More detailed version of the paper of the International Conference on
Multisensor Fusion and Integration for Intelligent Systems (MFI2001)
Baden-Baden, Germany, August 2001 (not included in the proceedings)

A Universal Sensor Control Architecture Considering Robot Dynamics

Friedrich Lange Gerd Hirzinger

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

Abstract

The paper presents a dynamical sensor control architecture that allows robot arms to perform tasks that with conventional feedback of sensor data fail because of delays or deviations due to the robot dynamics. The architecture distinguishes between robot positional control and refinement of desired positions using vision and / or other sensors. Each of these aspects is designed without the knowledge of the other. Communication between sensors and robot control may be slow and asynchronuous. Experiments show vision based control along a seen line at a speed of 0.7 m/s. Path deviations are about 0.6 mm.

1 Introduction

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 relatively slow sensors as vision systems with low computational power or high task complexity this assumption is reasonable. But there are tasks in which the robot dynamics are important, e.g. in contact situations which require force control. Unfortunately force control methods are very specific and cannot be generalized to other sensors.

This paper presents an architecture for arbitrary sensors, including vision systems as well as force / torque sensors. We allow both, complex sensing tasks and dynamical problems in which communication delays as well as dynamical forces interfere.

A generic architecture including robot dynamics and arbitrary sensors has not been proposed yet. However, there is previous work concerning consideration of dynamics in visual servoing. Corke [5] proposes to estimate the target speed and to use velocity feedforward to improve tracking. Conticelli and Allotta [4] consider the robot dynamics. They investigate decoupling and stabilization of the system by an additional positional feedback loop. Vincze [24] emphasizes the processing architecture. Given a processing power, a parallel architecture at the highest possible image acquisition rate is found optimal. For fixed image rate an on-the-fly configuration is preferred, if possible.

We refer to robots with positional interface because despite some advantages in force control, torque interfaces did not prevail. We assume a robot interface that accepts absolute (joint or cartesian) positions as commands and that outputs the current position, both at the sampling rate of the joint positional controller or slightly slower.

For these type of robots, sensor control usually feeds back sensor signals directly to the robot input commands, e.g.

$$\mathbf{x}_c(k) = \mathbf{x}_c(k-1)+K_p\cdot\Delta\mathbf{s}(k)+K_d\cdot(\Delta\mathbf{s}(k) - \Delta\mathbf{s}(k-1)) \quad (1)$$

where $\mathbf{x}_c(k)$ and $\mathbf{s}(k)$ are the vectors of commanded cartesian positions and sensor signals at time instant k, respectively. $\Delta\mathbf{s}$ means the difference between desired and actual sensor values. In spite of the positional commands we call this approach direct sensor feedback. Sensor signals can be forces as well as deviations sensed by an (image-based) visual servoing algorithm or distances measured with a laser range sensor. The controller gains are represented by matrices K_p and K_d which are tuned according to the configuration. This means that whenever the sensor is modified, the controller has to be designed or adapted again, taking into account that the whole dynamic system has changed. The same is true if an additional sensor is provided. Then the whole system has to be reconsidered to preserve stability.

Another drawback of direct sensor feedback has been formulated by Hirzinger [9] as "space time problem". Without control error no path correction is in-

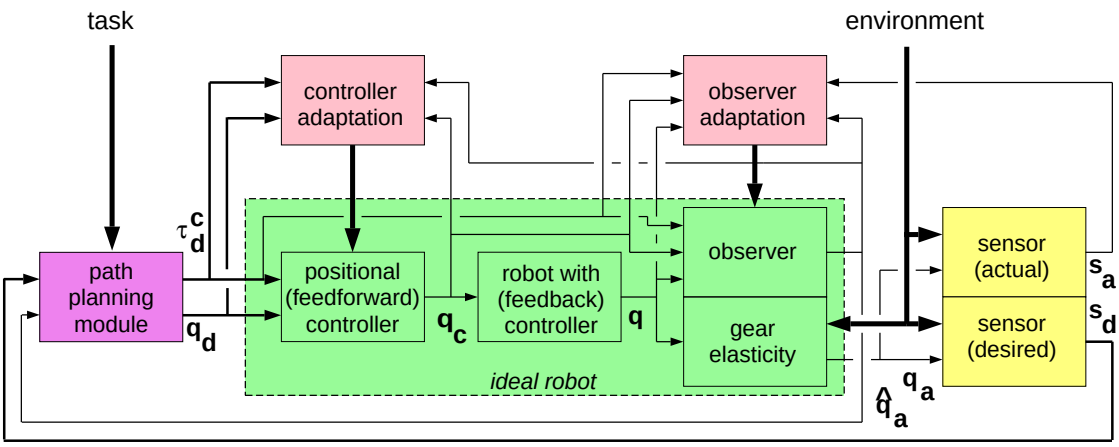

Figure 1: Architecture

duced. And when a path deviation is sensed, some time is required until counter-measures are effective. This delay is due to sensor processing and robot dynamics. It hardly can be reduced because of hardware restrictions.

One limiting factor is the interface between sensor and robot. For usual cascaded control systems the sensor feedback rate is not exceeding the outer loop rate. In contrast for positional control the inner loops are important. They show a much higher rate or even a continuous implementation. DeSchutter et. al. [6] stated that in case of low sensor feedback rate, control with the position as intermediate level is advantageous to low bandwidth feedback to motor torques. This confirms the positional approach of this paper.

The paper is organized as follows: In Sect. 2 we present our dynamical sensor control architecture and the interface between the modules. In Sect. 3 we concentrate on the inner loop, positional control. In Sect. 4 we propose a generic structure for the outer loop. Then in Sect. 5 we show a sample task in which a robot is controlled online according to visually sensed information.

2 Dynamical sensor control architecture

2.1 Overview

The architecture (Fig. 1) distinguishes between control of robot dynamics and sensory feedback. Key idea is the definition of an *ideal robot*. It executes the desired trajectory exactly and without time-delay. The *ideal robot* comprises the real system and additional blocks. We further provided modules for the adaptation of the additional blocks, i.e. modules to make the robot ideal. The *ideal robot* and its realization will be outlined in Sect. 3.

The outer loop interpretes the task description, i.e. there is a module to process sensor information to execute the desired task. This is the topic of Sect. 4.

A special problem is that the actual positions $\mathbf{q}$ provided by the industrial control system do not represent the arm angles $\mathbf{q}_a$ but the motor angles scaled by the gear transmission ratio. Joint angles and scaled motor positions are different if the gears are elastic. And for high accelerations almost all gears show some elasticity. Sect. 3 treats this case, too.

Fig. 1 distinguishes between a sensor to detect the desired path and a sensor to measure the actual robot motion. The latter is only used in combination with elasticity and then only for adaptation and not during task execution. The separation between the two kinds of sensors is logical. Both systems can be of the same hardware, a camera e.g. detecting fixed landmarks as well as features of the desired path. The discrimination is done since the actual arm position is required during task independent adaptation of the *ideal robot*.

Concerning the figure we can state that not only signals are displayed but also dependencies as e.g. the influence of the robot position to sensor data.

2.2 Interface of the *ideal robot*

Input of the *ideal robot* is, first of all, the desired path of the tool center point (TCP). This path is given by the joint positions of the sampling instants. A *cartesian ideal robot* can be defined alternatively (Fig. 2). We discourage from this since inaccuracies of the kinematic transformations are not relevant with sensor control. On the other side additional degrees of freedom (DOF's) of the inverse kinematic transformation can be used in the path planning module.

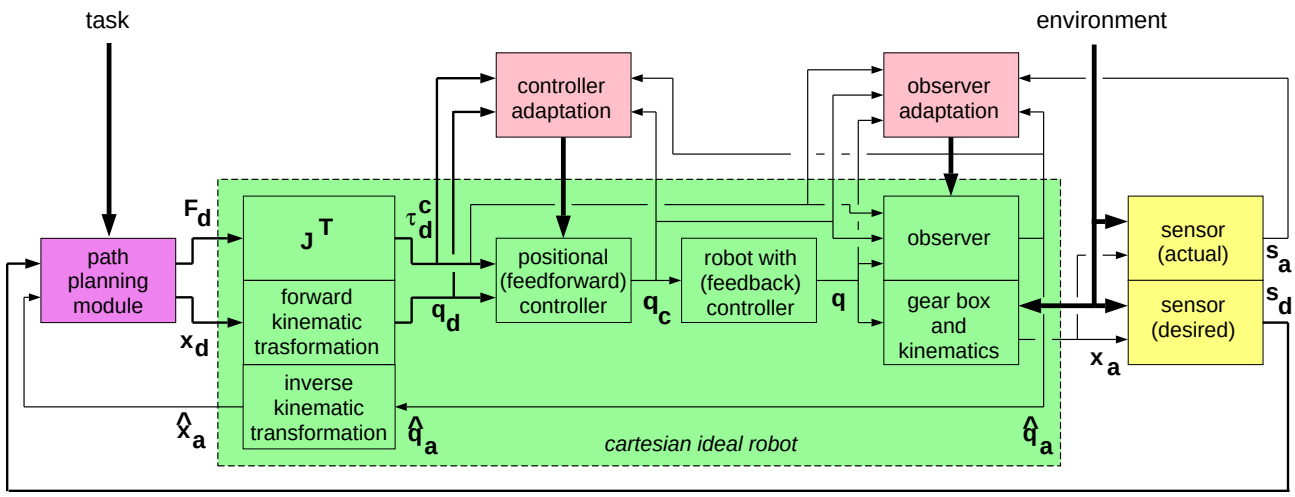

Figure 2: Alternative architecture with *cartesian ideal robot*

Output of the *ideal robot* is the actual position of the TCP, represented in joint space, too.

At the output only the current position $\hat{\mathbf{q}}_a(k)$ at timestep k is defined. In addition at the input we provide several desired positions $\mathbf{q}_d(k) \cdots \mathbf{q}_d(k+n_d-1)$. Therefore the lines in Figs. 1 and 2 are drawn bold face. The use of single desired setpoints is not sufficient to achieve the behaviour of an *ideal robot*, i.e., to avoid path deviations, velocity errors, or time-delays. n_d is in the order of magnitude of the number of sampling steps corresponding to the main time constant of the robot.

In case of contact forces acting on the robot, these forces or their expected (desired) values $\mathbf{F}_d$, or τ_d^c for the joint values respectively have to be known by the *ideal robot*. And this not only for the current time instant but for the same planning horizon of some sampling steps and therefore drawn as bold face line, too. Only with these torques the information is sufficient to execute the desired motion correctly i.e. to reach $\mathbf{q}_a(k) = \mathbf{q}_d(k)$.

2.3 Discussion

It is possible to use the optimized positional controller without sensors. As well sensory feedback may be applied without the realization of an *ideal robot*, i.e. without the use of future desired positions.

The latter may be relevant when sensor data are used because then future timesteps of the desired path are not available, depending on the sensor type. Extrapolation of measured sensor signals may be a solution (see Sect. 4.3) or, in case of vision, evaluation with respect to future TCPs, not only for the current one.

In our experiments the results were only slightly inferior without feedforward controller ($n_d = 1$). In force control this results from the fact that predictions are not reliable. In contrast in vision prediction is almost sufficient if the refinements are computed for the current desired position (usually at the image border) and not for the current (delayed) actual position (usually in the image center). With $n_d = 1$ path accuracy is dependent on the quality of the industrial control system.

Instead of the internal structure of the *ideal robot* with adaptive elements we could design a model based implementation of controller and observer without the use of the industrial robot controller (see Sect. 3.1). In this case no adaptation modules are required. Then another interface of the *ideal robot* is reasonable. It uses derivatives of the desired path at the current timestep instead of future positions.

The dynamical sensor control architecture preserves stability as long as the positional robot feedback controller is stable. Sensor and position feedback in Figs. 1 and 2 seem to affect stability, but actually the signals describe the location of the target, not of the robot. Stability is touched only when the robot position itself is fed back. This may be the case if the transformation from sensor values to desired robot positions is totally wrong because of an erroneously assumed distance in image processing or an incorrectly measured compliance in force control (details are in [11]).

Besides, intentionally or unintentionally (due to noise) high bandwidth desired paths may excite the elastic modes of the robot. If no sensor for the actual

arm position and no observer is provided, such oscillations have to be avoided since they may yield incorrect sensing (for details see Sect. 4).

3 Realization of the *ideal robot*

3.1 Overview

This section discusses the inner loop if the total architecture is regarded as an inner loop / outer loop implementation of sensor control. So we design a controller which enables the robot to accurately execute all desired paths nearby the working point in use.

In Fig. 1 the *ideal robot* itself has three modules. This results if the architecture shall be easily implemented into usual industrial controllers. The block in the center represents the mechanical robot and its standard industrial control system which controls the actual joint angles $\mathbf{q}$ according to joint commands $\mathbf{q}_c$ (usually called desired values). The joint commands and the measured angles $\mathbf{q}$ represent the so called sensor interface provided by robot manufacturers.

The industrial control 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.

Therefore we propose to extend the controller with a module that can process the above defined planning horizon of n_d sampling steps of the desired path. This will be done in Sect. 3.2. The controller is set adaptive to be able to be adapted to the robot dynamics. In most cases it turns out that it is sufficient to use feedforward control and no feedback of the actual positions. Since the *ideal robot* is independent of the task to be executed, it is sufficient to perform the adaptation once, at the installation of the robot. Additional adaptations can compensate for abrasion.

If elasticity has to be considered we have to use observed values for $\mathbf{q}_a$ during adaptation of the controller. The observer is set adaptive, too.

For the architecture according to Fig. 2 we also have to compensate (static) errors of usually unprecise kinematic transformations by modifying the Denavit-Hartenberg parameters. In addition, not only dynamic motion but also static accuracy is influenced by elasticity [7]. But static effects are not relevant if path planning is sensor based since static deviations resemble to differences between nominal and sensed paths. Nevertheless robots with abolute static accuracy facilitate control since the influence of static errors is only approximately constant because of nonlinear transformations.

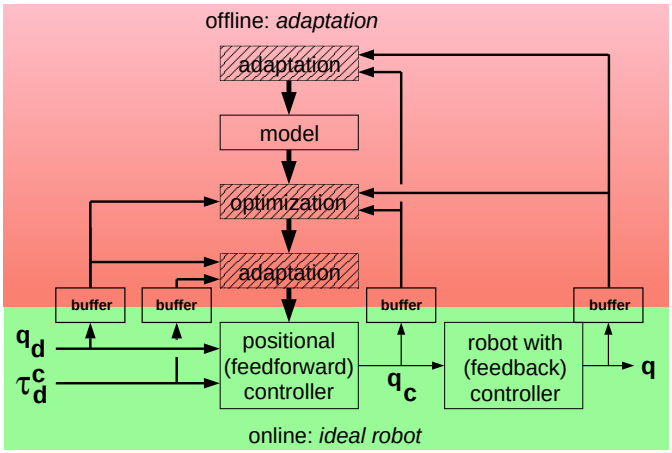

Figure 3: Structure of adaptation (for rigid systems)

3.2 Adaptive feedforward control to improve accuracy

In this article we discuss only dynamical path accuracy. Model based approaches [2] interfere into the existing control system by directly commanding joint torques. This implies high sampling rates and precise model knowledge. Therefore it will not be followed further. Instead, we implement the modules "positional controller" and "controller adaptation" from Figs. 1 or 2. To do so we have different possibilities. We could simply adapt the parameters of a usual controller or we could use predictive control [8] where the controller is designed using an identified model. Instead, now we will summarize an iterative method [12] which was previously designed by the authors. This method allows iterative reduction of remaining control errors (Fig. 3).

The adaptation is executed in three steps (hatched blocks) using selected trajectories. After the first run the robot (including the industrial (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 controller parameters to fulfill the optimal trajectory (3rd step). This exceeds *iterative learning control* (ILC) [17, 3] since there no controller is adapted. For the *ideal robot* we need such a controller which, at least in the neighbourhood of the current working point, is able to control arbitrary paths using the inputs defined in Sect. 2.

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 com-

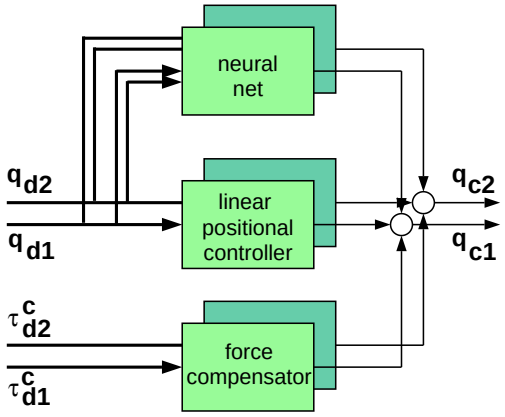

Figure 4: Modules of the positional controller for a 2-axis robot (Inputs are understood to represent scalar desired values of the current and several future sampling steps.)

mands $\mathbf{q}_c$ ist not realizable with sufficient accuracy. So we use that the robot is almost linear because the invariant motor inertia is dominant compared with the position dependent arm inertia. Therefore nonlinear parts may be scaled finer. So we propose an extended setup with linear approach, neural nets to compensate for couplings and, in case of contact, an additional controller to compensate for external forces (Fig. 4). As neural nets we use multilayer perceptrons with well suited training algorithms [21, 10].

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.

3.3 Consideration of gear elasticity

In the adaptive architecture of Fig. 1 elasticity has only to be considered if the position of the TCP is not directly measurable because the measured motor angles are not meaningful. Then two cases have to be distinguished:

In the first case an external measurement device is available and it is able to measure all DOF's online. Then we do not need an observer but we can transform the sensor signals to joint positions. Then adaptation and specification of the desired path are possible as usual.

In the other case the use of the measuring device is restricted, e.g. to few DOF's or to offline evaluation. This may be sufficient for adaptation. In [13] a method is presented in which the positional controller is adapted using offline available data. The measurements cover only 2 DOF's that are used to estimate all 6 joint values. But, at least to specify the desired path, an observer has to be designed or to be adapted which provides the required information during task execution.

Such an observer has to be better than the model in Fig. 3 since elasticity may excite almost undamped oscillations. On one hand the planning horizon of the observer has to cover several seconds if it is limited to the inputs shown in Fig. 1. On the other hand high accuracy is required to prevent a phase shift with respect to the real oscillation. This may demand a model-based approach or additional sensors (as in [1]). There are further investigations necessary concerning this point.

For the present we propose to assume $\hat{\mathbf{q}}_a \approx \mathbf{q}_d$ which is valid for a good positional controller which has been adapted using offline available measurements. Then online path planning as in [14] may be possible in spite of elasticity.

4 Sensor-based path planning

4.1 Generic structure

The task description is usually given in natural language as e.g. "apply glue at 1 cm from the edge". This message has to be transferred to a desired path for the *ideal robot*. If available, we use a world model that, in the example, specifies the nominal location and shape of the edge. In addition, sensor data are evaluated if this is useful for the completion of the task. With other words a nominal path is computed and a control law to modify it in relation to sensor data if there are uncertainties in the world model, in our example, if the location or the shape of the edge may vary. So sensor data specify the actual edge and thus indirectly the desired path.

Fig. 5 shows a generic structure which is applicable to force control [11] as well as to control with fixed or robot mounted cameras [14] or other sensors. In contrast to direct sensor control as e.g. classical force control this structure is not limited to single sensors. Instead, different types of sensors may be fused as long as each sensor outputs a positional difference.

Fig. 5 begins with a stored path given by points $\mathbf{x}_p$ in cartesian space, e.g. four points for a rectangle, as it is usual for industrial robots. We added the corresponding desired sensor values $\mathbf{s}_p$. Trajectory generation means that a desired velocity profile is used to interpolate the points and the corresponding sensor values for all sampling instants. Usually the overall time for the trajectory is minimized [22] or a trapezoidal velocity profile is prescribed. The result is a

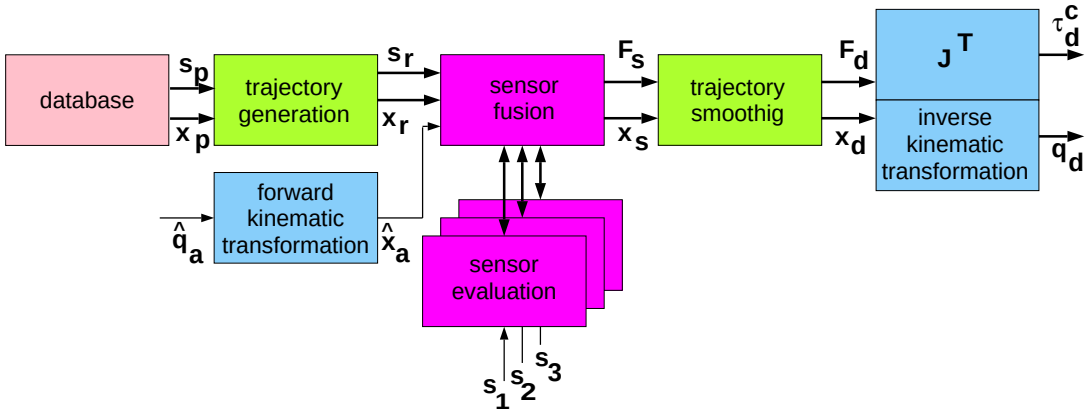

Figure 5: Generic path planning module

reference path $\mathbf{x}_r$ with reference sensor values $\mathbf{s}_r$.

In the next block sensor data are integrated. The idea of the dark blocks in Fig. 5 is that in contrast to usual control we compare reference positions $\mathbf{x}_r$ and reference sensor values $\mathbf{s}_r$ with the actual values. We then get a sensed desired trajectory.

At least for noisy sensors the sensed desired path $\mathbf{x}_s$ has to be smoothed since the *ideal robot* tries to execute even big accelerations without error. Note that sensor values are not more computed for the sensed desired trajectory except for the force. This is because only (external) forces are required by the ideal robot. Other sensors are not more essential after sensor fusion. So we restrict to the desired path $\mathbf{x}_d$ and the desired contact force $\mathbf{F}_d$.

When using the architecture depicted in Fig. 1 rather than the cartesian approach of Fig. 2, we have to add the kinematic transformations. We prefer an absolute inverse kinematic transformation to the approximation by the inverse Jacobian since the latter may result in a small drift with respect to uncontrolled DOF's.

4.2 Evaluation of sensor data

The dark blocks in Fig. 5 will be further outlined. We assume that the actual sensor values in combination with the actual position of the TCP uniquely define a sensed desired position $\mathbf{x}_s$ of the TCP. This is a generic formulation which also allows the use of vision data [14], force control [11] or impedance control [23], and even general sensor based insertion tasks as in [19].

A simple approach interpretes the difference between the desired and the actual sensor value to be a positional error. The sensed desired position is then the sum of the estimated actual position $\hat{\mathbf{x}}_a$ and the positional error $\Delta\mathbf{x}_a$ or, in unsensed DOF's, the reference trajectory $\mathbf{x}_r$ respectively.

$$\mathbf{x}_s = \begin{cases} \hat{\mathbf{x}}_a + \Delta\mathbf{x}_a & \text{in sensed direction} \\ \mathbf{x}_r & \text{else} \end{cases} \tag{2}$$

For force control with known elasticity E the difference between the reference force $\mathbf{F}_r$ and the actual force $\mathbf{F}$ in direction of the actual force vector yield the modification from the actual trajectory to the sensed desired trajectory.

$$\Delta\mathbf{x}_a = \frac{\mathbf{F}_r - \mathbf{F}}{E} \tag{3}$$

For the task of following a visually sensed edge at known distance d, an edge point $s(\mathbf{x}_r(k))$ in the image plane with a focal length of f is detected which corresponds to the point $\mathbf{x}_r(k)$ of the reference path. The image point is compared with the image of the edge point $s_r(\mathbf{x}_r(k))$ in the reference scenario (nominal world model).

$$\Delta\mathbf{x}_a(k) = (s_r(\mathbf{x}_r(k)) - s(\mathbf{x}_r(k)))\frac{d}{f} \tag{4}$$

If multiple features can be sensed it is further possible to extract the distance d or additional DOF's or timesteps of the sensed desired path $\mathbf{x}_s$ from the image (see Fig. 7).

4.3 Sensor fusion of asynchronuous data

For a more sophisticated approach Eq. (2) is rewritten as the equation of fusion Eq. (5) and the equation of prediction Eq. (6).

$$\Delta\mathbf{x}_r = \begin{cases} \Delta\mathbf{x}_a - (\mathbf{x}_r - \hat{\mathbf{x}}_a) & \text{if sensable} \\ 0 & \text{else} \end{cases} \tag{5}$$

$$\mathbf{x}_s = \mathbf{x}_r + \Delta\mathbf{x}_r \tag{6}$$

The fall differentiation is meant spatial. Computed values for $\Delta\mathbf{x}_r$ remain valid until new sensor values are received. The equation of fusion can be extended to different orthogonal sensors each providing a difference $\Delta\mathbf{x}_{ri}$ from the reference path by measuring the sensed control difference $\Delta\mathbf{x}_{ai}$ at the estimated actual position $\hat{\mathbf{x}}_{ai}$. The sensed desired position $\mathbf{x}_s$ in Fig. 5 is the sum of reference position and sensable deviations $\Delta\mathbf{x}_{ri}$ of the setup with respect to the reference.

In case of non-orthogonal sensors the equation of fusion has to be replaced by a filter, e.g. an extended Kalman Filter (EKF) or an optimization as in [15], which allows to consider the accuracy of the individual sensors as well as their contribution to the respective DOF's.

In contrast to efficient methods in literature (see e.g. [18, 16]) our method does not require high sensor bandwidth supposed that the differences between actual and expected sensor values are time-invariant or slowly varying. As extreme example single sensing is sufficient, e.g. a single force vector or a single camera snapshot. This will be used to update the sensed desired trajectory according to Eq. (6) and then to control the robot with the inner loop control rate. So in many cases integration of vision according to Eqs. (4) and (5) is sufficient using the 50 Hz field rate of off-the-shelf CCD-cameras. Then the *ideal robot* will track the desired trajectory $\mathbf{x}_d$ with high accuracy, even at high speed.

Asynchronuous sensing is possible, too. Then instead of projecting the sensed values to the next sampling step as in [20] we compute the position of the time instant of sensing and then evaluate the measured sensor data for the latter time instant.

4.4 Discussion

The crucial difference of Eq. (1) with respect to our approach is that $\mathbf{x}_c(k) - \mathbf{x}_c(k-1)$ is set proportional to $\Delta\mathbf{x}_a(k)$ and not $\mathbf{x}_d(k)$ in relations to $\Delta\mathbf{x}_r(k)$. Delays cause multiple corrections $\mathbf{x}_c(k) - \mathbf{x}_c(k-1)$ before the control error $\Delta\mathbf{x}_a$ is reduced. Then overshooting will occur except for very low control gain, which results from the fact that the actual arm position $\mathbf{x}_a$ is not used. In addition extrapolation because of delayed or asynchronous sensing is not possible since a reference path $\mathbf{x}_r$ is not defined. Therefore the approach of Eq. (1) is sensitive to delays in signal processing. Summarizing, in contrast to our architecture the usual method of Eq. (1) requires high sampling rates and small delays for both, sensors and robot interface.

The existence of a reference trajectory seems to be a restriction for generic tasks. Reconsidered such a reference may be degenerated to a single point, a working point or starting point of motion. E.g. contour following in [11] needs no reference trajectory. Instead, the local desired velocity is computed using only the actual sensor values.

The demand for all sensors to measure positions requires calibration of the sensors. So e.g. in case of a force / torque sensor the natural compliance of the system has to be known or at least to be measurable. This may be a handicap of the proposed architecture since direct sensor control needs no calibrated sensors because the transformation from sensor values to commands is part of the controller. On the other hand, calibration can simply be reached by moving the sensor and comparing the sensor values with the simlutaneously measured actual positions of the TCP. Positional calibration of force / torque sensors requires a minimal compliance which fortunately is always present if the robot can be force controlled by a positional interface [11].

5 Experiments

As experiment we considered line following [14] which is similar to industrial applications as laser cutting or applying glue where the target motion is defined relative to one or more edges and the positions or the shapes of the edges differ from execution to execution. Such task should be executed with high speed insuring high accuracy at the same time. In our ex-

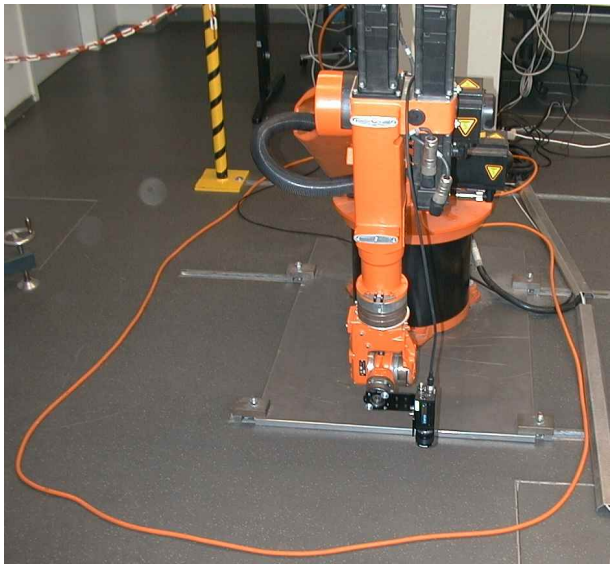

Figure 6: Experimental setup with KUKA KR6/1 industrial robot and endeffector mounted camera

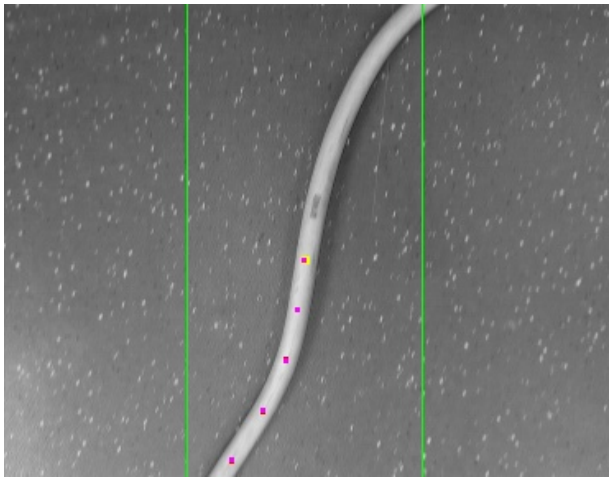

Figure 7: Displayed camera image with markers representing the used window, the image center, and five detected line points

periment a single line is tracked (see Figs. 6 and 7), represented by a polynom which is computed from 5 points. Camera fields are grabbed and evaluated according to the architecture of Figs. 1 and 5 without assuming gear elasticity. The computational amount is so small that the robot controller is able to refine the desired path every 20 ms without using additional computing power. The mean control error is about 0.6 mm for a top speed of 0.7 m/s.

Other applications are pick-and-place operations from coarsely known pickup positions to coarsely positioned insertion places. In this case the advantage of the generic structure of the path planning module is that during approaching no extra stop is required for image acquisition or for hand over of control to a force controller. Instead, visually measured target position and contact forces / moments are fused to reach that not only jamming is prevented but additional DOF's are controlled during insertion.

6 Conclusion

The article presents an architecture which defines sensor control as the combination of robot positional control and path planning. Both parts are designed independently without knowledge of the other. In contrast to usual methods we do not require high bandwidth signal processing.

The reported experiments and cited papers of the authors prove that accuracy of an elastic KUKA robot can be improved substantially for a fixed path, leaving a reduced path error of 30% with respect to the standard industrial controller. Tracking deviations with respect to a seen line are about 0.6 mm at a speed of 0.7 m/s.

References

[1] T. Alban and H. Janocha. Dynamic calibration of industrial robots with inertial measurement systems. In *Proc. European Control Conference*, Karlsruhe, Germany, August / Sept. 1999.

[2] A. Albu-Schäffer and G. Hirzinger. State feedback controller for flexible joint robots: A globally stable approach implemented on DLR's lightweight robot. In *Proc. IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, Takamatsu, Japan, Oct./Nov. 2000.

[3] Y. Chen. Updates on iterative learning control 2000, 2000. http://www.crosswinds.net/˜ygchen /ilc2000/.

[4] 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, California, April 2000.

[5] 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.

[6] J. De Schutter, H. Bruyninckx, H. Zhu, and M. W. Spong. Force control: A bird's eye view. In B. Siciliano and K. Valvanis, editors, *Control Problems in Robotics and Automation*, pages 1–17. Springer Verlag, 1998.

[7] P. Drouet, S. Dubowsky, and C. Mavroidis. Compensation of geometric and elastic deflection errors in large manipulators based on experimental measurements: Application to a high accuracy medical manipulator. In J. Lenarčič and M. L. Husty, editors, *Advances in Robot Kinematics: Analysis and Control*. Kluwer Academic Publishers, 1998.

[8] 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, California, April 2000.

[9] G. Hirzinger. Adaptiv sensorgeführte Roboter mit besonderer Berücksichtigung der Kraft-Momenten-Rückkopplung. *Robotersysteme*, 1:161–171, 1985.

[10] F. Lange. Fast and accurate training of multilayer perceptrons using an extended Kalman filter (EKFNet), Sept. 1995. http://www.robotic.dlr.de/Friedrich.Lange/.

[11] 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.

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

[13] 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.

[14] F. Lange, J. Langwald, and G. Hirzinger. Predictive feedforward control for high speed tracking tasks. In *Proc. European Control Conference*, Karlsruhe, Germany, August / Sept. 1999.

[15] J.-W. Lee and S. Lee. A new data fusion method and its application to state estimation of nonlinear dynamic systems. In *Proc. IEEE Int. Conference on Robotics and Automation*, pages 3525–3530, San Francisco, California, April 2000.

[16] R. Maaß, V. Zahn, M. Dapper, and R. Eckmiller. Hard contact surface tracking for industrial manipulators with (SR) position based force control. In *Proc. IEEE Int. Conference on Robotics and Automation*, pages 1481–1486, Detroit, Michigan, May 1999.

[17] K. L. Moore and J.-X. Xu. Special issue on iterative learning control. *Int. Journal of Control*, 73(10):819–823, 2000. (other papers on ILC see pp. 824-999).

[18] Y. Nakabo, M. Ishikawa, H. Toyoda, and S. Mizuno. 1ms column parallel vision system and it's application of high speed target tracking. In *Proc. IEEE Int. Conference on Robotics and Automation*, pages 650–655, San Francisco, California, April 2000.

[19] W. S. Newman, M. S. Branicky, H. A. Podgurski, S. Chhatpar, L. Huang, J. Swaminathan, and H. Zhang. Force-responsive robotic assembly of transmission components. In *Proc. IEEE Int. Conference on Robotics and Automation*, pages 2096–2102, Detroit, Michigan, May 1999.

[20] C. Robl and G. Färber. System architecture for synchronizing, signal level fusing, simulating and implementing sensors. In *Proc. IEEE Int. Conference on Robotics and Automation*, pages 1639–1644, San Francisco, California, April 2000.

[21] P. van der Smagt. Feed-forward networks, 1998. http://www. robotic.dlr.de/Smagt/research/feed-forward/.

[22] B. Tondu and S. A. Bazaz. The three-cubic method: An optimal online robot joint trajectory generator under velocity, acceleration, and wandering constraints. *The International Journal of Robotics Research*, 18(9):893–901, 1999.

[23] T. Valency and M. Zacksenhouse. Instantaneous model impedance control for robots. In *Proc. IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, Takamatsu, Japan, Oct./Nov. 2000.

[24] M. Vincze. Dynamics and system performance of visual servoing. In *Proc. IEEE Int. Conference on Robotics and Automation*, pages 644–649, San Francisco, California, April 2000.