IEEE Int. Conf. on Robotics and Automation ICRA2005, Barcelona, Spain, April 2005

Stability Preserving Sensor-Based Control for Robots with Positional Interface

Friedrich Lange and Gerd Hirzinger
Deutsches Zentrum für Luft- und Raumfahrt e. V. (DLR)
Oberpfaffenhofen, D-82234 Weßling, Germany
http://www.robotic.dlr.de/Friedrich.Lange/

***Abstract*— When industrial robot arms are controlled using sensor data the performance is dependent on the sensor sampling rate, on delays in signal processing, and on the robot dynamics. The paper presents an approach in which control is inherently stable as long as the time instant of sensing is known, independently of delays. In addition to sensor data the method uses the actual robot pose to compute a desired pose which is then controlled by the existing positional control loop. Updated sensor data affect the system as a refined target for positional control. So the positional control and the use of sensor data are decoupled. This is useful for the integration of a priori information on the task. The method is applicable especially for force control tasks as contour following and for visual servoing.**

***Index Terms*— force control, visual servoing, tracking, stability, control architecture**

I. Introduction

The paper discusses the problem of controlling an industrial robot using on-line sensor data originating from a force/torque sensor, a camera, or another kind of sensor. In contrast to literature which distinguishes between papers on force control as [1], [2] and on visual servoing as [3] the authors emphasize that there are common features for all kinds of "sensor based control." Such a global approach will probably be useful to facilitate sensor fusion.

This paper considers contour following tasks using a force/torque sensor (Fig. 1) and visual servoing along curved lines (Fig. 2). A direct feedback of sensor data as in [1] is not designed in most industrial robot controllers. Therefore for both kinds of applications usually a PD control law as

$$\mathbf{x}_c(k) = \mathbf{x}_c(k-1) + \mathbf{K}_p \cdot \Delta\mathbf{x}(k) + \mathbf{K}_d \cdot (\Delta\mathbf{x}(k) - \Delta\mathbf{x}(k-1)) \tag{1}$$

is used, where $\mathbf{x}_c$ is the commanded Cartesian pose of the robot, and $\Delta\mathbf{x}$ is the control error of the sensor data. In contrast to our notation $\mathbf{x}_c$ is commonly referred as the desired value for the robot feedback controller. It is accessible in most industrial robot interfaces, as in [4].

The initial value of $\mathbf{x}_c$ is taken from

$$\mathbf{x}_c(0) = \mathbf{x}_a(0) \tag{2}$$

which represents the actual pose of the tool center point (tcp).

In this paper we assume that the sensed control error is computed from the desired sensor data $\mathbf{s}_d$ and the actual sensor data $\mathbf{s}_a$ by

$$\Delta\mathbf{x} = \begin{cases} \begin{pmatrix} \mathbf{R}_s \cdot (\mathbf{s}_d - \mathbf{s}_a) \\ \mathbf{0} \end{pmatrix} & : \text{ measured components} \\ \mathbf{0} & : \text{ other components} \end{cases} \tag{3}$$

where $\mathbf{R}_s$ is the rotation matrix which transforms the sensor data to the robot base system. This notation restricts sensor data to provide the position. For the sake of clarity, sensors for the orientation are not discussed here since this would afford a more complex notation.

In these equations sensor data are understood as Cartesian positional values that are measured in a sensor fixed

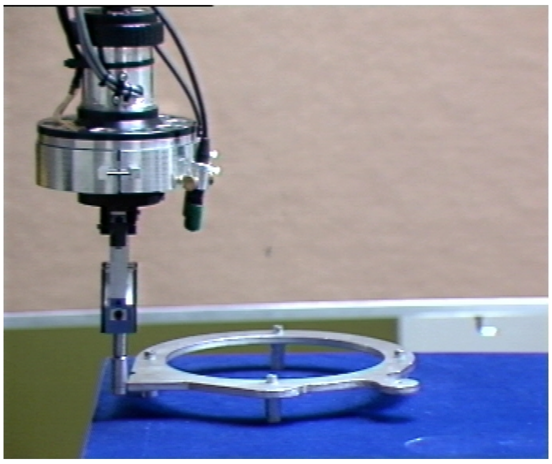


Fig. 1. Force controlled contour following

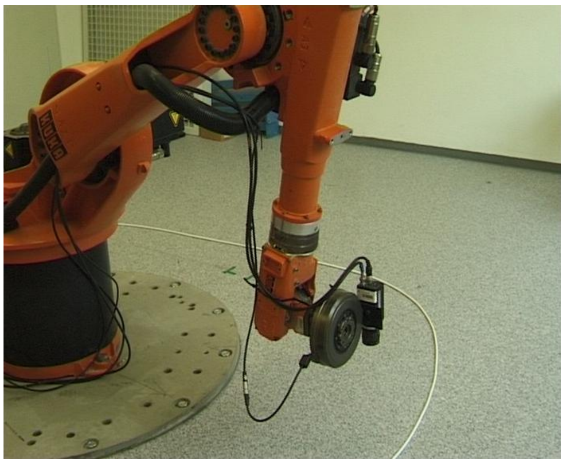
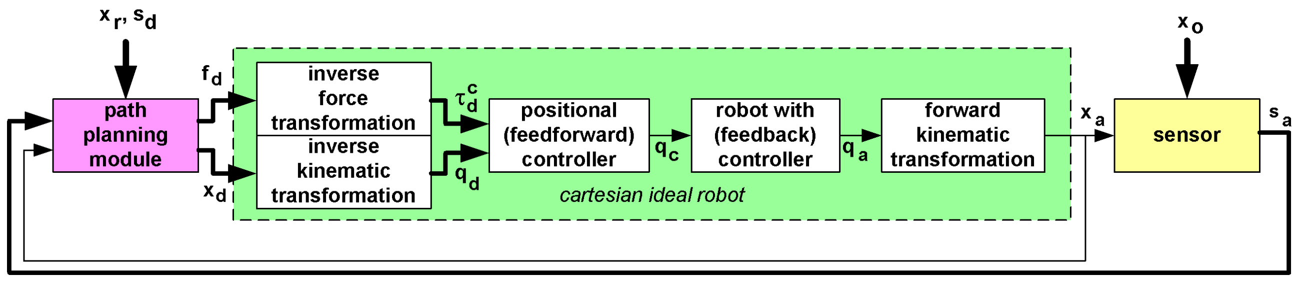


Fig. 2. Visual servoing along a cable



x_r, s_d
path planning module
f_d
x_d
inverse force transformation
inverse kinematic transformation
τ_d^c
q_d
positional (feedforward) controller
q_c
robot with (feedback) controller
q_a
forward kinematic transformation
cartesian ideal robot
x_a
x_o
sensor
s_a

Fig. 3. Control architecture with positional control as inner loop (green) and the determination of the desired path as outer loop

coordinate system. This means that original sensor data as forces or image data are transformed.

In fact, most force/torque sensors as [5] measure positional deviations and then compute forces and torques using the known compliance. Forces can then be transformed back to positions using the elasticity of the robot, the hand, or the environment. With a compliant force/torque sensor at the wrist the Cartesian 6x6 compliance $\mathbf{E}_s^{-1}$ of the sensor is dominant, yielding

$$\mathbf{s}_a = \begin{pmatrix} \mathbf{I} & \mathbf{0} \end{pmatrix} \cdot \mathbf{E}_s^{-1} \cdot \mathbf{f}_a \tag{4}$$

where $\mathbf{f}_a$ is the actual force/torque vector expressed in the sensor system. For stiff sensors refer to [6] for a more detailed discussion.

In the same way visual servoing algorithms - position-based as well as image-based - yield positional differences $\Delta\mathbf{x}$. For the scope of this paper we consider only small differences so that the components are independent from each other.

The drawback of the approach (1) is that delays in signal processing or dynamical delays of the robot worsen the performance. With a time-delay of several sampling steps a sensed static control error will increase the value of $\mathbf{x}_c(k)$ many times before the sensor signal changes. Then the robot will move and - in relation to the gains - either overshoot or creep to the desired position rather slowly. So big time delays or low sensor sampling rates cause a rather unsatisfactory step response. Unmodeled dynamical delays can affect stability.

Another drawback is that the computation of the gain matrices $\mathbf{K}_p$ and $\mathbf{K}_d$ depends on the characteristics of the sensor interface as well as on the robot dynamics since both contribute to the total delay. So the total system is not modular.

In this paper we present a different approach in which the robot dynamics and the sensor characteristics are decoupled. As usual, with this approach the performance will decrease inevitably with big time delays. But in contrast to the PD control method of (1) our approach is inherently stable as long as the time delays are **known**, irrespective of their amount. The approach distinguishes between the computation of the desired pose using sensor data, and the positional control using the internally measured joint positions.

Previous work exists by Zhang et al. [7] who as well compensate delays due to image processing and data transfer. In contrast to the approach presented here they do not include delays caused by the robot dynamics.

On the other side Conticelli and Allotta [8] present a two level architecture which separates visual feedback from the robot's dynamics. The robot is controlled in the lower level which thus represents a linear process for the outer loop controller.

In all these approaches the inner loop does not totally compensate the unwanted robot characteristics. So a decoupling between control of the robot and task specific execution of the sensor task is not reached.

In Sect. II our basic method is presented. Sect. III then deals with the positional control. The integration of a priori information and special applications are discussed in Sects. IV and V. The method is finally demonstrated in Sect. VI using an example of visual servoing.

II. Fundamental approach for sensor-based control

The method is based on the separation between the existing positional control loop and the integration of sensor data, as shown in Fig. 3. Sensor data are used to compute a desired pose of the robot.

In Fig. 3 positional control is executed by joint controllers since on joint level there are less couplings than in Cartesian space. The kinematical transformations between the Cartesian pose vectors $\mathbf{x}$ and the vectors of joint positions $\mathbf{q}$ are known since we do not consider redundant robots or singular configurations. $\mathbf{q}_d$ and $\mathbf{q}_a$ are the desired and the actual joint values respectively. $\mathbf{q}_c$ is the vector of commanded positions which is the usual input to industrial control systems. It is computed using an adaptive feedforward controller [9], [10] which processes in each sampling step the desired values of the current and of some future sampling steps (bold face lines in Fig. 3 denote values of several sampling steps). A sufficient number of future values theoretically allows the realization of an ideal robot, i.e. a robot which precisely executes the desired trajectory without time delays. For fixed programmed paths without using sensor data this is valid as has been demonstrated in [9].

If sensor data are used, however, future desired values are not known and coarse predictions will cause control errors. Accurate predictions and therefore precise tracking has been shown for a kind of visual servoing tasks [11] but in this paper we treat the general case. Special cases will be discussed in Sect. IV. So far, as prediction we use an extrapolation of order zero, i.e. the currently sensed desired pose is taken for future values as well. This yields

$$\mathbf{x}_c(k) = \mathbf{x}_d(k) \tag{5}$$

when using the basic configuration of the feedforward controller [9]. Fig. 3 shows an additional branch with vectors τ_d^c as input to the feedforward controllers. This is an extension to the feedforward controllers that will be explained in Sect. III-A.

Similar to [7] we compute the desired pose by

$$\mathbf{x}_d(k) = \mathbf{x}_a(k) + \Delta\mathbf{x}(k). \tag{6}$$

The difference with respect to (1) is on the one hand that $\mathbf{x}_a(k)$ is used instead of $\mathbf{x}_c(k-1)$. In fact, (6) does not accumulate sensor data to a commanded pose but computes a desired pose which will be constant if there are no disturbances. Former robot interfaces did not provide the current values of $\mathbf{x}_a(k)$ in each sampling step. This may be a reason why the approach of (1) is common. See Sect. III-B for this case.

On the other hand a gain matrix $\mathbf{K}$ which usually accounts for the closed loop characteristics is not required since (6) represents a geometrical relation which is independent of dynamical or temporal parameters.

Strictly speaking, (6) does not close a loop but simply computes a pose namely the pose $\mathbf{x}_d$. If this pose is reached, i.e. $\mathbf{x}_a(k) = \mathbf{x}_d(k)$ then $\mathbf{s}_a(k) = \mathbf{s}_d(k)$. See Fig. 4 for an example.

To avoid a drift in not measured components a reference trajectory $\mathbf{x}_r(k)$ is introduced that has to be followed. A typical example is

$$\mathbf{x}_r(k) = \mathbf{x}_a(0) \forall k \tag{7}$$

for all components but the feed which is a function of time.

Then, in contrast to (3)

$$\Delta\mathbf{x} = \begin{cases} \begin{pmatrix} \mathbf{R}_s \cdot (\mathbf{s}_d - \mathbf{s}_a) \\ \mathbf{0} \end{pmatrix} & : \text{ measured components} \\ \mathbf{x}_r - \mathbf{x}_a & : \text{ other components} \end{cases} \tag{8}$$

is defined which represents all components. $\mathbf{R}_s$ can be extracted from $\mathbf{x}_a(k)$ since it represents the orientation of the sensor with respect to the base system.

This corresponds to a hybrid position and sensor based control

$$\mathbf{x}_d(k) = \mathbf{S} \cdot \begin{pmatrix} \mathbf{x}_d^m(k) \\ \mathbf{0} \end{pmatrix} + (\mathbf{I} - \mathbf{S}) \cdot \mathbf{x}_d^o(k) \tag{9}$$

with $\mathbf{S}$ as selection matrix and $\mathbf{x}_d^m$ and $\mathbf{x}_d^o$ as the position vector of the measured components and the pose vector of the other components respectively. This yields

$$\mathbf{x}_d^m(k) = \mathbf{x}_a^m(k) + \mathbf{R}_s(k) \cdot (\mathbf{s}_d(k) - \mathbf{s}_a(k)). \tag{10}$$

The actual sensor vector $\mathbf{s}_a(k)$ represents the actual sensor pose $\mathbf{x}_a(k)$ with respect to the pose $\mathbf{x}_o(k)$ of a sensed object (represented by sensible contour points or visible features), as far as the components are sensed.

$$\mathbf{R}_s(k) \cdot \mathbf{s}_a(k) = (\mathbf{x}_a^m(k) - \mathbf{x}_o^m(k)) \tag{11}$$

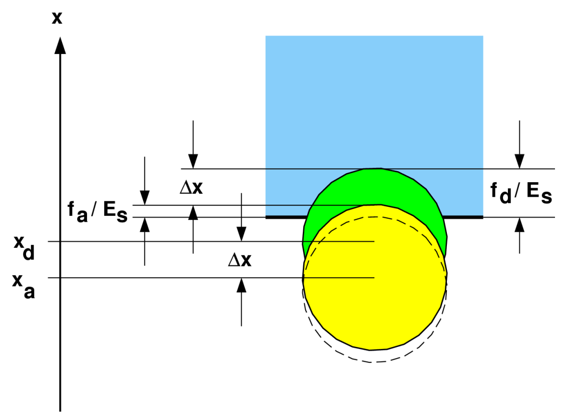


Fig. 4. Computation of the desired tcp position x_d for a pin (yellow) pressing against a fixed block (blue). The desired force f_d will be reached for the green tcp position. The real (dashed) pin position differs by the sensor deflection and remains unchanged.

When inserting (11) in (10), the latter becomes

$$\mathbf{x}_d^m(k) = \mathbf{R}_s(k) \cdot \mathbf{s}_d(k) + \mathbf{x}_o^m(k). \tag{12}$$

This means that (10) does not feed back the actual robot pose. The same is true for the not measured components. Consequently the total system is asymptotically stable as long as the positional controller is stabilizing the system.

Nevertheless the actual robot pose is fed back if the sensor is miscalibrated. If the scale factor is κ instead of 1 and if there is an unknown time difference Δk between measuring of the sensor data and of the robot pose, (12) will become

$$\begin{aligned} \mathbf{x}_d^m(k) = \mathbf{x}_a^m(k) + \kappa \cdot [&\mathbf{R}_s(k) \cdot \mathbf{s}_d(k) \\ &- \mathbf{x}_a^m(k - \Delta k) + \mathbf{x}_o^m(k - \Delta k)] \quad . \end{aligned} \tag{13}$$

Therefore [12] presents a method to adapt the parameters κ and Δk of the sensor data processing. A known time shift between sensor data and the measurements of the robot pose is no problem since we can interpolate between the robot poses of two consecutive time-steps.

A formal stability criterion would be

$$|1 - \hat{G}(z)^{-1} \cdot G(z)| < 1 \tag{14}$$

with $G(s)$ being the transfer function resulting from (13) and the robot dynamics. This means that stability is touched if $\kappa > 2$ or the phase shift caused by Δk is greater than 180^o. The latter means that the temporal uncertainty is in the order of magnitude of the robot time constant. Either condition is only fulfilled with a very poor knowledge of the system.

III. Features of the position controlled robot

Actually, the method of Sect. II works with almost any kind of positional loop of the robot. There are however some features which affect the performance and should therefore be considered.

A. *Positional control including disturbance compensation*

As controller setup we selected the standard feedback controller of the robot manufacturer and an additional feedforward controller (see Fig. 3) since then - provided that the predictions of the desired pose are available (see Sect. IV) - the control error can be reduced to zero. This is true if there are no external forces acting on the robot. So in the strict sense, for force control tasks the positional controller is not optimal. Ref. [13] shows experimental results on the effect of external forces on the positional control accuracy.

To prevent such degradation we extend the feedforward controller. In addition to the current and future desired positional values we process the current and future forces in the feedforward controller. The resulting setup is

$$\begin{aligned}\mathbf{q}_c(k) = \mathbf{q}_d(k) \quad &+ \quad \sum_{i=0}^{n_d} \mathbf{K}_{qi} \cdot (\mathbf{q}_d(k+i) - \mathbf{q}_d(k)) \\ &+ \quad \sum_{i=0}^{n_d} \mathbf{K}_{\tau i} \cdot (\tau_d^c(k+i) - \tau_d^c(k))\end{aligned} \tag{15}$$

where τ_d^c is the part of the desired joint torque that is required to exert the external contact force $\mathbf{f}_d$ at the tcp. The feedforward controller parameters are adapted and represented by the matrices $\mathbf{K}_q$ and $\mathbf{K}_\tau$ which usually are diagonal.

To clarify, the use of force data in the positional control loop is no force control but a compensation of the effect of forces on the positional control.

B. *Acquisition of the actual pose*

Equation (6) assumes that the actual pose $\mathbf{x}_a(k)$ is known in each sampling step k. There are two possibilities to approximate $\mathbf{x}_a(k)$ if it is not provided by the industrial robot interface:

The more advanced approach is to use an observer for $\mathbf{x}_a(k)$. This observer has to model the robot dynamics.

The simpler method is just to use $\mathbf{x}_c(k-k_\Delta)$ instead of $\mathbf{x}_a(k)$. In this case k_Δ is the number of sampling steps which corresponds to the robot time constant.

$$\mathbf{x}_d(k) = \mathbf{x}_c(k-k_\Delta) + \Delta\mathbf{x}(k) \tag{16}$$

With a well tuned robot feedback controller this approach works quite well. Compared with the approach of (1) the use of $k_\Delta > 1$ allows to increase the performance. Nevertheless, if the robot time constant is widely unknown there may be a gain matrix $\mathbf{K} < \mathbf{I}$ advantageous as in (1).

IV. Prediction of future values of the desired pose

Equation (6) recommends to compute the desired pose as the sum of the actual pose and the sensed deviation. Future values of the latter have been computed by extrapolation of order zero. Instead, similar to (12),

$$\mathbf{x}_d^m(k+i) = \mathbf{R}_s(k+i) \cdot \mathbf{s}_d(k+i) + \hat{\mathbf{x}}_o^m(k+i) \tag{17}$$

could be used to compute $\mathbf{x}_d(k+i)$ when the sensor data of time-step k are received.

For $i = 0$

$$\hat{\mathbf{x}}_o^m(k) = \mathbf{x}_o^m(k) = \mathbf{x}_a^m(k) - \mathbf{R}_s(k) \cdot \mathbf{s}_a(k). \tag{18}$$

In contrast, for $i > 0$ there are several possibilities to predict $\hat{\mathbf{x}}_o^m(k+i)$, depending on the knowledge about the process.

A. *Implicit prediction*

If no a priori knowledge is available a general extrapolation is defined by

$$\hat{\mathbf{x}}_o^m(k+i) = \mathbf{x}_o^m(k) + \sum_{j=1}^{n_p} \mathbf{K}_{eij} \cdot (\mathbf{x}_o^m(k) - \mathbf{x}_o^m(k-j)) \tag{19}$$

with usually diagonal matrices $\mathbf{K}_{eij}$ as parameters that can be adapted. This corresponds to a smooth contour that has to be followed, where the extrapolation from previously sensed values predicts the future ones. For unpredictable contours the parameters $\mathbf{K}_{eij}$ will be small. In contrast for linear contours $n_p = 1$ will be sufficient with $\mathbf{K}_{ei1} = i \cdot \mathbf{I}$.

In the general case the parameters $\mathbf{K}_{eij}$ will be adapted either before the experiment at a test path or on-line during the experiment. They are computed to fit (18) with $\hat{\mathbf{x}}_o^m(k+i) = \mathbf{x}_o^m(k+i)$ which is available shortly after time-step $(k+i)$.

Equations (17), (18), and (19) correspond to (1) for a contour following task using a force sensor.

B. *Use of a priori knowledge for explicit prediction*

There are tasks however, which allow a more advanced control since the nature of the disturbances is known. This may be a known equation of motion as a parabola for free flying objects under gravity (see e.g. [14] for such a task), a conveyor belt with known speed and rotation as in [15] or oscillations with measurable cycle time as in surgical applications as in [16]. For all such tasks our method is applicable with (19) replaced by the a priori known setup. The unknown parameters as offset or phase will be adapted on-line.

In the simpler case of tracking a known contour with uncertainty in the measured positional components, the reference path will contain the required data.

$$\hat{\mathbf{x}}_o^m(k+i) = \mathbf{x}_r^m(k+i) - \mathbf{x}_r^m(k) + \mathbf{x}_a^m(k) - \mathbf{R}_s \cdot \mathbf{s}_a(k) \tag{20}$$

C. *Direct measurements of future values*

Depending on the sensor it might be possible to measure the “future” sensor values. This is true for vision systems which are used to track unmoved objects. Each image will allow to measure $\mathbf{x}_o^m$ not only at the current tcp but in a region which corresponds to several sampling steps [11].

If no cameras are available, additional copies of the sensor will be required. These copies are not mounted at the tcp but in front of it, i.e. in the direction of motion. They are then called leading or predictive sensors [17], [18]. This allows to measure at different positions. With a known speed this corresponds to different time steps. The required values of the time-steps $(k+i)$ can then be computed more precisely since interpolation instead of extrapolation is used.

Unfortunately such predictions are sensitive with respect to parameters as the pose of the sensor. Ref. [12] gives an example of the adaptation of these parameters.

D. Smoothing of the desired trajectory

So far there is no filtering of sensor data designed. It is fundamental, however, if there is significant noise on the data. This is assumed here to be suppressed within the sensor.

Filtering is further required if the difference between actual and desired poses is so big that without smoothing, saturation effects of the actuators are interfering. This means that the approach is not suited to record a step response. Instead, at the beginning of a trajectory or after a step of the desired value a smooth reference $\mathbf{s}_d(k)$ has to be defined to avoid saturation of the actuators. In contrast, a general reduction of $\Delta\mathbf{x}$ would limit the performance of trajectory following tasks.

V. Possible applications

So far force control or visual servoing tasks have been discussed. Now we want to emphasize that other tasks are possible as well as regards

1) hybrid position and force control (or hybrid position and sensor based control)
2) impedance control
3) combination of force and vision-based control (or, in general, control with different types of sensors)
4) asynchronous control and sensing (or sensors with large computation time)
5) switching between sensors
6) tracking of a time variant desired sensor vector $\mathbf{s}_d(k)$

Such tasks are possible because of the modularity of our approach. So e.g. for a control task using a force sensor an equation of the desired impedance as

$$\mathbf{E}_d \cdot (\mathbf{x}_d^m - \mathbf{x}_r^m) + \begin{pmatrix} \mathbf{R}_s & \mathbf{0} \end{pmatrix} \cdot (\mathbf{f}_d - \mathbf{f}_r) = 0 \qquad (21)$$

can be integrated to the computation of the desired pose. $\mathbf{E}_d$ is the desired 3x3 elasticity matrix. Then it is not the positional reference $\mathbf{x}_r$ which is sent to the robot controller but the desired pose $\mathbf{x}_d$. $\mathbf{E}_d \neq 0$ means that instead of the reference force $\mathbf{f}_r$ a desired force $\mathbf{f}_d$ wil be reached. This will be outlined in a further paper.

Accordingly, with multiple sensors of different accuracy a weighted desired pose can be computed and tracked by the robot. The weighting may consider different scales and accuracies but has not to take the robot dynamics into account. This is a fundamental difference with respect to a fusion of sensor-based control laws like (1).

It is evident that control is continued without new sensor data using the current robot state and an extrapolation of the previously measured state of the environment. The latter will be updated when new sensor data arrive. So continuously at robot control rate measuring sensors are not required.

Switching between sensors means asynchronous measurements of different sensors. As well in this case the robot dynamics are continuously processed. There is no need to stop before switching, but smoothing may be advantageous, particularly if the sensors record different degrees of freedom (dof).

VI. Experiments

Currently the setup of [6] (Fig. 1) is not more available. Since there the basic setup of our approach has already been demonstrated using a force sensor, we are now free to take another sensor to present the more advanced features. Actually we take image data to control the robot along a cable (Fig. 2). Ref. [11] has documented the high performance when measuring future sensor values according to Sect. IV-C. In contrast, here we restrict to the measurements of the cable position in the center row of the image. So our experiment corresponds to the more difficult case of contour following using a force sensor, thus allowing only implicit prediction of the contour.

The use of vision instead of tactile sensor data allows to restrict to the first line of the positional control law (15). The parameters $\mathbf{K}_{qi}$ have been estimated according to [9].

We use a KUKA KR6 robot with the industrial controller KRC1. So we have a sampling time of 12 ms and a time constant of about 80 ms of the robot and the controller, including a built-in filter. The camera is a standard monochrome device which provides image fields every 20 ms. The mean delay between the image capture and the control sampling step in which it is used is 30 ms.

Sensor data are only used to control the radial component of our planar path. The remaining 5 dof are given by a reference path which turns the tcp back and forth at 0.2 m/s around the robot base. This corresponds to a hybrid position and force control problem with a tool oriented task frame.

The right hand side of Fig. 5 compares our approach with a manually tuned PD controller according to (1) with $K_p = 0.3$ and $K_d = 0.2$. This controller is slightly superior to our basic approach. The reason might be that the tuning implicitly considers the steep but smooth layout of the cable (see left hand side of Fig. 5). But our approach is by far superior when using an extrapolation according to (19) with off-line estimated parameters for $n_p = 10$.

This performance is only slightly affected if the actual tcp position is not provided by the robot manufacturer. Fig. 6 shows the performance with (16) (with $k_\Delta = 7$) instead of (6).

Fig. 6 further shows the result when the estimated sensor gain is totally wrong. This gain is represented by the distance from the camera to the cable in our case, or by the elasticity in case of a force control scenario. Both parameters can be tuned easily. In contrast, for the last experiment we used twice the real value, which corresponds to the stability limit. Therefore the red curve in Fig. 6 displays a small-scale oscillation but, after all, an acceptable result which by chance is slightly superior to the PD controller, because the higher gain improves tracking of

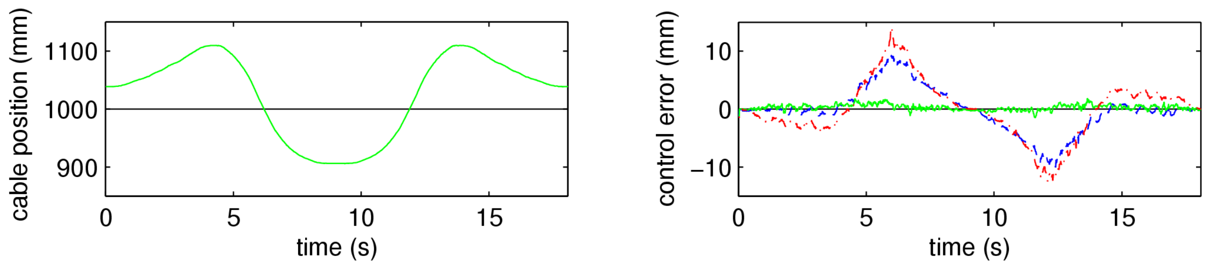


Fig. 5. Radial component of the cable position and of the control error using a PD-controller according to (1) (dashed blue line), our approach (6) (dash-dotted red line), and our approach with extrapolation according to (19) (solid green line)

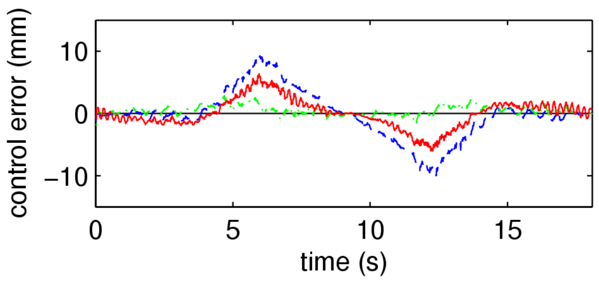


Fig. 6. Control error with PD-controller according to (1) (dashed blue line), our approach with extrapolation but with (16) instead of using the positional measurements (dash-dotted green line), and our approach without extrapolation, using a wrong gain according to the stability limit $\kappa = 2$ (solid red line)

the non modeled cable layout. Of course, the performance with modeled cable layout is not reached.

Since a better performance generally requires bigger actuator values, the presented method may show more saturation effects than simpler methods, at least when big delays are present. This can be avoided by a scaling, either in the extrapolation (19) or in the feedforward equation (15), to reduce the accelerations of the robot commands $\mathbf{q}_c$. Unfortunately this limits the performance and therefore it has not been used in the experiments.

VII. Conclusion

We demonstrated that the presented approach in its basic form and without tuning is equivalent to the well known control setup of (1) with appropriate parameters. When our method is extended using an assumption on the type of disturbance, the reached accuracy outperforms the usual method by far, even in the general case without a priori information on the disturbance.

The improvements are not restricted by stability limits but only by saturation effects of the actuators. Besides, the system is very robust with respect to miscalibrated sensors.

In contrast to other inner loop / outer loop approaches we separately design the positional controller and the modeling of sensor data. So besides the strong performance our approach is well suited for a modular system, which has been required with the asynchronously measuring sensor.

References

[1] L. Sciavicco and B. Siciliano. *Modeling and Control of Robot Manipulators*. Electrical and Computer Engineering Series. McGraw-Hill, 1996.

[2] J. Roy and L. L. Whitcomb. Adaptive force control of position/velocity controlled robots: Theory and experiment. *IEEE Trans. on Robotics and Automation*, 18(2):121–137, April 2002.

[3] 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.

[4] F. Pertin and J.-M. Bonnet des Tuves. Real time robot controller abstraction layer. In *Proc. Int. Symposium on Robots (ISR)*, Paris, France, March 2004.

[5] J. Grewe, G. Hirzinger, R. R. Koeppe, C. Strobl, and B. Willberg. Compliant-force-torque-sensor. In *Proc. 10th Int. Workshop on Robotics in Alpe-Adria-Danube Region*, Vienna, Austria, May 2001.

[6] 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.

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

[8] 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, April 2000.

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

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

[11] F. Lange and G. Hirzinger. Spatial control of high speed robot arms using a tilted camera. In *Int. Symposium on Robotics (ISR 2004)*, Paris, France, March 2004.

[12] F. Lange and G. Hirzinger. Calibration and synchronization of a robot-mounted camera for fast sensor-based robot motion. In *Proc. 2005 IEEE Int. Conf. on Robotics and Automation (ICRA)*, Barcelona, Spain, April 2005.

[13] F. Lange. *Adaptiv vorausplanende Steuerung für schnelle sensorbasierte Roboterbewegungen*. PhD-Thesis, University of Karlsruhe, 2003. http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document =2003/informatik/1, in German.

[14] U. Frese et al. Off-the-shelf vision for a robotic ball catcher. In *Proc. IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, pages 1623–1629, Maui, Hawaii, Oct/Nov 2001.

[15] S. Jörg, J. Langwald, J. Stelter, G. Hirzinger, and C. Natale. Flexible robot-assembly using a multi-sensory approach. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 3687–3694, San Francisco, April 2000.

[16] R. Ginhoux et al. Beating heart tracking in robotic surgery using 500 hz visual servoing, model predictive control and an adaptive observer. In *Proc. 2004 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 274–279, New Orleans, LA, April 2004.

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

[18] J. Baeten and J. De Schutter. Combined vision / force control at corners in planar robotic contour following. In *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, Como, Italy, July 2001.