Proc. Int. Conf. on Robotics and Automation (ICRA) Anchorage, AK, May 2010

Classification and Prediction for Accurate Sensor-Based Assembly to Moving Objects

Friedrich Lange, Johannes Scharrer, and Gerd Hirzinger

***Abstract*— Typical industrial assembly tasks require an accuracy that cannot be realized by only feedback control if a minimum speed is given by a conveyor. Feed-forward has proven to be advantageous, using predictions of the desired trajectory which will be computed from sensor values. These predictions are improved by a model based classification of the sensor data to typical scenarios. In contrast to linear controllers this assures the fastest possible response to external disturbances, in spite of large dynamical delays. The method is demonstrated by assembling wheels to a car body that is moved by a conveyor, fusing sensor data using an extended Kalman filter.**

I. Introduction

This paper is related to industrial assembly of parts to objects that are moved by a conveyor. A typical application is the assembly of wheels to a car body that is being moved by a power-and-free conveyor (see Fig. 1). In this set-up each car body is transported by a conveyor unit that autonomously moves within the conveyor.

For such tasks the desired robot trajectory is online computed from sensors that track the conveyed object. Strictly speaking, the area in which the assembly is intended, has to be tracked, not the suspension of the moving object within the conveyor [1]. Instead, [2] off-line senses the assembly pose with respect to the conveyor unit and then uses a conveyor-based sensor to propagate the pose of the designated assembly point. This approach works well with a smooth conveyor motion, but in case of accelerations of the conveyor unit the accuracy is not sufficient. Then the conveyed object may swing around its suspension, which cannot be perceived by the position of the conveyor unit. Such oscillations may be tracked by [3] but there the object may be occluded when approaching to it.

The usual alternative besides manual assembly is to stop the conveyor unit and to fix the object by clamps or fixing pins. Then the assembly may be executed in a traditional way, i.e. by an optimized program without online modifications. This is called assembly at clocked cycles. This is applied, e.g. for the insertion of the windshield or a retractable roof. The disadvantage is the amount of time and space that is required.

Therefore in [4] a set-up has been presented that provides the sensors that are required to ensure an accurate assembly in motion and really track the designated assembly point of the conveyed object. The bottleneck is control, since an accuracy of about 1 mm is desired in presence of disturbances or uncertainties of some tens or hundreds of millimeters and changes of about a millimeter per sampling interval.

Fig. 1. Set-up at *iwb* for mounting wheels to a moving car body.

Since with feedback control there is always a delay or a bias when disturbances vary, this paper primarily uses a feed-forward approach where a feed-forward controller is understood as a filter that weights future desired poses. Thus the applicability depends on the availability of reliable predictions of the future desired trajectory. These are improved by sensor fusion for the detection of the current pose and by classification methods which select one of several model-based scenarios for prediction.

This paper is organized as follows. Next, the overall structure of control is outlined (Sect. II), followed by the two main parts: the sensor-based determination of the desired robot trajectory (Sect. III) and the predictive approach for control (Sect. IV). Thereby, Sect. III includes the sensor fusion and the classification for prediction. Finally, the performance is demonstrated by real experiments with the wheel assembly of Fig. 1 (Sect. V).

II. General Approach

We distinguish between the position-based control of the robot and the determination of the desired trajectory where the latter is based on sensory data [5]. Similar concepts are known as inner loop - outer loop methods. The inner loop or

F. Lange and G. Hirzinger are with the German Aerospace Center (DLR), Institute of Robotics and Mechatronics, D-82234 Wessling, Germany friedrich.lange@dlr.de

J. Scharrer is with the Technical University of Munich, Institute for Machine Tools and Industrial Management (*iwb*), Application Center Augsburg, D-86153 Augsburg, Germany

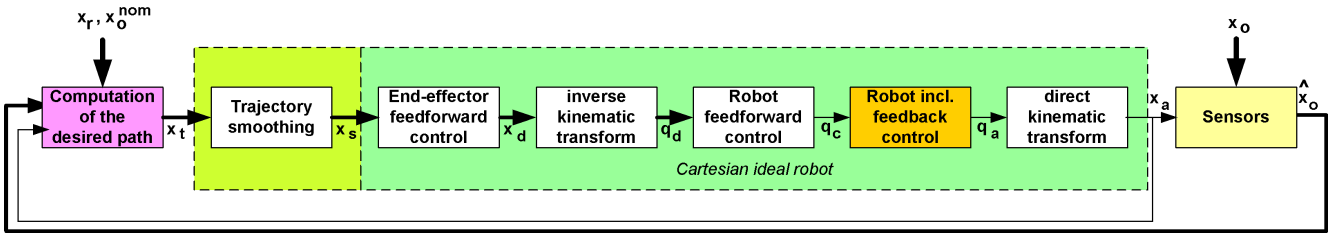

Fig. 2. Signal flow for predictive trajectory control of the robot.

lower level is represented by position control and the outer loop or upper level is realized by feedback of the sensed control error. Our approach is somewhat different, as we do not use pure feedback but a computation of the desired pose and the prediction of its trajectory.

This gives the opportunity to modularize the system. There is the robot dependent trajectory execution (green region in Fig. 2, see Sect. IV for more details) and the task dependent computation of the desired trajectory (Sect. III). Both modules communicate by a predictive interface, which allows for high accuracy during motion.

The separation facilitates independent optimizations, e.g. to consider end-effector oscillations or joint compliance in the control module. Similarly, the determination of the desired robot motion may be extended to reproduce a potential swinging of the conveyed object, or to model the speed ramp of the conveyor unit in the first seconds after stopping or restarting the conveyor unit. Besides, the modularization is advantageous when the setup is modified, e.g. by removing the linear axis that moves the robot parallel to the conveyor, or when the task is changed.

The input to the module for the computation of the desired motion is the task description in terms of a reference trajectory $\mathbf{x}_r(k)$ and the corresponding nominal poses of the target object $\mathbf{x}_o^{nom}(k)$, both given for all sampling steps k. In addition the current robot pose $\mathbf{x}_a(k)$ and the current sensed values of the object pose $\hat{\mathbf{x}}_o(k)$ are sampled in parallel.

The output to the robot interface are the commanded joint angles $\mathbf{q}_c(k)$. This interface is available for most robots [6]. The interface between the computation of the desired trajectory and the control is implemented as a list of frames $\mathbf{x}_t(k,i)$, which in each sampling step k predict the desired poses of the tool center point (tcp)[1] in time-steps $k+i$ with $i = 0, \cdots n$. In Sect. IV these predictions are used to improve tracking. Their generation is a challenge for the computation of the desired trajectory of Sect. III.

III. Sensor-Based Computation of the Desired Trajectory

Since a reference trajectory with a given velocity profile is present, there is no trajectory planning required but only a spatial modification of an existing trajectory.

As proposed in [1], we use three different types of sensors. The contact phase can only be controlled by a tactile sensor, in our case a compliant force-torque sensor with 6 degrees of freedom (dof) which is mounted at the suspension of the end-effector to the robot. Before the contact phase, the designated assembly point of the conveyed object is tracked by a CCD camera that is mounted on the robot, within the assembly tool. Fig. 3 shows an image from this camera. The third sensor is a distance sensor that gives the 1 dof position of the conveyor unit with respect to the robot base. It is required initially to activate the vision system and later, when no other sensor data are available, to detect irregularities in the conveyor motion, i.e. the motion of the suspension of the conveyed object. The latter is important since there is still a risk of collision when the robot retreats from the car after completion. In addition, tracking of the position of the conveyor unit can help to distinguish such irregularities from oscillations of the conveyed object with respect to its suspension. This will be discussed in a further paper.

Except for this distinction, the sensors are used sequentially. At least tactile and non-contact sensors are mutually exclusive. Nevertheless, it is emphasized to apply a fusion method in order to inhibit biases when the sensor is changed.

A. Sensor Fusion

Sensor fusion is implemented by a Kalman filter that weights all sensor data with their corresponding accuracies. In this way a Kalman filter is used for implicit data driven sensor selection. The speed of the target and thus the basis for the predictions that are needed in Sect. IV are obtained without extra effort. In addition, a Kalman filter can process more detailed model information, e.g. a prediction of the

Fig. 3. Typical image of the robot mounted camera when measuring the pose of the wheel hub from greater distance.

[1]The tcp is understood not only as a point but as a part of the tool, represented by position and orientation.

swinging of the conveyed object.

In the basic version of our method the Kalman state $\mathbf{x}(k)$ is a vector of 7 components. The first 6 components are defined by the fused sensed desired Cartesian pose of the tcp ${}^r\mathbf{x}_f(k)$, while the 7th component represents the target speed in the conveyed direction. All variables are expressed with respect to the reference system (i.e. the frame aligned to the current reference pose of the tcp) and in the reference system. Thus the measurement equation for a single component of one of the sensors is

$$x'_{oi} = \mathbf{h}_i^T \mathbf{x} \tag{1}$$

with $\mathbf{h}_i$ being a vector that selects the appropriate component of the state vector[2], and x'_{oi} the sensed component of the sensor correction[3].

The Kalman filter correction equations when processing the component i of a sensor vector $\mathbf{x}'_o(k+1)$ are

$$\begin{aligned}\mathbf{x}(k+1/k+1) = \mathbf{x}(k+1/k) + \mathbf{K}(k+1)\\ (x'_{oi}(k+1) - \mathbf{h}_i^T(k+1)\ \mathbf{x}(k+1/k))\end{aligned} \tag{2}$$

with Kalman gain[4]

$$\begin{aligned}\mathbf{K}(k+1) = \mathbf{P}(k+1/k)\mathbf{h}_i(k+1)\\ (\mathbf{h}_i^T(k+1)\mathbf{P}(k+1/k)\mathbf{h}_i(k+1) + \sigma_i(k+1))^{-1}\end{aligned} \tag{3}$$

and

$$\begin{aligned}\mathbf{P}(k+1/k+1) = \mathbf{P}(k+1/k) -\\ \mathbf{K}(k+1)\mathbf{h}_i^T(k+1)\ \mathbf{P}(k+1/k),\end{aligned} \tag{4}$$

where $\mathbf{P}(k+1/k)$ denotes the covariance matrix of the estimation errors at step $k+1$, given the measurements up to step k. $\sigma_i(k)$ is the current variance of the noise of the sensed component i. Like this, all sensors are weighted according to their assumed accuracy. If a sensor is not available, e.g. a force sensor without contact, $\sigma_i(k) = \infty$.

The estimation of ${}^r\mathbf{x}_f$ at future time-steps is considered in the Kalman prediction equations

$$\mathbf{x}(k+1/k) = \mathbf{A}(k)\ \mathbf{x}(k/k) + \mathbf{b}(k) \tag{5}$$

and

$$\mathbf{P}(k+1/k) = \mathbf{A}(k)\ \mathbf{P}(k/k)\mathbf{A}^T(k) + \mathbf{Q}(k), \tag{6}$$

where $\mathbf{Q}(k)$ is the covariance matrix of the non-modeled changes of $\mathbf{x}(k+1/k+1)$ with respect to $\mathbf{x}(k+1/k)$.

$\mathbf{A}$ and $\mathbf{b}$ represent the modelled state equation for $\mathbf{x}$. If no model of the motion is available, we take

$$\mathbf{A} = \mathbf{I}_{7\times 7} + \begin{pmatrix} & & 0 \\ & \mathbf{0}_{6\times 7} & 1 \\ & & \mathbf{0}_{1\times 5} \end{pmatrix} \text{ and } \mathbf{b} = \mathbf{0}_{1\times 7} \tag{7}$$

assuming y as the conveying direction. Otherwise $\mathbf{A}$ and $\mathbf{b}$ can contain a model for the pendulum motion around the direct axis of the car[5], with an extended state vector.

Alternatively, the non-diagonal components of the measurement equations (1) may be changed to

$$\begin{aligned}x'_{o1}(k) &= x_1(k) + a_1\cos(\omega(k-\tau(k))T_0)x_8(k)\\ &\quad + a_1\sin(\omega(k-\tau(k))T_0)x_9(k)\\ x'_{o2}(k) &= x_2(k) - \tau(k)x_7(k)\\ x'_{o3}(k) &= x_3(k) + a_3\cos(\omega(k-\tau(k))T_0)x_8(k)\\ &\quad + a_3\sin(\omega(k-\tau(k))T_0)x_9(k)\\ x'_{o5}(k) &= x_5(k) + \cos(\omega(k-\tau(k))T_0)x_8(k)\\ &\quad + \sin(\omega(k-\tau(k))T_0)x_9(k),\end{aligned} \tag{8}$$

where the additional states x_8 and x_9 implicitly represent the amplitude and the phase of the oscillation. The lengths a_1 and a_3, and the frequency ω are previously identified constant parameters, and $\tau(k)$ is the individually measured time delay of the vision system, or zero when (8) is used to include the tactile sensor.

When using the approach of (8) the prediction of the Kalman states is with (7) but the desired poses are computed from

$$\begin{aligned}{}^r x_{f1}(k+i) &= x_1(k) + a_1\cos(\omega(k+i)T_0)x_8(k)\\ &\quad + a_1\sin(\omega(k+i)T_0)x_9(k)\\ {}^r x_{f2}(k+i) &= x_2(k) + ix_7(k)\\ {}^r x_{f3}(k+i) &= x_3(k) + a_3\cos(\omega(k+i)T_0)x_8(k)\\ &\quad + a_3\sin(\omega(k+i)T_0)x_9(k)\\ {}^r x_{f5}(k+i) &= x_5(k) + \cos(\omega(k+i)T_0)x_8(k)\\ &\quad + \sin(\omega(k+i)T_0)x_9(k).\end{aligned} \tag{9}$$

A further state may be introduced to represent the offset between the distance sensor at the conveyor and the other sensors that really survey the designated contact point. This offset should be known a priori. But because of calibration errors or an uncertain position of the car with respect to the conveyor fixture it is proposed to estimate its value $x_{10}(k) = x_{10}$ too. This yields

$$x'_{o2(dist.sens.)}(k) = x_2(k) + x_{10}(k) \tag{10}$$

B. Classification to Typical Scenarios

Sect. III-A gives an optimal estimate of the current pose of the assembly point at the conveyed object. For control purposes it is required however (see Sect. IV), to compute not only a pose but a trajectory that covers some tenth of a second, beginning with the time instant of the estimation. For steady state conveyor motion this will be done by predictions (9) assuming that the current speed $x_7(k) = x_7$ will be constant and that the oscillation parameters x_8 and x_9 do so as well.

[2]When using the camera for sensing in the conveying direction, a nonzero value $h_{i7} = -\tau$ is required additionally to account for the delay since the time instant of the exposure.

[3]Since all variables are expressed with respect to the reference system, $\mathbf{x}'_o = {}^r\mathbf{x}_o - {}^r\mathbf{x}_o^{nom}$ where ${}^r\mathbf{x}_o$ is computed from the sensed object pose $\hat{\mathbf{x}}_o$. (Strictly speaking, a product of homogeneous transformation matrices is used instead of the sum of vectors, since the orientational deviations may be substantial.)

[4]In contrast to the usual notation, $\mathbf{K}$ is not a matrix but a vector.

[5]For a suspension of the conveyed object as in Fig. 1 all swinging besides oscillations around the direct axis may be neglected.

In contrast, in reality it happens that a conveyor unit is stopped in order to prevent a collision with a preceding car. After being stopped, the conveyor unit will restart again when there is free space, maybe already in the deceleration phase. Since the robot control is not connected with the autonomously acting conveyor units, a stop or a restart has to be perceived from sensor data. A classification is advantageous, since otherwise the predictions will be far from the required accuracy.

So there are three classes, i.e. continuous motion with the current speed, conveyor stop with a known deceleration (104 mm/s^2 in our experiments) until speed is zero, and conveyor start with a known acceleration (13 mm/s^2) until speed is nominal (106 mm/s).

The classification takes into account that the sensors have a limited resolution and that there are outliers. The challenge is to detect a changing class as early as possible. This is realized rule based by:

- The selected class switches if a certain prediction error is exceeded when using the old class for prediction. Then the class with the least prediction error is selected as a candidate and applied tentatively.
- The change of class is confirmed when the prediction error further increases when predicting with the old class.
- It will be cancelled if the new class is inferior than the old class. Then an outlier is assumed.

Since the distance sensor on hand has a poor resolution, a conveyor start is detected about 1.5 s after its beginning which can be determined afterwards. It is confirmed 0.3 s later. This is quite fast since the whole acceleration phase lasts about 8 s. A conveyor stop is perceived after 0.16 s and confirmed 0.12 s later, 0.74 s before coming to a stop. The classification is faster if the other sensors are active.

Like this, prediction errors and thus control errors are inevitable. But they are much less than without classification.

C. Computation of a Feasible Trajectory

The computation of the desired trajectory is performed in two steps. Firstly, the sensor correction is computed from the sensed contact point of the conveyed object, as described above. Secondly, the resulting fused sensed trajectory $\mathbf{x}_f$ is revised in order to guarantee a feasible trajectory $\mathbf{x}_t$ for the real motion of the tcp. For example, the robot is stopped in the beginning, when the conveyed object is still not close enough.

After that, Sect. III-A yields tracking of the sensed target at a pre-programmed, time-varying distance in order to approximate to the target at the given speed. In this phase, a nonlinear heuristic scheme defines the transient motion from the sensed desired trajectory $\mathbf{x}_f$ to the trajectory $\mathbf{x}_t$ that the tcp is desired to execute. This avoids the typical step response from the programmed to the sensor controlled motion when first image data are received. A quadratic approach further assures that the sensed desired trajectory is reached smoothly, without exciting oscillations of the robot end-effector. This takes place before the pre-programmed path reaches the nominal target object, which, in the case of accurate control, coincides with the moment of the real impact.

All this computation of the desired trajectory is revised at every sampling step. Therefore the trajectory which is interface to the feed-forward controllers of Sect. IV remains always up-to-date. The computed trajectories are represented by the Cartesian poses at the current and subsequent sampling steps, with a time horizon of some tenths of a second. This enables the robot to follow the desired motion smoothly and in the presence of limited actuation torques.

IV. Predictive Trajectory Control

The trajectory control is realized using model-based predictions of the smoothed desired trajectory $\mathbf{x}_s$ of the tcp, as input to a two-stage feed-forward filter (Fig. 2). Its upper-level part considers the elastic mounting of the end-effector, which is caused by compliance in the force-torque sensor. Such elasticity is required in the contact phase but may cause oscillations when moving in free space. Its damping is outlined in Sect. IV-A. The lower-level part concerns the dynamical control of the robot joints. This will be explained in Sect. IV-B.

This approach allows for very small control errors in spite of the end-effector's and the robot's dynamics as long as the motion predictions for the end-effector trajectory do not change. This means e.g. that the sensed trajectory of the designated assembly point of the conveyed object is correct.

Assuming that the predictions of Sect. III are accurate enough, there is no feedback controller required besides the standard position controller of the robot manufacturer. Instead, control concentrates on feed-forward modules. Thus the system is inherently stable [5].

Smoothing of this trajectory is required since a controller that guarantees small control errors is very sensitive to the noise of a sensed trajectory. In Sect. III-C, discontinuities of the desired path have been prevented, but discontinuities of the desired speed still occur. Their smoothing is shown in Sect. IV-C.

For dynamical reasons the control concentrates on the 6 axes of the robot arm. The speed of the redundant linear axis is not manipulated as long as the conveyor unit does not stop. In that case the robot completes the assembly while its base is stopped, if needed. This is demonstrated in Sect. V.

A. Damping of the End-Effector Oscillations

Oscillations caused by compliance at the suspension of the end-effector are best compensated in Cartesian space since such deflections are independent of the robot joints (see Fig. 2).

Oscillation damping is designed as an input shaping filter [7], which means feed-forward with the smoothed desired tcp trajectory $\mathbf{x}_s$ as input. This yields a desired trajectory $\mathbf{x}_d$ of a virtual robot with a rigidly modelled end-effector.

$$\mathbf{x}_d(k+i) = \mathbf{x}_s(k+i) + \sum_{j=-n_l}^{n_u} \mathbf{R}_j^{osc}(\mathbf{x}_s(k+i+j) - \mathbf{x}_s(k+i)). \tag{11}$$

$\mathbf{R}_j^{osc}$ are matrices of the controller parameters that are computed off-line from the measured deflections of the force-torque sensor. See [7] for details on the controller design.

Feedback of the actual deflections is possible but not used since its benefit is highly restricted by the limited actuator torques.

B. Joint-Based Feed-Forward Control

Since robot controllers that are supplied by a robot manufacturer typically control positions instead of trajectories, we use an adaptive feed-forward controller that supplements the standard feedback controller. Both controllers are implemented on joint level, since in this level there are less couplings between the individual dofs.

$$\mathbf{q}_c(k) = \mathbf{q}_d(k) + \sum_{i=1}^{n_{rob}} \mathbf{R}_i^{rob}(\mathbf{q}_d(k+i) - \mathbf{q}_d(k)), \quad (12)$$

where $\mathbf{q}_c$ is the vector of commanded joint values that are input to the standard control system of the robot manufacturer. $\mathbf{R}_i^{rob}$ are diagonal matrices with the parameters, each matrix operating with the individual joint motions in a certain prediction period of i time steps. Details on the structure and the adaptation of the $\mathbf{R}_i^{rob}$ can be found in [8]. A similar approach is reported in [9].

C. Trajectory Smoothing

Smoothing of the desired trajectory may be assigned either to the control part or to the computation of the desired trajectory. It is designed similar to impedance control by defining the characteristics of the desired trajectory by a mass $\mathbf{M}_d$, a damper $\mathbf{D}_d$, and a spring $\mathbf{E}_d$, where $\mathbf{E}_d = \mathbf{0}$ inhibits static deviations. This impedance law is desired with respect to the reference, in order to smooth the deviations from the reference path and not the programmed motion itself.

As explained in more detail in [10], the system of equations

$$(\mathbf{I} + \mathbf{E}_d)^r \mathbf{x}_s(k+i) = {}^r\mathbf{x}_t(k+i) \quad (13)$$

$$\mathbf{D}_d \frac{{}^r\mathbf{x}_s(k+i+1) - {}^r\mathbf{x}_s(k+i-1)}{2T_0} = 0 \quad (14)$$

$$\mathbf{M}_d \frac{{}^r\mathbf{x}_s(k+i+1) - 2{}^r\mathbf{x}_s(k+i) + {}^r\mathbf{x}_s(k+i-1)}{T_0^2} = 0 \quad (15)$$

is optimized beginning from the current time-step k with $i = 0$, until a prediction horizon $i = n_{imp}$ that is chosen large in order not to restrict the smoothing. Since (13) - (15) are linear, the optimization can be executed off-line, yielding a feed-forward filter as (11) or (12).

$$\mathbf{x}_s(k+i) = \mathbf{x}_t(k+i) + \sum_{j=1}^{n_{imp}} \mathbf{R}_j^{imp}(\mathbf{x}_t(k+i+j) - \mathbf{x}_t(k+i)) \quad (16)$$

The matrices $\mathbf{R}_j^{imp}$ represent the smoothing parameters.

Altogether a prediction horizon of $n = n_{rob} + n_u + n_{imp}$ time steps has to be provided by the computation of the desired trajectory. This is about 0.5 s for our system, where small changes in future predictions have only a little effect.

V. Experiments

A typical experiment with the setup of Fig. 1 is demonstrated in Fig. 4. The plot displays the fused estimate for the pose $\mathbf{x}_f$ of the wheel hub[6] and the robot trajectory $\mathbf{x}_t$ that is computed from this. The left hand diagram represents the position while the right hand side concerns the orientation which is about 23 degrees off from the nominal value.

The plot begins when the car reaches the start position. Then, within about a second, the pose and the orientation of the wheel hub is perceived and a smooth trajectory is computed that reaches the hub pose at time 6 s. In the conveyed direction (y) a constant speed of the car is detected (see black dash-dotted line), until at time 5 s the conveyor unit is stopped. After the deceleration phase the conveyor unit stands still. At time 9 s the conveyor unit continues its motion. At time 16 s the old speed is reached. In parallel, the robot approaches (z direction) to the wheel hub, and a wheel is mounted and fixed. At time 9.5 s the screws are fixed, which can be seen by a small orientational step. The linear axis is stopped from time 6 s to 10 s. At time 15 s the robot retreats from the conveyor.

In spite of the accelerations the control errors are less than 1 mm (2 mrad) during the approaching phase and during contact (see Fig. 5). They become larger when the robot retreats from the conveyor. Note the different scale with respect to the computed motion.

The attached video clip displays the robot motion and the images of the used camera. There are small prediction errors visible since the conveyor unit stops just before contact is reached. Nevertheless the assembly is finished robustly. Similar video clips can be found at our web site www.robotic.de/212/.

Without prediction, the robot follows the wheel hub with delay, such that assembly fails, at least if the conveyor speed differs from the nominal speed. Without classification, a conveyor stop is not allowed since it would damage the setup, unless contact is released by an emergency retraction.

VI. Conclusion and Future Work

The paper explains the computation of a feasible robot trajectory and its control such that all accuracy requirements are kept. The computation of the desired robot pose is done by fusing the estimated poses of the designated assembly point from different kinds of sensors, taking into account that the conveyed object may swing around its suspension. Then a desired robot trajectory is predicted, using a classification of the fused poses to select one of three possible scenarios for the conveyor motion. These predictions give the opportunity for a very precise control of the robot pose, such that the desired robot trajectory is executed properly. The control comprises an oscillation damping of the elastically mounted end-effector and a feed-forward control of the robot joints. Like this, a dynamic model of the conveyed setup and a proper calibration of the camera [11] can prevent a hard

[6]Strictly speaking, $\mathbf{x}_f$ includes the desired distance between the robot and the wheel hub, such that both curves correspond.

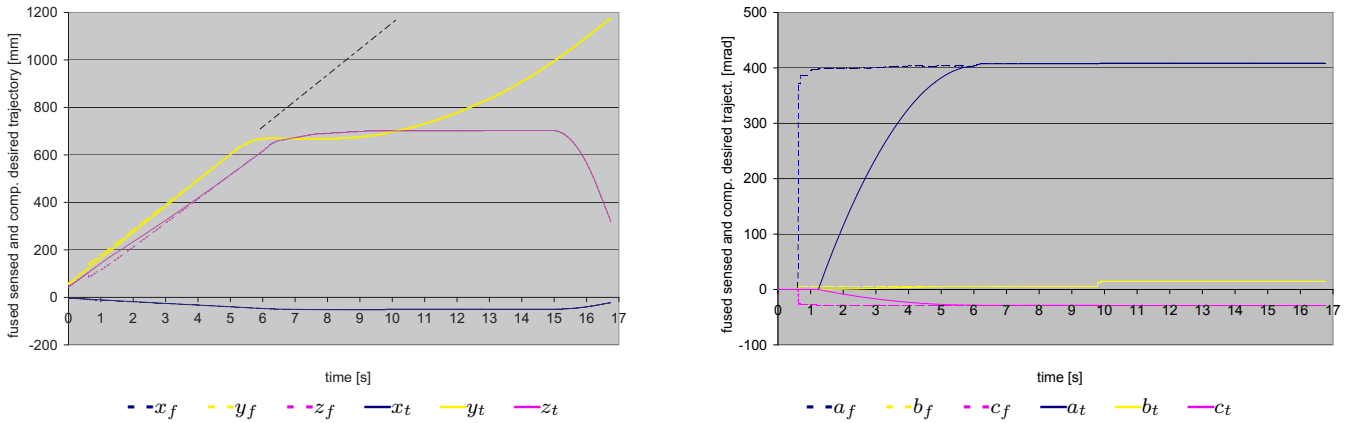

Fig. 4. Fused sensed trajectory $\mathbf{x}_f$ (dashed) and computed desired robot trajectory $\mathbf{x}_t$ (solid).

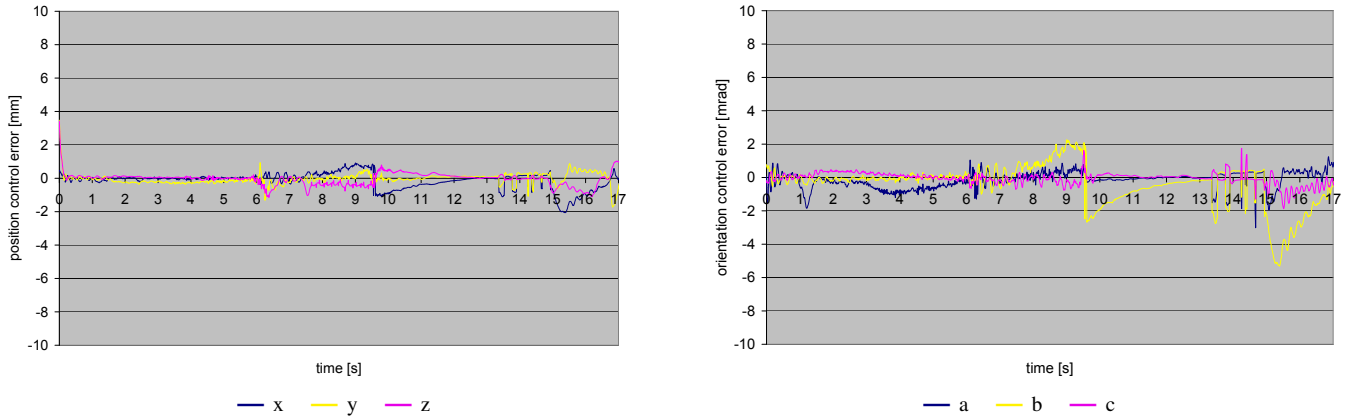

Fig. 5. Control error during the wheel assembly.

impact. The remaining prediction errors are within the range of compliance of the sensor.

Future work will, firstly, replace the position sensor at the conveyor, such that a changed scenario can be detected earlier. Secondly, oscillations will be modeled around other than the direct axis. This is designed in order to tolerate disturbances that act on the conveyed object if a second robot or even a human has contact to it as well. In this case the evaluation and prediction of sensor data has to include a distinction between a jerk caused by contact forces and a conveyor stop that accelerates the same dof.

ACKNOWLEDGMENT

This work is funded by KUKA Roboter GmbH. It is based on a previous project that had been supported by the Bayerische Forschungsstiftung.

REFERENCES

[1] G. Reinhart and J. Werner. Flexible automation for the assembly in motion. *Annals of the CIRP*, 56(1):25–28, 2007.

[2] W. Meyer. Mounting wheels automatically on moving car bodies: Increasing cost-effectiveness with on-the-fly assembly. ISRA VISION SYSTEMS, 2006. http://www.isra.de/media/public/pdf2006/97_Pressenotiz_Radmontage_e_final.pdf.

[3] H. Chen, W. Eakins, J. Wang, G. Zhang, and T. Fuhlbrigge. Robotic wheel loading process in automotive manufacturing automation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3814–3819, St. Louis, USA, Oct. 2009.

[4] F. Lange, J. Werner, J. Scharrer, and G. Hirzinger. Assembling wheels to moving car bodies using a standard industrial robot. In *Proc. 2010 IEEE Int. Conf. on Robotics and Automation (ICRA)*, Anchorage, AK, USA, May 2010.

[5] F. Lange and G. Hirzinger. Stability preserving sensor-based control for robots with positional interface. In *Proc. 2005 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1712–1717, Barcelona, Spain, April 2005.

[6] KUKA Roboter GmbH. *Robot Sensor Interface (RSI), Release 2.0*, 2006.

[7] A. Kamel, F. Lange, and G. Hirzinger. New aspects of input shaping control to damp oscillations of a compliant force sensor. In *Proc. 2008 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2629–2635, Pasadena, CA, May 2008.

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

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

[10] F. Lange, M. Frommberger, and G. Hirzinger. Is impedance-based control suitable for trajectory smoothing? In *Preprints 8th IFAC Symposium on Robot Control (SYROCO 2006)*, Bologna, Italy, Sept. 2006.

[11] K. H. Strobl, W. Sepp, S. Fuchs, C. Paredes, and K. Arbter. DLR CalDe and DLR CalLab. Institute of Robotics and Mechatronics, German Aerospace Center (DLR). http://www.robotic.dlr.de/callab/.