed by the feedforward controller in the outer loop robot control rate, e.g. 83 Hz. Feedback control is then executed in the inner loop control rate or even continuously. This allows to perform high bandwidth motion in spite of a low sensor rate. The approach even tolerates asynchronous sensing. As extreme example a single image is sufficient to accurately track a visible line. In contrast, usual visual servoing algorithms increment positional commands until the tracking error diminishes. Due to delays this will often cause overshooting when reaching the target pose.

Limiting for the performance of our implementation is the existence of compliance, mainly in the robot joints. Then the actual joint positions are not available. In [20] a method is presented in which the positional controller is adapted using offline available measurements of the arm positions. These measurements cover only 2 dof's that are used to estimate all 6 joint values.

3 Sensor based path planning

3.1 Definition of the task

In addition to previous approaches as [17] we now describe the general case. This will cause a quite complex notation, it is essential however since in real systems camera and task frame are not exactly aligned and tiny misalignments can cause substantial errors if they are not considered. As well temporal precision is important since at high speed the way corresponding to a single sampling step is an order of magnitude greater than the desired accuracy. Therefore interpolations are required throughout the paper.

We assume a *reference path* that is defined with respect to *nominal lines* and that is modified by *sensed (real) lines* in the images. The result is denoted as *desired path* (see Fig. 4).

There are three reasons for the assumption of a reference path. First, we need a default path for

the case that not all dof's of the robot can be extracted from the images. Second, image data yield only a geometric description of the desired path. A velocity profile has to be defined in addition, e.g. according to the curvature of the desired path. Using a reference trajectory we can adopt the velocities. Third, the uncertainty of the path is of low bandwidth since it comes from unprecise mounting of parts or from inaccurate execution of preceding production steps. Therefore the sensed path may be broadly smoothed. On the other side the path itself will have regions of high bandwidth, e.g. corners or path segments with high curvature. Sensing of such path characteristics forbids smoothing. But it is possible to smooth the modifications with respect to the reference path. So the use of a reference path allows high bandwidth path segments in spite of low bandwidth path sensing. Finally, for industrial tasks a reference path is always available so that its assumption is no problem. If it should be, we can define a straight line or a single point as reference path. In that case sensed path modifications will have to be quite large.

So the task is to *refine* a path according to sensor data. The original path is called reference path

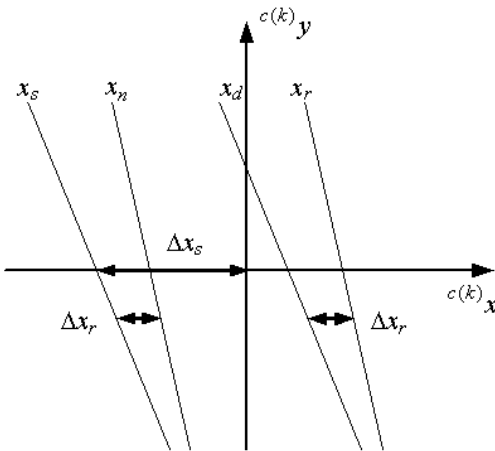


Figure 4: Notation of the individual paths and lines with respect to the current camera frame if only a single line is tracked to control one dof (x_r = reference path, x_d = desired path, x_s = sensed (actual) line, x_n = nominal line, Δx_s = sensed displacement from of the line, Δx_r = modification of the reference path)

and is denoted by a sequence of transformation matrices ${}^w\mathbf{T}_{r(k)}$ which express the reference frame r at sampling steps k with respect to the world coordinate system w . In this notation the difference between the reference frame and the world frame is represented in the world coordinate system w (for further notation see appendix B).

Assume that at time instant k an image has been recorded. Then, using the information of this image, the reference path ${}^w\mathbf{T}_{r(k+i)}$ has to be modified for $i = 0, 1, 2, ..$ subsequent robot sampling steps to define a refined path which is called the desired path ${}^w\mathbf{T}_{d(k+i/k)}$. The argument of the lower index denotes the prediction from time step k to time step $k + i$.

Image data are used to detect the pose of one or several edges or lines. In the specification of the reference path besides the reference pose of the top ${}^w\mathbf{T}_{r(k+i)}$ we assume that nominal poses ${}^w\mathbf{p}_{n(k+i,l)}$ for all lines l are provided. The sensed (real) line point of line l is denoted by ${}^w\mathbf{p}_{s(j,l)}$, where within line l j is the index of a point which approximately corresponds to time step $k + i$.

The task might be formulated as the computation of a sensor correction ${}^{r(k+i)}\mathbf{T}_{d(k+i/k)} = {}^w\mathbf{T}_{r(k+i)}^{-1} \cdot {}^w\mathbf{T}_{d(k+i/k)}$ for $i = 0, 1, 2, ..$ by sensing several points j on several lines l in such a way that the detected line points ${}^{d(k+i(j,l))}\mathbf{p}_{s(j,l)}$ in the sensed desired coordinate system defined by ${}^w\mathbf{T}_{d(k+i(j,l)/k)}$ are identical with the nominal line points ${}^{r(k+i(j,l))}\mathbf{p}_{n(k+i(j,l),l)}$ in the reference system. This means

$${}^{d(k+i(j,l))}\mathbf{p}_{s(j,l)} = {}^{r(k+i(j,l))}\mathbf{p}_{n(k+i(j,l),l)}, \quad (1)$$

where $k + i(j, l)$ is the time instant corresponding to point j of line l .

3.2 Representation of the sensed path corrections

Using Eq. (1) for a line l we get a sequence of points j of the sensor correction ${}^{r(k+i(j,l))}\mathbf{T}_{d(k+i(j,l)/k)}$. Normally $k + i(j, l)$ will

not coincide with the integral sampling steps $k + i$ of the controller. But we need a sequence of sensor corrections at the sampling steps, i.e. ${}^{r(k+i)}\mathbf{T}_{d(k+i/k)}$ with integral values of $i = 0, \dots$. We avoid the problem by first estimating a polynomial ${}^{r(k+i(j,l))}\mathbf{T}_{d(k+i(j,l)/k)}$ using the points j of lines l and then computing the values ${}^{r(k+i)}\mathbf{T}_{d(k+i/k)}$ at the sampling steps. The latter is an interpolation or extrapolation from sensed poses at points of the desired path to the desired pose at time step $k + i$.

Since a polynomial of an orthonormal transformation matrix ${}^{r(k+i)}\mathbf{T}_{d(k+i/k)}$ is hard to compute we use a pose vector ${}^{r(k+i)}\mathbf{x}_{d(k+i/k)}$ with 6 nonredundant elements, 3 translational and 3 rotational pose components. For the representation of the orientation we select cardan angles since the angles of path refinements are usually small. So for each of the 6 pose parameters a polynomial is estimated.

Since the point positions are sensed in the camera frame we estimate ${}^{r(k+i(j,l))}\mathbf{T}_{d(k+i(j,l)/k)}$ from line points ${}^{c(k)}\mathbf{p}_{s(j,l)}$ expressed in the actual camera frame ${}^w\mathbf{T}_{c(k)}$. This camera frame is assumed to be known since it can be computed from the actual robot joint angles. But due to high speed the image may be blurred in the direction of motion. So we assume all information in the direction of motion as disturbed. Therefore the location of lines perpendicular to motion cannot be used. So we assume that lines are more or less parallel to motion. This is reasonable for most applications.

Without loss of generality we rotate the camera such that motion is mainly vertical (in y -direction) in the image (see Fig. 1). This simplifies the equations. On the other hand, in field mode only every second image row is defined so that the vertical accuracy is inferior anyway.

The polynomial is expressed with respect to the position and orientation of the camera since like this the representation is simplified. Exactly speaking, the y -component of the sampled reference path ${}^{c(k)}\mathbf{T}_{r(k+i)}$ serves as input, assuming that this is unambiguous. A polynomial with respect to the y -component of ${}^{r(k+i)}\mathbf{T}_{c(k)}$ would be

possible as well. But then even in case of small rotations of the reference pose it would not be guaranteed that the y -component is increasing monotonically. So the demands with respect to the reference path are still more restrictive. That is why we define the polynomial to be

$${}^{r(k+i)}\mathbf{x}_{d(k+i/k)} = \mathbf{pol}({}^{c(k)}y_{r(k+i)}), \quad (2)$$

where every dof of \mathbf{x} is represented by a scalar polynomial

$$\mathit{pol}_{j_p}({}^{c(k)}y_{r(k+i)}) = \sum_{i_p=0}^{n_p-1} \theta_{j_p, i_p}(k) \cdot ({}^{c(k)}y_{r(k+i)})^{i_p} \quad (3)$$

with n_p parameters θ that are estimated.

3.3 Discussion

So the desired path is computed by the following steps:

1. Computation (interpolation) of the nominal points ${}^w\mathbf{p}_{n(k+i(j,l),l)}$ and the corresponding points of the reference path ${}^w\mathbf{T}_{r(k+i(j,l))}$ (i.e. computation of $i(j, l)$ from j and l)
2. Computation of the desired values ${}^{r(k+i(j,l))}\mathbf{x}_{d(k+i(j,l))}$ and the input ${}^{c(k)}y_{r(k+i(j,l))}$ of the polynomials
3. Estimation of the polynomial parameters $\theta_{j_p, i_p}(k)$ using a Least-Square algorithm or the equations of correction of a Kalman filter
4. Computation of the sensor correction of steps k to $k + n_d - 1$ by insertion of ${}^{c(k)}y_{r(k+i)}$ and transformation of the result to ${}^w\mathbf{T}_{d(k+i/k)}$

In the subsequent robot sampling steps, i.e. as long as no new images are available, only the last step is executed.

On availability of new image data the representation is changed since the point of reference ${}^w\mathbf{T}_c$ of the polynomial representation has changed. When using a Kalman filter this change is done by the equations of prediction due to a known camera motion. In case of low noise in edge detection the prediction is not obligatory since the polynomial can be estimated without the knowledge of past image data.

Elasticity has been mentioned at the implementation of the ideal robot. It further has to be considered for path planning since with elastic joints the actual camera pose is not precisely known from the internal measurements. This problem can be solved by measurements of external devices, as for the adaptation. Other approaches are the use of an observer, or the assumption that the ideal robot is ideal, meaning that $\mathbf{x}_a \approx \mathbf{x}_d$. There are further investigations necessary concerning this point.

In contrast to [5] we do not use the representation with normalized Plücker coordinates. This is because we do not map straight lines but only points of lines. And these lines are not restricted to be straight. In contrast we consider polynomial representations. This exceeds the approach with Plücker coordinates which is useful for other tasks.

4 Using sensed lines for pose definition

4.1 Basic approach

Eq. (1) can be expressed by

$$\begin{aligned} d(k+i(j,l)/k)\mathbf{T}_{r(k+i(j,l))} &\cdot r(k+i(j,l))\mathbf{P}_{s(j,l)} \\ &= r(k+i(j,l))\mathbf{P}_{n(k+i(j,l),l)} \end{aligned} \quad (4)$$

$$\begin{aligned} r(k+i(j,l))\mathbf{T}_{d(k+i(j,l)/k)} &\cdot r(k+i(j,l))\mathbf{P}_{n(k+i(j,l),l)} \\ &= r(k+i(j,l))\mathbf{P}_{s(j,l)} \end{aligned} \quad (5)$$

$$\begin{aligned} r(k+i(j,l))\mathbf{T}_{d(k+i(j,l)/k)} &\cdot r(k+i(j,l))\mathbf{P}_{n(k+i(j,l),l)} \\ &= {}^w\mathbf{T}_{r(k+i(j,l))}^{-1} \cdot {}^w\mathbf{T}_{c(k)} \cdot {}^{c(k)}\mathbf{P}_{s(j,l)}. \end{aligned} \quad (6)$$

In this equation ${}^{c(k)}\mathbf{P}_{s(j,l)}$ is a detected point of the 3D line with respect to the camera. This can be sensed if the distance is known. ${}^w\mathbf{T}_{c(k)}$ is the pose of the camera at the time instant of the exposure. This pose is assumed to be known. ${}^w\mathbf{T}_{r(k+i(j,l))}$ and ${}^{r(k+i(j,l))}\mathbf{P}_{n(k+i(j,l),l)}$ are the definition of the reference path and the nominal lines respectively. They are given as well. So the sensor correction ${}^{r(k+i(j,l))}\mathbf{T}_{d(k+i(j,l)/k)}$ might be computed if the distance between the line and the camera is known.

In the one-dof case of Fig. 4 where the all orientations coincides with the world system, Eq. (5) corresponds to

$$\Delta x_r = x_d - x_r = x_s - x_n \quad (7)$$

and can easily be solved.

Unfortunately, in the general case the elements of ${}^{r(k+i(j,l))}\mathbf{T}_{d(k+i(j,l)/k)}$ cannot be extracted from Eq. (5) or (6).

To study this we assume small rotations between the reference pose and the desired pose. So we can use cardan angles or the roll, pitch, yaw representation for the orientation matrix in ${}^{r(k+i(j,l))}\mathbf{T}_{d(k+i(j,l)/k)}$. With ${}^{r(k+i(j,l))}\gamma_{d(k+i(j,l)/k)}$, ${}^{r(k+i(j,l))}\beta_{d(k+i(j,l)/k)}$, ${}^{r(k+i(j,l))}\alpha_{d(k+i(j,l)/k)}$ for the roll, pitch, and yaw angles we get

$${}^{r(k+i(j,l))}\mathbf{T}_{d(k+i(j,l)/k)} = \begin{pmatrix} 1 & -\alpha & \beta & x \\ \alpha & 1 & -\gamma & y \\ -\beta & \gamma & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (8)$$

where the indices ${}^{r(k+i(j,l))} \dots {}^{d(k+i(j,l)/k)}$ have been omitted because of the limited layout.

${}^{c(k)}\mathbf{P}_{s(j,l)}$ has been said to be sensible if the distance is known. It can be expressed from 2D

points (ξ, η) in a fictive image plane with a focal length of 1 by

$${}^{c(k)}\mathbf{p}_{s(j,l)} = \begin{pmatrix} \xi(k, j, l) \cdot {}^{c(k)}z_{s(j,l)} \\ \eta(k, j, l) \cdot {}^{c(k)}z_{s(j,l)} \\ {}^{c(k)}z_{s(j,l)} \\ 1 \end{pmatrix}. \quad (9)$$

This yields

$$\begin{pmatrix} 1 & -\alpha & \beta & x \\ \alpha & 1 & -\gamma & y \\ -\beta & \gamma & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot {}^{r(k+i(j,l))}\mathbf{p}_{n(k+i(j,l),l)} \\ = {}^{r(k+i(j,l))}\mathbf{T}_{c(k)} \begin{pmatrix} \xi(k, j, l) \cdot {}^{c(k)}z_{s(j,l)} \\ \eta(k, j, l) \cdot {}^{c(k)}z_{s(j,l)} \\ {}^{c(k)}z_{s(j,l)} \\ 1 \end{pmatrix} \quad (10)$$

where

$${}^{r(k+i(j,l))}\mathbf{T}_{c(k)} = {}^w\mathbf{T}_{r(k+i(j,l))}^{-1} \cdot {}^w\mathbf{T}_{c(k)}. \quad (11)$$

For n_l detected lines Eq. (10) has $6+n_l$ unknowns, namely the 6 elements of ${}^{r(k+i(j,l))}\mathbf{T}_{d(k+i(j,l)/k)}$ and n_l distances ${}^{c(k)}z_{s(j,l)}$ of the sensed lines. So several image points $(\xi(k, j, l), \eta(k, j, l))$ have to be sensed for a solution. This will be studied in Sects. 4.3 and 4.4.

Before, we analyze how to correlate points j of lines l and time instants $k+i(j, l)$ of the reference path or the nominal lines.

4.2 Correlation of line points and time instants of the desired path

In the last section we assumed that detected line points ${}^{c(k)}\mathbf{p}_{s(j,l)}$ can be related to nominal line points ${}^{r(k+i(j,l))}\mathbf{p}_{n(k+i(j,l),l)}$. Reconsidered this implies a relation between the position of a point and a time step of the reference path.

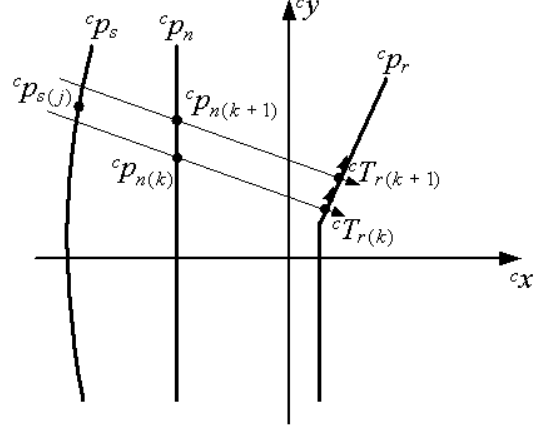


Figure 5: Lines and paths in the camera system (${}^c\mathbf{p}_r$ = reference path, ${}^c\mathbf{p}_s$ = sensed (actual) line, ${}^c\mathbf{p}_n$ = nominal line)

First we define the relation between a point on the reference path and the corresponding point on a line in such a way that the y -component of a nominal line point with respect to the reference system, i.e. ${}^{r(k+i(j,l))}y_{n(k+i(j,l),l)}$ is the y -component of the corresponding point on the reference path (see Fig. 5).

$${}^{r(k+i(j,l))}y_{n(k+i(j,l),l)} = {}^{r(k+i(j,l))}y_{r(k+i(j,l))} = 0 \quad (12)$$

Then according to Eq. (1) we define

$${}^{r(k+i(j,l))}y_{n(k+i(j,l),l)} = {}^{d(k+i(j,l))}y_{s(j,l)} \quad (13)$$

For small orientational sensor corrections this can be expressed by

$$\begin{aligned} {}^{d(k+i(j,l))}y_{s(j,l)} &= 0 \\ &= (0100) \cdot {}^{d(k+i(j,l))}\mathbf{T}_{c(k)} \cdot {}^{c(k)}\mathbf{p}_{s(j,l)} \\ &\approx (0100) \cdot {}^{r(k+i(j,l))}\mathbf{T}_{c(k)} \cdot {}^{c(k)}\mathbf{p}_{s(j,l)} \end{aligned} \quad (14)$$

This means that the sensor correction is computed from line points in the x - z -plane of the reference system at the considered time instant $k+i(j, l)$. (The influence of orientational sensor corrections will be discussed in Sect. 4.5.)

At the first glance it seems to be a problem that we cannot sense 3D line points ${}^{c(k)}\mathbf{p}_{s(j,l)}$ but only 2D points (ξ, η) . So we have to use estimates ${}^{c(k)}\hat{z}_{s(k+i,j,l)}$ for ${}^{c(k)}z_{s(j,l)}$ in Eq. (9). These estimates will be the distances to the nominal lines ${}^{c(k)}z_{n(k+i(j,l),l)}$ in the first step. Then, according to Eq. (5), ${}^{c(k)}\hat{z}_{s(k+i,j,l)}$ might be set to

$${}^{c(k)}\hat{z}_{s(k+i,j,l)} = (0010) \cdot {}^{c(k)}\mathbf{T}_{r(k+i(j,l))} \cdot {}^{r(k+i(j,l))}\mathbf{T}_{d(k+i(j,l)/k-1)} \cdot {}^{r(k+i(j,l))}\mathbf{p}_{n(k+i(j,l),l)} \quad (15)$$

using the previously computed sensor correction ${}^{r(k+i(j,l))}\mathbf{T}_{d(k+i(j,l)/k-1)}$. Since the ${}^{c(k)}z_{s(j,l)}$ later will be determined according to Sect. 4.3, the estimates may be modified iteratively if they differ too much from the values which have been computed from the previous image. (The polynomial values of ${}^{c(k-1)}z_{s(j,l)} = \text{pol}_{z_s}({}^{c(k-1)}y_{r(k-1+i(j,l))})$ are not applicable for initialization since they are related to the previous camera location.)

At least in case of lens distortion it is not possible to localize a line point by specification of a certain component of it. So we cannot sense image line points $(\xi(k, j, l), \eta(k, j, l))$ with given ${}^{r(k+i(j,l))}y_{s(j,l)}$. Instead we sense image line points $(\xi(k, j, l), \eta(k, j, l))$ in selected image rows that coarsely cover the region of the next n_d time steps. Then we compute the corresponding time instants $k+i(j, l)$ such that Eq. (13) is satisfied. Thereafter the reference path and the nominal lines have to be interpolated since they are defined pointwise for integral $k+i$.

With given ${}^w\mathbf{T}_{c(k)}$ we begin by computing the transformations ${}^{r(k+i)}\mathbf{T}_{c(k)}$ and ${}^{r(k+i)}\mathbf{p}_{n(k+i,l)}$ for all integral time instants $k+i$ under consideration.

Then every detected line point $(\xi(k, j, l), \eta(k, j, l))$ is represented with respect to each reference system ${}^w\mathbf{T}_{r(k+i)}$, where because of Eqs. (12) to (14)

$$\begin{aligned} d(k+i)y_{s(j,l)} &\approx {}^{r(k+i)}y_{s(j,l)} \\ &= {}^{r(k+i)}y_{c(k)} + \sum_{i_1=0}^2 {}^{r(k+i)}t_{c(k),1i_1} \cdot {}^{c(k)}\hat{p}_{s(k+i,j,l),i_1} \end{aligned} \quad (16)$$

should be zero. ${}^{c(k)}\hat{p}_{s(k+i,j,l),i_1}$ is computed using Eq. (9) with ${}^{c(k)}\hat{z}_{s(k+i,j,l)}$ from Eq. (15).

By testing the integral values of i we get a lower limit \underline{i} and an upper limit $\underline{i}+1$ for $i(j, l)$. By linear interpolation

$$\begin{aligned} &(\underline{i}+1-i(j,l)) \cdot \left({}^{r(k+\underline{i})}y_{c(k)} \right. \\ &\quad \left. + \sum_{i_1=0}^2 {}^{r(k+\underline{i})}t_{c(k),1i_1} \cdot {}^{c(k)}\hat{p}_{s(k+\underline{i},j,l),i_1} \right) \\ &+ (i(j,l)-\underline{i}) \cdot \left({}^{r(k+\underline{i}+1)}y_{c(k)} \right. \\ &\quad \left. + \sum_{i_1=0}^2 {}^{r(k+\underline{i}+1)}t_{c(k),1i_1} \cdot {}^{c(k)}\hat{p}_{s(k+\underline{i}+1,j,l),i_1} \right) \\ &= {}^{r(k+i(j,l))}y_{s(j,l)} \approx 0. \end{aligned} \quad (17)$$

a floating point value $i(j, l)$ can be computed.

Then we can compute ${}^{r(k+i(j,l))}\mathbf{T}_{c(k)}$ for the detected line point j . Since it is not allowed to interpolate transformation matrices we first have to replace the transformation matrix by a vector ${}^{r(k+i(j,l))}\mathbf{x}_{c(k)}$ of nonredundant elements. These elements are the 3 translational values of ${}^{r(k+i(j,l))}\mathbf{T}_{c(k)}$ and three values of rotation. For the rotation a representation is required which is not singular. This is e.g. the cardan angle representation which is unambiguous for small angles. The latter is guaranteed as for small differences between nominal and sensed lines the real (camera) pose will not be very different from the reference path.

Since with the orientation of the reference path we mean the orientation of the tcp when following the reference path, we now have to suppose that the orientation of the tcp is not perpendicular to the real optical axis of the camera.

So we interpolate ${}^{r(k+i(j,l))}\mathbf{x}_{c(k)}$ by

$$\begin{aligned} {}^{r(k+i(j,l))}\mathbf{x}_{c(k)} &= (\underline{i}+1-i(j,l)) \cdot {}^{r(k+\underline{i})}\mathbf{x}_{c(k)} \\ &\quad + (i(j,l)-\underline{i}) \cdot {}^{r(k+\underline{i}+1)}\mathbf{x}_{c(k)}. \end{aligned} \quad (18)$$

The result is expressed as transformation ma-

trix $r^{(k+i(j,l))} \mathbf{T}_{c(k)}$ which in combination with $r^{(k+i(j,l))} \mathbf{P}_{n(k+i(j,l),l)}$ is required in Eq. (10). $r^{(k+i(j,l))} \mathbf{P}_{n(k+i(j,l),l)}$ can be interpolated, too. In many cases $r^{(k+i(j,l))} \mathbf{P}_{n(k+i(j,l),l)}$ is constant.

4.3 Directly sensible dof's

Eq. (5) can be represented by 3 scalar equations.

$$\begin{pmatrix} 1 & -\alpha & \beta & x \end{pmatrix} \cdot r^{(k+i(j,l))} \mathbf{P}_{n(k+i(j,l),l)} = r^{(k+i(j,l))} x_{s(j,l)} \quad (19)$$

$$\begin{pmatrix} \alpha & 1 & -\gamma & y \end{pmatrix} \cdot r^{(k+i(j,l))} \mathbf{P}_{n(k+i(j,l),l)} = r^{(k+i(j,l))} y_{s(j,l)} \quad (20)$$

$$\begin{pmatrix} -\beta & \gamma & 1 & z \end{pmatrix} \cdot r^{(k+i(j,l))} \mathbf{P}_{n(k+i(j,l),l)} = r^{(k+i(j,l))} z_{s(j,l)} \quad (21)$$

With Eq. (14) the right hand side of the second equation is always zero so that only two equations are left. With $r^{(k+i(j,l))} y_{n(k+i(j,l),l)} = 0$ from Eq. (12) and Eq. (10) we have

$$\begin{aligned} & r^{(k+i(j,l))} x_{n(k+i(j,l),l)} \\ + & r^{(k+i(j,l))} \beta_{d(k+i(j,l)/k)} \cdot r^{(k+i(j,l))} z_{n(k+i(j,l),l)} \\ + & r^{(k+i(j,l))} x_{d(k+i(j,l)/k)} \\ = & r^{(k+i(j,l))} t_{c(k),00} \cdot \xi(k, j, l) \cdot c^{(k)} z_{s(j,l)} \\ + & r^{(k+i(j,l))} t_{c(k),01} \cdot \eta(k, j, l) \cdot c^{(k)} z_{s(j,l)} \\ + & r^{(k+i(j,l))} t_{c(k),02} \cdot c^{(k)} z_{s(j,l)} \\ + & r^{(k+i(j,l))} t_{c(k),03} \end{aligned} \quad (22)$$

and

$$\begin{aligned} - & r^{(k+i(j,l))} \beta_{d(k+i(j,l)/k)} \cdot r^{(k+i(j,l))} x_{n(k+i(j,l),l)} \\ + & r^{(k+i(j,l))} z_{n(k+i(j,l),l)} \\ + & r^{(k+i(j,l))} z_{d(k+i(j,l)/k)} \\ = & r^{(k+i(j,l))} t_{c(k),20} \cdot \xi(k, j, l) \cdot c^{(k)} z_{s(j,l)} \\ + & r^{(k+i(j,l))} t_{c(k),21} \cdot \eta(k, j, l) \cdot c^{(k)} z_{s(j,l)} \\ + & r^{(k+i(j,l))} t_{c(k),22} \cdot c^{(k)} z_{s(j,l)} \\ + & r^{(k+i(j,l))} t_{c(k),23}. \end{aligned} \quad (23)$$

In these equations the variables $c^{(k)} z_{s(j,l)}$, $r^{(k+i(j,l))} x_{d(k+i(j,l)/k)}$, $r^{(k+i(j,l))} z_{d(k+i(j,l)/k)}$, and $r^{(k+i(j,l))} \beta_{d(k+i(j,l)/k)}$ are unknown. In addition the other components of the sensor correction $r^{(k+i(j,l))} \mathbf{T}_{d(k+i(j,l)/k)}$ have to be computed, too. But this will require additional equations (see Sect. 4.4). Reconsidered, the number of unknowns that can be computed by Eqs. (22) and (23) is depending on the number of lines that are visible (see Table 1). Possibly we need assumptions on some of the components of $r^{(k+i(j,l))} \mathbf{T}_{d(k+i(j,l)/k)}$.

If $r^{(k+i(j,l))} z_{d(k+i(j,l)/k)}$ and $r^{(k+i(j,l))} \beta_{d(k+i(j,l)/k)}$ are given and only $r^{(k+i(j,l))} x_{d(k+i(j,l)/k)}$ is desired we do not need more than one line ($n_l = 1$). We have to solve two equations ((22) and (23)) for the two unknowns $c^{(k)} z_{s(j,l)}$ and $r^{(k+i(j,l))} x_{d(k+i(j,l)/k)}$. This can be done separately for each j .

If in addition $r^{(k+i(j,l))} z_{d(k+i(j,l)/k)}$ and / or $r^{(k+i(j,l))} \beta_{d(k+i(j,l)/k)}$ is desired, we have to use additional lines. Since $i(j,l)$ and thus $r^{(k+i(j,l))} \mathbf{T}_{d(k+i(j,l)/k)}$ might be different for points on different lines we now have to combine the estimation of the polynomial parameters θ with

Variable	1 line	2 lines	≥ 3 lines
$r^{(k+i(j,l))} x_{d(k+i(j,l)/k)}$	yes	yes	yes
$r^{(k+i(j,l))} y_{d(k+i(j,l)/k)}$	no	no	no
$r^{(k+i(j,l))} z_{d(k+i(j,l)/k)}$	no	yes	yes
$r^{(k+i(j,l))} \alpha_{d(k+i(j,l)/k)}$	yes	yes	yes
$r^{(k+i(j,l))} \beta_{d(k+i(j,l)/k)}$	no	no	yes
$r^{(k+i(j,l))} \gamma_{d(k+i(j,l)/k)}$	no	yes	yes

Table 1: Components of the sensor correction that can be computed by a given number of lines

the solution of Eqs. (22) and (23). This is done by replacing the unknowns ${}^{c(k)}z_{s(j,l)}$, ${}^{r(k+i(j,l))}x_{d(k+i(j,l)/k)}$, ${}^{r(k+i(j,l))}z_{d(k+i(j,l)/k)}$, and ${}^{r(k+i(j,l))}\beta_{d(k+i(j,l)/k)}$ by polynomials according to Eq.(3). Note that the ${}^{c(k)}z_{s(j,l)}$ may be different for different lines. Therefore with three lines we have six polynomials. Then, for each combination of j and l we get a linear equation which finally allows to estimate all polynomial parameters θ .

A heuristical interpretation is that without rotation between reference path and camera, the distance ${}^{r(k+i(j,l))}z_{d(k+i(j,l)/k)}$ is computed from the displacement between at least two lines. If we want to determine ${}^{r(k+i(j,l))}\beta_{d(k+i(j,l)/k)}$ as well, we will need at least 3 lines, where the ratio of the displacements among them finally gives ${}^{r(k+i(j,l))}\beta_{d(k+i(j,l)/k)}$.

The estimation of polynomials for ${}^{c(k)}z_{s(j,l)}$ seems redundant since ${}^{c(k)}\mathbf{p}_{s(j,l)}$ may be expressed by ${}^{r(k+i(j,l))}\mathbf{T}_{d(k+i(j,l)/k)}$ and given transformations using Eq. (6). Besides, this approach would provide further dof's of the sensor correction. Unfortunately it yields a singular system of equation. So it cannot be applied.

The reason is, that the remaining components of the pose, i.e. ${}^{r(k+i(j,l))}\alpha_{d(k+i(j,l)/k)}$, ${}^{r(k+i(j,l))}\gamma_{d(k+i(j,l)/k)}$, and ${}^{r(k+i(j,l))}y_{d(k+i(j,l)/k)}$ cannot be determined from line points of a single time instant. So Eq. (1) cannot be used.

4.4 Additional dof's

We define a second criterion to compute the other components of the sensor corrections ${}^{r(k+i(j,l))}\mathbf{T}_{d(k+i(j,l)/k)}$. Remember that in Sect. 3.1 we defined the detected line points ${}^{d(k+i(j,l))}\mathbf{p}_{s(j,l)}$ in the sensed desired coordinate system to be identical with the nominal line points ${}^{r(k+i(j,l))}\mathbf{p}_{n(k+i(j,l),l)}$ in the reference system. Now we add the orientations of the lines to be identical as well.

We define the orientation of a line by the difference vector between consecutive line points. With $j(i,l)$ as the inverse of the function $i(j,l)$ we get

$$\begin{aligned} & {}^{d(k+i)}\mathbf{p}_{s(j(i+1,l),l)} - {}^{d(k+i)}\mathbf{p}_{s(j(i,l),l)} \\ &= {}^{r(k+i)}\mathbf{p}_{n(k+i+1,l)} - {}^{r(k+i)}\mathbf{p}_{n(k+i,l)} \end{aligned} \quad (24)$$

According to Eq. (1) we can replace ${}^{d(k+i)}\mathbf{p}_{s(j(i,l),l)}$ by ${}^{r(k+i)}\mathbf{p}_{n(k+i,l)}$. So we get

$${}^{d(k+i)}\mathbf{p}_{s(j(i+1,l),l)} = {}^{r(k+i)}\mathbf{p}_{n(k+i+1,l)} \quad (25)$$

This can be transformed to

$$\begin{aligned} & {}^{d(k+i)}\mathbf{T}_{d(k+i+1)} \cdot {}^{d(k+i+1)}\mathbf{p}_{s(j(i+1,l),l)} \\ &= {}^{r(k+i)}\mathbf{T}_{r(k+i+1)} \cdot {}^{r(k+i+1)}\mathbf{p}_{n(k+i+1,l)} \end{aligned} \quad (26)$$

Using Eq. (1) again we get

$$\begin{aligned} & {}^{d(k+i)}\mathbf{T}_{d(k+i+1)} \cdot {}^{r(k+i+1)}\mathbf{p}_{n(k+i+1,l)} \\ &= {}^{r(k+i)}\mathbf{T}_{r(k+i+1)} \cdot {}^{r(k+i+1)}\mathbf{p}_{n(k+i+1,l)} \end{aligned} \quad (27)$$

This equation holds if the matrices are identical, i.e.

$${}^{d(k+i)}\mathbf{T}_{d(k+i+1)} = {}^{r(k+i)}\mathbf{T}_{r(k+i+1)} \quad (28)$$

Note that this equation is independent of individual sensed points or lines. It only depends on frames of the tcp.

The right hand side of Eq. (28) is the definition of the reference path.

$${}^{r(k+i)}\mathbf{T}_{r(k+i+1)} = {}^w\mathbf{T}_{r(k+i)}^{-1} \cdot {}^w\mathbf{T}_{r(k+i+1)} \quad (29)$$

The left hand side

$$\begin{aligned} & {}^{d(k+i)}\mathbf{T}_{d(k+i+1)} \\ &= {}^{r(k+i)}\mathbf{T}_{d(k+i)}^{-1} \cdot {}^{r(k+i)}\mathbf{T}_{d(k+i+1)} \end{aligned} \quad (30)$$

will be considered now. For brevity we denote

$${}^{r(k+i)}x_{d(k+i/k),j_p} = \theta'_{j_p,0} \quad (31)$$

$${}^{r(k+i)}x_{d(k+i+1/k),j_p} = \theta'_{j_p,0} + \theta'_{j_p,1} \quad (32)$$

and

Inserting this into Eq. (8) yields

$${}^{r(k+i)}\mathbf{T}_{d(k+i+1)} = \begin{pmatrix} 1 & -\theta'_{\alpha,0} - \theta'_{\alpha,1} & \theta'_{\beta,0} + \theta'_{\beta,1} & \theta'_{x,0} + \theta'_{x,1} \\ \theta'_{\alpha,0} + \theta'_{\alpha,1} & 1 & -\theta'_{\gamma,0} - \theta'_{\gamma,1} & \theta'_{y,0} + \theta'_{y,1} \\ -\theta'_{\beta,0} - \theta'_{\beta,1} & \theta'_{\gamma,0} + \theta'_{\gamma,1} & 1 & \theta'_{z,0} + \theta'_{z,1} \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (33)$$

We get ${}^{r(k+i)}\mathbf{T}_{d(k+i)}^{-1}$ by inverting Eq. (33) and omitting the $\theta'_{j_p,1}$

$${}^{d(k+i)}\mathbf{T}_{r(k+i)} = \begin{pmatrix} 1 & \theta'_{\alpha,0} & -\theta'_{\beta,0} & -\theta'_{x,0} - \theta'_{\alpha,0} \cdot \theta'_{y,0} + \theta'_{\beta,0} \cdot \theta'_{z,0} \\ -\theta'_{\alpha,0} & 1 & \theta'_{\gamma,0} & \theta'_{\alpha,0} \cdot \theta'_{x,0} - \theta'_{y,0} - \theta'_{\gamma,0} \cdot \theta'_{z,0} \\ \theta'_{\beta,0} & -\theta'_{\gamma,0} & 1 & -\theta'_{\beta,0} \cdot \theta'_{x,0} + \theta'_{\gamma,0} \cdot \theta'_{y,0} - \theta'_{z,0} \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (34)$$

Using Eq. (30) the left hand side of Eq. (28) can be computed. We concentrate on the translational elements Eqs. (35) to (37).

$${}^{r(k+i)}\alpha_{d(k+i/k)} = \theta'_{\alpha,0} = \frac{-\theta'_{x,1} + {}^{r(k+i)}\mathbf{T}_{r(k+i+1),03} + \theta'_{\beta,0} \cdot \theta'_{z,1}}{\theta'_{y,1}} \quad (38)$$

$${}^{d(k+i)}\mathbf{T}_{d(k+i+1),03} = \theta'_{x,1} + \theta'_{\alpha,0} \cdot \theta'_{y,1} - \theta'_{\beta,0} \cdot \theta'_{z,1} \quad (35) \quad \text{and}$$

$${}^{d(k+i)}\mathbf{T}_{d(k+i+1),13} = -\theta'_{\alpha,0} \cdot \theta'_{x,1} + \theta'_{y,1} + \theta'_{\gamma,0} \cdot \theta'_{z,1} \quad (36)$$

$${}^{r(k+i)}\gamma_{d(k+i/k)} = \theta'_{\gamma,0} = \frac{\theta'_{z,1} - {}^{r(k+i)}\mathbf{T}_{r(k+i+1),23} + \theta'_{\beta,0} \cdot \theta'_{x,1}}{\theta'_{y,1}}. \quad (39)$$

Then Eq. (36) becomes a quadratic equation in $\theta'_{y,1}$

$${}^{d(k+i)}\mathbf{T}_{d(k+i+1),23} = \theta'_{\beta,0} \cdot \theta'_{x,1} - \theta'_{\gamma,0} \cdot \theta'_{y,1} + \theta'_{z,1} \quad (37)$$

Now we have 3 equations to determine ${}^{r(k+i)}\alpha_{d(k+i/k)} = \theta'_{\alpha,0}$, ${}^{r(k+i)}\gamma_{d(k+i/k)} = \theta'_{\gamma,0}$, and $\theta'_{y,1}$ from $\theta'_{x,1}$, $\theta'_{z,1}$, and $\theta'_{\beta,0}$ computed in Sect. 4.3 and from ${}^{r(k+i)}\mathbf{T}_{r(k+i+1)}$ given by the reference path. $\theta'_{x,1}$, $\theta'_{z,1}$, and $\theta'_{\beta,0}$ can be directly taken from the polynomial (3). From Eqs. (28), (35), and (37) we then get

$$\begin{aligned} & \theta'_{x,1} \cdot \theta'_{x,1} + \theta'_{y,1} \cdot \theta'_{y,1} + \theta'_{z,1} \cdot \theta'_{z,1} \\ &= {}^{r(k+i)}\mathbf{T}_{r(k+i+1),03} \cdot \theta'_{x,1} \\ &+ {}^{r(k+i)}\mathbf{T}_{r(k+i+1),13} \cdot \theta'_{y,1} \\ &+ {}^{r(k+i)}\mathbf{T}_{r(k+i+1),23} \cdot \theta'_{z,1} \end{aligned} \quad (40)$$

which is solved by

$$\begin{aligned}
\theta'_{y,1} &= {}^{r(k+i)}\mathbf{T}_{r(k+i+1),13}/2 \\
&\pm \left(\left({}^{r(k+i)}\mathbf{T}_{r(k+i+1),13}/2 \right)^2 \right. \\
&\quad - \left(\theta'_{x,1} \cdot \theta'_{x,1} + \theta'_{z,1} \cdot \theta'_{z,1} \right. \\
&\quad \left. \left. - {}^{r(k+i)}\mathbf{T}_{r(k+i+1),03} \cdot \theta'_{x,1} \right. \right. \\
&\quad \left. \left. - {}^{r(k+i)}\mathbf{T}_{r(k+i+1),23} \cdot \theta'_{z,1} \right) \right)^{1/2}. \tag{41}
\end{aligned}$$

Since $\theta'_{x,1} \approx {}^{r(k+i)}\mathbf{T}_{r(k+i+1),03}$ and $\theta'_{z,1} \approx {}^{r(k+i)}\mathbf{T}_{r(k+i+1),23}$ the square root can be evaluated and yields $\theta'_{y,1} \approx {}^{r(k+i)}\mathbf{T}_{r(k+i+1),13}$. This implies that the sign of the square root has been chosen as the sign of ${}^{r(k+i)}\mathbf{T}_{r(k+i+1),13}$.

$\theta'_{y,1}$ is now used in Eqs. (38) and (39) to determine ${}^{r(k+i)}\alpha_{d(k+i/k)}$ and ${}^{r(k+i)}\gamma_{d(k+i/k)}$. Note that for $\theta'_{y,1} = 0$ the orientation cannot be determined. This is typically the case at the beginning and the end of a trajectory.

4.5 Discussion

- So far, ${}^{r(k+i)}y_{d(k+i/k)} = \theta'_{y,0}$ is not fixed since from almost parallel lines we cannot determine more than 5 dof's. So we can define ${}^{r(k+i(j,l))}y_{d(k+i(j,l)/k)} = 0$.
- In Eq. (14) we neglected orientational sensor corrections when computing the time instants corresponding to sensed line points. With ${}^{r(k+i)}\alpha_{d(k+i/k)}$ and ${}^{r(k+i)}\gamma_{d(k+i/k)}$ we now can evaluate Eq. (20) to determine

$$\begin{aligned}
&{}^{r(k+i(j,l))}\hat{y}_{s(j,l)} \\
&= {}^{r(k+i(j,l))}\alpha_{d(k+i(j,l)/k)} \\
&\quad \cdot {}^{r(k+i(j,l))}x_{n(k+i(j,l),l)} \\
&\quad - {}^{r(k+i(j,l))}\gamma_{d(k+i(j,l)/k)} \\
&\quad \cdot {}^{r(k+i(j,l))}z_{n(k+i(j,l),l)} \tag{42}
\end{aligned}$$

These ${}^{r(k+i(j,l))}\hat{y}_{s(j,l)}$ have to be considered at the right hand side of Eq. (17). Only for small angles of ${}^{r(k+i(j,l))}\alpha_{d(k+i(j,l)/k)}$

and ${}^{r(k+i(j,l))}\gamma_{d(k+i(j,l)/k)}$ and smooth nominal lines this might be neglected, thus disclaiming an iteration as for ${}^{c(k)}\hat{z}_{s(k+i,j,l)}$.

- If high precision requirements are not met by the estimations explained so far, we emphasize to replace the linearized setup of Eq. (8) by the nonlinear equations (see e.g. Eq. (2.57) in [21]), using the parameters of preceding estimations as working point. Such a refined linearized approach may be iterated, if needed, including updates for ${}^{c(k)}\hat{z}_{s(k+i,j,l)}$ and ${}^{r(k+i(j,l))}\hat{y}_{s(j,l)}$.

Example

Let us look at the case in which the orientation of the reference pose does not change with time and translation changes between two sampling steps by a constant vector $(0 \ dy \ 0)^T$. This means

$${}^{r(k+i)}\mathbf{T}_{r(k+i+1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{43}$$

and therefore Eqs. (35) to (37) become

$$\theta'_{x,1} + \theta'_{\alpha,0} \cdot \theta'_{y,1} - \theta'_{\beta,0} \cdot \theta'_{z,1} = 0 \tag{44}$$

$$-\theta'_{\alpha,0} \cdot \theta'_{x,1} + \theta'_{y,1} + \theta'_{\gamma,0} \cdot \theta'_{z,1} = dy \tag{45}$$

$$\theta'_{\beta,0} \cdot \theta'_{x,1} - \theta'_{\gamma,0} \cdot \theta'_{y,1} + \theta'_{z,1} = 0 \tag{46}$$

This may be interpreted as follows:

With $\theta'_{x,1} = \theta'_{z,1} = 0$ we have a constant sensor correction and therefore $\theta'_{y,1} = dy$, $\theta'_{\alpha,0} = 0$, and $\theta'_{\gamma,0} = 0$.

With $\theta'_{\beta,0} = 0$, $\theta'_{x,1} \neq 0$ means a yaw angle $\theta'_{\alpha,0} \neq 0$. So an orientational difference between sensed and nominal lines will cause a translational as well as a rotational sensor correction. As well, with $\theta'_{\beta,0} = 0$ and $\theta'_{z,1} \neq 0$ we get a roll angle $\theta'_{\gamma,0} \neq 0$, meaning that an increasing distance also causes a rotation.

5 Experimental verification

5.1 Implementational aspects

In simulations and real experiments we proved that implementational aspects can be crucial for the success of the method. Therefore here we report some implementational details.

The parameters $\theta_{j_p, i_p}(k)$ of Eq. (3) can be computed by a least-square method. For numerical reasons in contrast to Eq. (3) we use

$$r^{(k+i)} x_{d(k+i/k), j_p} = \sum_{i=0}^{n_p-1} \theta'_{j_p, i_p} \cdot Y^{i_p} \quad (47)$$

with

$$Y = ({}^c y_{r(k+i)} - {}^c y_a) / y_{max} \quad (48)$$

where $y_{max} = \eta_{max} \cdot {}^c z_n$ is the maximal value that can be seen by the camera at a nominal distance. ${}^c y_a$ is the y-component of the distance of the tcp with respect to the camera. With a suitable implementation of the ideal robot ($\mathbf{x}_a \approx \mathbf{x}_d$) and small sensor corrections ($\mathbf{x}_r \approx \mathbf{x}_d$) it yields $Y \approx 0$ for line points corresponding to ${}^c y_{r(k)}$. This speeds up convergence when using the polynomials for the update of Eq. (15). Besides, with Eq. (48) sensed points will approximately satisfy $-1 \leq Y \leq 1$. This is required if the simple monomial representation (47) of the polynomials is used instead of a Bezier or B-spline approach.

According to Eqs. (22) and (23) the distances ${}^c z_{s(j,l)}$ have to be estimated. Since in the nominal case the distances between camera and lines might be computed we can improve our approximation by directly estimating the differences with respect to these nominal distances. This is advantageous if the ${}^c z_{s(j,l)}$ differ substantially from point to point because the frames of the camera and the reference system are not aligned, as in Fig. 6.

As least-square method we use the equations of correction of a Kalman filter. We assume the co-

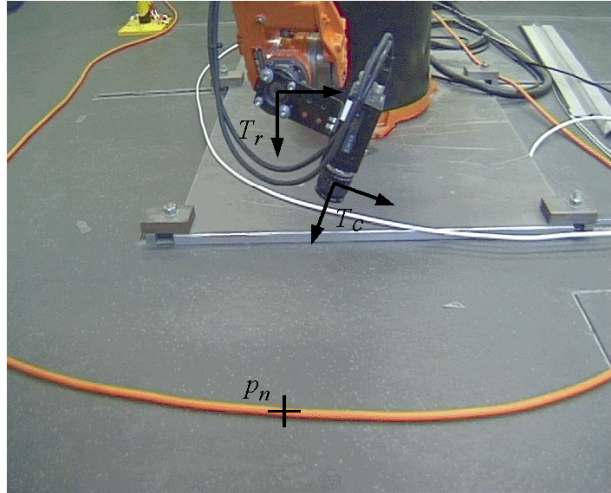


Figure 6: Typical setup with orientation of the tcp and the camera

variance of the noise according to the resolution of the camera and the uncertainty due to motion. The covariance of the unknowns is chosen differently for translational and rotational parameters.

The equations of prediction of the Kalman filter can be used, too, since the polynomial is represented with respect to the camera pose. Instead, so far we restart the estimation for every new image since the information of a single image is sufficient for our computation. Only for α and γ we smooth by using the previously estimated values as initial values.

5.2 Experimental setup

The method is demonstrated at a back and forth motion on a horizontal circular reference path with a radius of 1 m. Accelerations are performed with 4.6 m/s^2 until the reference speed of 0.7 m/s is reached. The target lines are represented by cables which lie on the floor as in Fig. 6. The nominal lines are concentric circles with a spacing of 100 mm.

As robot we use a KUKA KR6/1 with its industrial controller KRC1. The latter uses a 400 MHz Pentium pc for the higher level control and sensing tasks. In standard configuration the feedback controlled robot has a time constant of about 80 ms.

Therefore according to Sect. 2.2 the interface to the ideal robot expects the desired positions of $n_d = 14$ sampling steps of 12 ms. The feedforward controller had been trained before according to Sect. 2.2 (see [18]) using smaller circular paths to identify a linear 4th order model in joint space and to optimize the controller parameters.

For the experiments to be applicable in industrial environments, we restrict to a minimal hardware configuration. As camera a standard monochrome CCIR camera is used. The frame grabber and the vision algorithms run on the same pc as the robot controller. With respect to robot control they have lower priority. Nevertheless every 20 ms a new field of 768×288 pixels can be processed. The standard image rate means that sensor values are available asynchronously with respect to the controller’s sampling steps. So the time instant of the exposure has to be measured to be able to interpolate the current camera pose from the sampling steps of the KRC1. Besides, the images are delayed by about three time steps of 12 ms with respect to the current control step. So the desired path has to be determined for the next $n_d + 3 = 17$ sampling steps.

In our setup the camera is about 0.3 m above the floor. Since the tcp is the center of a tool, we propose a lateral mounting of the camera with a tilt angle such that a point ${}^{c(k)}\mathbf{p}_{n(k)}$ is mapped near the image center (see Fig. 6). This results in a tilt angle of 22 deg for our configuration. With a focal length of 6 mm one pixel in the image center corresponds to a resolution of 0.4 mm at the distance of the cables. Because of the inclined mounting the maximum range that can be seen in direction of motion is about 0.1 m for the worse case. With 0.7 m/s this corresponds to 12 sampling steps. So extrapolation is required to provide 17 time steps for the ideal robot.

The external and internal camera parameters had been identified before [22]. Especially the orientation of the camera is extremely important for extrapolation since a rotation by not more than 0.2 deg results in a displacement of one pixel at the image border.

The polynomials of the sensor corrections are set

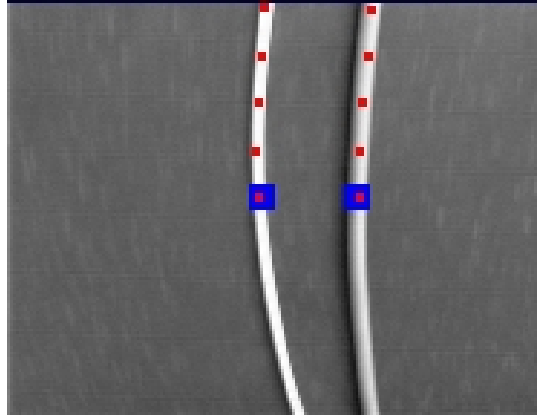


Figure 7: Displayed camera image with (coloured) blocks representing the nominal line positions and five detected line points

with only $n_p = 4$ parameters, each. These parameters are computed from 5 points of each line. This reduces the edge detection algorithms to be applied to at most 3 times 5 regions. Thus the computational amount is small enough so that a single processor is sufficient for robot joint control, evaluation of 50 fields per second for sensor correction, and online display of the scene as in Fig. 7. Even software compensation of lens distortion is possible for the detected points.

5.3 Simulation results

The method is first demonstrated by a task with three lines. In this case the robot is simulated using the linear model of Sect. 2.2 with image acquisition in floating point resolution. The simulated lines differ from the nominal ones only in 2 dof’s (see Fig. 8). This should cause ${}^r x_d$ to increase from 20 mm to about 140 mm. As well ${}^r \alpha_d$ is desired to reach 0.07 rad. All other elements of ${}^r \mathbf{x}_d$ should remain zero. Instead, the simulation shows small sensor corrections in all dof’s. This comes from the estimations which use no a priori information. But the simulated robot follows the lines quite good. The mismatch of the detected line points is in the subpixel range. This verifies the presented algorithm. So in this example the limited visual range, the linearized interpolation between sampling steps, the linearized setup according to Eq. (8) and numerical errors are toler-

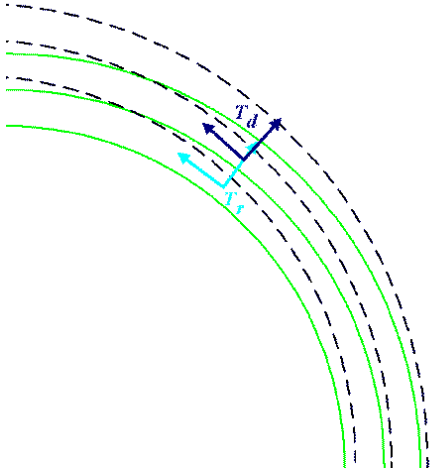


Figure 8: Set-up for simulation with nominal (solid) and simulated (dashed) lines. The reference path lies between the first two solid lines, the desired path of the tcp is between the leftmost dashed lines. Note that for a fixed time instant both paths result in slightly different orientations of the tcp.

able (see Table 2). But it should be noted that for higher orientational sensor corrections we would need an extended approach.

Unfortunately a low image error does not guarantee low control errors of the individual components of the pose vector. The latter result from the configuration, e.g. the distance between camera and lines or the spacing between the lines. Therefore in Table 2 the sensitivities of the individual components have been included, i.e. the partial derivatives. But even sensitivity cannot provide an upper limit for the errors of the desired path

	control error	sensitivity
image	0.2 (0.6) pixels	-
x	0.6 (2.1) mm	0.4 mm/pixel
z	0.2 (0.5) mm	0.5 mm/pixel
α	0.038 (0.052) rad	0.020 rad/pixel
β	0.001 (0.006) rad	0.002 rad/pixel
γ	0.002 (0.003) rad	0.025 rad/pixel

Table 2: Control error (mean values, maximal values in parentheses) when tracking three lines in simulation and geometrical resolution corresponding to one pixel



Figure 9: Experimental setup with KUKA KR6/1 industrial robot and endeffector mounted camera

since e.g. x and β have a very similar influence on the image error. This may cause deviations of x and β by much more than a pixel's equivalent, provided that the sum of their effects is in the sub-pixel range. For all that the results of Table 2 are satisfying.

Generally, a smaller distance between the lines and the camera is advantageous with respect to sensitivity or geometrical accuracy. Unfortunately this implies a smaller visible range for prediction. For a given distance and focal length of the camera accuracy is best if lines are placed so that they take advantage of the available image width.

5.4 Experimental results

Real experiments might be worse because of a miscalibrated camera or because of erroneous compensation of the robot dynamics. Especially elasticity in the robot joints was not considered when our controller had been trained.

When using real image data of individually placed cables as in Fig. 9 the desired path is not as smooth as in Fig. 8. Especially a non nominal

spacing between the lines will cause disturbances since then the computed distances of the camera ${}^{c(k)}\hat{z}_s(k+i,j,l)$ (Eq. (15)) do not match with the real distances that are estimated as ${}^{c(k)}z_s(j,l)$ in Eqs. (22) and (23). Therefore we provided for variations of the spacing of not more than 10 mm around the nominal value of 100 mm.

In addition, small desired changes of the orientation may demand for big changes of the robot joint angles. Therefore in the first experiment we restrict to translational sensor corrections. In this case two lines are sufficient (see Fig. 9 and Exts. 1 and 2). The video clips show that in the image center (near the tcp) the two lines seem unmoved whereas the overall camera shows substantial robot accelerations, especially in the vertical direction. The control errors of Table 3 verify the above assessment. The mean deviation is about 2 pixels. The individual components of the control error are included in Table 3 using the polynomials (2) since the real components of the control error are not known. Further experiments show that the deviations are increasing with robot speed and accelerations of the sensor corrections. In addition, neglecting rotations around ${}^r z_d$ causes a systematic error of factor $1/\cos\alpha$ for the computation of ${}^r z_d$. ${}^r z_d$ might be quite inaccurate anyway, if the nominal lines are close together since its sensitivity is reciprocal to their spacing. On the other side large translational deviations from the reference path are no problem.

In another experiment the robot is desired to follow a single cable which lies on the floor (Fig. 6). The height of the camera with respect to the floor is known. But now tracking of the orientation is desired as well. Exts. 3 and 4 and Table 4 show that the translational error is again about 2 pixels.

	${}^r \mathbf{x}_d$	${}^d \mathbf{x}_a$
image	-	1.7 (5.3) pixels
x	51 (107) mm	0.8 (1.8) mm
z	69 (123) mm	0.9 (2.3) mm

Table 3: Sensor correction and sensed control error (mean values, maximal values in parentheses) when tracking two real lines (rotations are disabled)

	${}^r \mathbf{x}_d$	${}^d \mathbf{x}_a$
image	-	1.8 (6.0) pixels
x	31 (67) mm	0.4 (0.8) mm
α	0.10 (0.21) rad	0.02 (0.08) rad

Table 4: Sensor correction and sensed control error (mean values, maximal values in parentheses) when tracking a single real line

In contrast to the preceding experiment, now in the image center the orientation of the line stays approximately vertical. This is reached by high bandwidth rotations according to the non smooth progression of the line. The limits of the linearized approach and of the robot’s bandwidth will be reached with bigger rotations. On the other side, restriction to translational sensor corrections reduces the control error to 1 pixel. In both cases, bigger translational deviations are no problem.

In both experiments the accuracy is restricted by the unprecise placing of the cables. If the experiment would be the tracking of a misaligned but precisely produced workpiece with smooth edges, control of all 5 dof’s is possible as in the simulation.

Only for substantial rotations an extended approach will be required. Future investigations will study alternatives as e.g. a second order approach instead of Eq. (8).

6 Conclusion

For new industrial applications, robots are required to follow sensed paths with high speed and high accuracy. The paper presents an architecture with which it is possible to fulfill the demand even for tasks with multiple dof’s.

In simple examples we demonstrated that fast robot motion is possible almost without deviations from the sensed path. Remaining inaccuracies in the current set-up come from the discrepancy between limited acceleration abilities of the robot and the coarse layout of the lines.

The paper provides the equations to control the

robot in up to 5 dof's. Anyway, in many applications it will be sufficient to control the *position* of the tcp. Control of the *orientation* is important only for special tools. On the other side three visible lines are not provided by all applications. In most cases we can only evaluate one or two lines, predominantly piecewise straight lines. This can be treated with the approach on hand since only the robot path is represented by polynomials, not the nominal lines.

Since the robot executes sharp bends of the reference path at low speed, vertices, points of intersection of lines, or simply lines that are perpendicular to robot motion might be used to sense the 6th dof as well. This will be integrated into the presented approach by future work.

References

- [1] 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.
- [2] 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.
- [3] U. Frese, B. Bäuml, S. Haidacher, G. Schreiber, I. Schäfer, M. Hähle, and G. Hirzinger. Off-the-shelf vision for a robotic ball catcher. In *Proc. IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pages 1623–1629, Maui, Hawaii, Oct/Nov 2001.
- [4] Y. Nakabo, I. Ishii, and M. Ishikawa. 3D tracking using two high-speed vision systems. In *Proc. IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pages 360–365, Lausanne, Switzerland, Oct. 2002.
- [5] N. Andreff, B. Espiau, and R. Horaud. Visual servoing from lines. In *Proc. IEEE Int. Conference on Robotics and Automation*, pages 2070–2075, San Francisco, CA, April 2000.
- [6] R. Mahony and T. Hamel. Visual servoing using linear features for under-actuated rigid body dynamics. In *Proc. IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pages 1153–1158, Maui, Hawaii, Oct/Nov 2001.
- [7] 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.
- [8] J. Baeten, W. Verdonck, H. Bruyninckx, and J. De Schutter. Combining force control and visual servoing for planar contour following. *Machine Intelligence and Robotic Control*, 2(2):69–75, 2000.
- [9] J. Baeten and J. De Schutter. Combined vision / force control at corners in planar robotic contour following. In *Proc. IEEE/ASME Int. Conference on Advanced Intelligent Mechatronics*, Como, Italy, July 2001.
- [10] 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, Aug. 2002.
- [11] F. Conticelli and B. Allotta. Two-level visual control of dynamic look-and-move systems. In *Proc. IEEE Int. Conference on Robotics and Automation*, pages 3784–3789, San Francisco, CA, April 2000.
- [12] G. Pritschow, A. Horn, and K. Grefen. Dynamisches Verhalten und Grenzen sensorgeführter Industrieroboter mit vorausblickendem Sensor (Dynamic behaviour and limitation of sensor guided industrial robots with look ahead mounted sensor). *Roboter-systeme*, 8:155–161, 1992. in German.
- [13] A. Jacubasch, H.-B. Kuntze, C. Arber, and J. Richalet. Anwendung eines neuen Verfahrens zur schnellen und robusten Positionregelung von Industrierobotern (Application of a new concept of fast and robust position

control of industrial robots). *Robotersysteme*, 3:129–138, 1987. in German.

- [14] D. W. Clarke, C. Mohtadi, and P. S. Tuff. Generalized predictive control - part I. the basic algorithm. *Automatica*, 23(2):137–148, 1987.
- [15] J. A. Gangloff and M. F. de Mathelin. High speed visual servoing of a 6 DOF manipulator using MIMO predictive control. In *Proc. IEEE Int. Conference on Robotics and Automation*, pages 3751–3756, San Francisco, CA, April 2000.
- [16] S. Chroust, E. Zimmer, and M. Vincze. Pro and cons of control methods of visual servoing. In *Proc. 10th Int. Workshop on Robotics in Alpe-Adria-Danube Region*, Vienna, Austria, May 2001.
- [17] F. Lange and G. Hirzinger. A universal sensor control architecture considering robot dynamics. In *Proc. Int. Conf. on Multi-sensor Fusion and Integration for Intelligent Systems*, pages 277–282, Baden-Baden, Germany, August 2001.
- [18] F. Lange and G. Hirzinger. Learning of a controller for non-recurring fast movements. *Advanced Robotics*, 10(2):229–244, April 1996.
- [19] F. Lange. Fast and accurate training of multi-layer perceptrons using an extended Kalman filter (EKFNNet), Sept. 1995. <http://www.robotic.dlr.de/Friedrich.Lange/>.
- [20] F. Lange and G. Hirzinger. Learning accurate path control of industrial robots with joint elasticity. In *Proc. IEEE Int. Conference on Robotics and Automation*, pages 2084–2089, Detroit, Michigan, May 1999.
- [21] J. J. Craig. *Introduction to Robotics - Mechanics & Control*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.
- [22] 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.

A Index to multi-media Extensions

The multi-media extensions to this article can be found online by following the hyperlinks from www.ijrr.org.

Extension	Media type	Description
1	Video	Robot motion along two sensed lines
2	Video	Camera view during motion along two sensed lines
3	Video	Robot motion along a single sensed line
4	Video	Camera view during motion along a single sensed line

B Notation

B.1 General notation

$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}$ Homogenous matrix [21] with translational part \mathbf{p} and rotational matrix \mathbf{R}

${}^w\mathbf{T}_{c(k)}$ Transformation matrix denoting the camera pose at time instant k with respect to the world system in the world system

${}^w\mathbf{p}_{c(k)}$ Vector of translations of ${}^w\mathbf{T}_{c(k)}$

${}^w t_{c(k),ij}$ Element ij of ${}^w\mathbf{T}_{c(k)}$

${}^w\mathbf{x}_{d(k+i/k)}$ Representation of ${}^w\mathbf{T}_{d(k+i/k)}$ by a vector of 3 translational elements and 3 cardan angles (or roll, pitch, yaw angles)

${}^w x_{d(k+i/k),i_1}$ Element i_1 of ${}^w\mathbf{x}_{d(k+i/k)}$

${}^w x_{d(k+i/k)}$ Translational x-component of ${}^w\mathbf{x}_{d(k+i/k)}$

${}^w y_{d(k+i/k)}$ Translational y-component of ${}^w\mathbf{x}_{d(k+i/k)}$

${}^w z_{d(k+i/k)}$	Translational z-component of ${}^w \mathbf{x}_{d(k+i/k)}$	r	Reference system or reference path of the tcp
\mathbf{q}_d	Representation of ${}^w \mathbf{T}_d$ in robot joint angles	s	Sensed (real) line position
$\theta_{j_p, i_p}(k)$	Parameter i_p of the polynomial of output j_p of the image taken at time step k	w	World system
$(\xi(k, j, l), \eta(k, j, l))$	Image point in a fictive image plane with focal distance of 1. This point represents point j on line l at time step k .		

B.2 Indices

a	Actual arm pose (tcp pose)
c	Actual camera system or actual camera pose
c	Commanded joint angles (only used in \mathbf{q}_c)
d	Desired pose of the tcp
$d(k + i/k)$	Desired pose of the tcp of time instant $k + i$, computed from an image of time step k
j	Index of a detected point in the image
j_p	Element of ${}^{r(k+i)} \mathbf{x}_{d(k+i/k)}$
k	Index of the robot sampling step of the exposure
$k + i$	Index of a future robot sampling step
$k + i(j, l)$	Time instant corresponding the sensed point j of line l in the image
l	Index of a line
n	Nominal line
n_l	Number of lines
n_p	Number of parameters of the polynomial