A flexible multi-sensor positioning system for industrial robots

PHILIPP ROEBROCK
VMT Vision Machine Technic
Bildverarbeitungssysteme GmbH
Mallaustraße 50-56
68219 Mannheim
GERMANY
philipp.roebrock@vmt-gmbh.com

Abstract: This paper offers an approach for a sensor-based robot positioning system which solves the problem of positioning the tool of an industrial robot relative to a work object for a subsequent assembly or application in the same way as it was done once during setup with a reference work object. The approach takes sensors as abstract data sources and is therefore able to use different kinds of sensors at the same time to fulfill its task. The proposed solution focusses on a flexible system that suits the needs of the industry in respect of good maintainability, good robustness and high accuracy.

Key–Words: Visual Servoing, Robot Positioning System, Multi-Sensor System

1 Introduction

The modern industry is affected by an increasing level of automation to assure higher levels of product quality or just for reasons of economy. With the increasing number of automation applications grow the requirements of accuracy and flexibility of todays industrial robot applications. The components that have the most important influence over the accuracy of the complete process are the manipulator system itself (consisting of the robot, of the tool and probably of external axes), the conveyor system and the amount of production tolerances in the work object. To compensate these tolerances the usage of sensor systems for robot control is inevitable.

Besides sensor-based path correction systems for industrial robots that are capable of controlling the path of the robot along a usually fixed work object (see e.g. [11] or [16]) there is a class of applications that is considered in this paper. They solve the problem of positioning the robot relative to the fixed work object for a subsequent handling or alternatively track the work object with the robot while it is moved with the conveyor system. The user defines the nominal position of the robot relative to a reference object once during system setup. Each production cycle the system tries to position the robot relative to the work object in an identical way compensating the tolerances mentioned above. Fig. 1a shows the schematic of the situation of the system setup with the nominal positioning of the tool relative to the reference work object. In Fig. 1b there is a different work object in a different position. The system uses the four sensors to pose the robot tool relative to the work object in a same way like it was during setup. Remaining sensor residuals because of form deviations in the work object are evenly distributed over all sensors. The result of this behavior is shown in Fig. 1c.

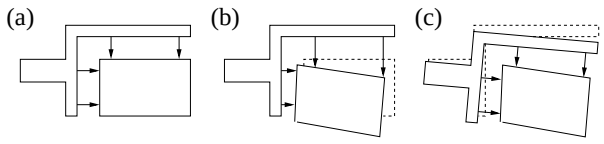

Fig. 1: Functionality of robot positioning system

A system for sensor controlled robots has to have some important features to suit the needs of the industry. Some of these qualities like a good usability can be realized when finally implementing the system, but others have to be taken into consideration much earlier when developing the basic methods. These are discussed in the following paragraph.

Modern technology offers a wide range of different kinds of sensors, each having its individual advantages and disadvantages. So it would be nice if the system has the flexibility to use different kinds of sensors even at the same time (“multi sensor system”), to give the user the ability to choose the best sensor type for a certain task. So the system should be general enough not to be a special solution for a

special problem. Other of these qualities are a quick and easy installation and a good maintainability. A time-consuming setup procedure or longer downtime due to complicated repairs reduces the availability and therefore the productivity. A replacement of a defect sensor for example or the addition of a further one should be possible without difficult re-calibration. To even avoid failures the system should be capable to handle smaller problems like temporary malfunction of single sensors without failure of the total system, which addresses the quality of robustness. Last but not least the system should be able to reach a high accuracy to realize the desired quality in the production process.

This paper presents a solution for a positioning system for industrial robots that solves the given problem and that sets value on the following qualities:

- High flexibility
- Easy installation, good maintainability
- Good robustness
- High accuracy

First of all Section 2 gives an overview over the system, Section 3 shows some references to related publications. Section 4 provides a more in-depth description of single interesting aspects. Section 5 is a short survey.

2 OVERVIEW

The three main components in this system are the robot, the sensors and the position control system itself. The robot usually is a general-purpose industrial robot with six joints and six degrees of freedom (*DOF*). Because the position control system requires a real-time position control interface to the robot, it has to provide such an interface. The position control system is implemented in software which runs on an industrial-proof computer system. This computer has to have an interface to connect it with the robot and one or more interfaces for the sensors involved. To allow the usage of different kinds of sensors, the system uses abstract data sources as input. Such a data source can be a channel of an I/O interface card as well as the result of an evaluation algorithm that works with more complex sensor data or the output of multiple sensors (see Section 4.1). During system setup the robot and the reference work object are posed in a way that should be reproduced during production. For self-calibration the system executes now a number of robot moves and automatically determines from the data collected during this procedure an approximate relationship between robot movements and sensor value changes (see Section 4.2). This knowledge is used during production to calculate robot position corrections from sensor values. The system establishes a closed-loop control to feed the robots real time interface with current correction values (see Section 4.3).

3 RELATED WORK

The usage of cameras as sensors to control manipulators and robots has a long history, starting at the first definition of term *visual servoing* in [7] in the year 1979. The most extensive overview over the field of visual servoing before 1993 can be found in [4]. This work is complemented by the tutorial [9] which offers an in-depth introduction into the subject as well as a classification of systems controlling robots with sensors. A more recent discussion can be found in [2] and [3].

The self-calibration capability of sensor-controlled robot systems is a common practice specified in many works, see for example [19], [18], [8] or [16]. Most approaches have in common the calculation of the Jacobian via the inversion of the feature Jacobian instead of a direct calculation of the Jacobian by solving a linear system as proposed in this paper. An interesting paper discussing the advantages for a direct calculation over the inversion is [12]. We have discussed in this paper the advantage of the direct calculation in respect to the weighting of sensor signals with similar information content and with different signal-noise ratio. The sensor signals carrying more noise are suppressed due to the least-squares condition. When determining the feature Jacobian, each DOF and each feature is taken separately which leads to equal-weighting of sensors with different noise ratio.

The thought of using an abstraction of sensors is quite common, a similar approach can be found in [5], where the term *logical sensors* is used for abstract sensors. The referred paper mainly focuses on the subject of multi-sensor data fusion.

4 DETAILS

4.1 Sensors

To be able to handle different kinds of sensors in the system calls for an abstraction of the sensor interface

(see Fig. 2). Sensors are pieces of hardware in the first place. After execution of a measurement we get raw sensor data. This data can be simple enough to be used directly for robot control like distance information from a laser point sensor measuring a single distance. But usually the data is more complex like a monochrome camera picture or a set of distance points from a laser stripe sensor. So there has to be a step where the sensor raw data is evaluated by an algorithm to extract interesting geometrical features like the position of a screw-nut in a camera picture expressed as two numerical values. We call these values *sensor signals*. A sensor signal is a numerical value expressing a feature independent from the sensor type, no matter if it is a value read from an I/O interface with some kind of external sensor attached to it or the result of a complex calculation on raw data. Sensor signals should be in the unit of millimeters or degrees to simplify conception when used in connection with robot movements but thats not obliged.

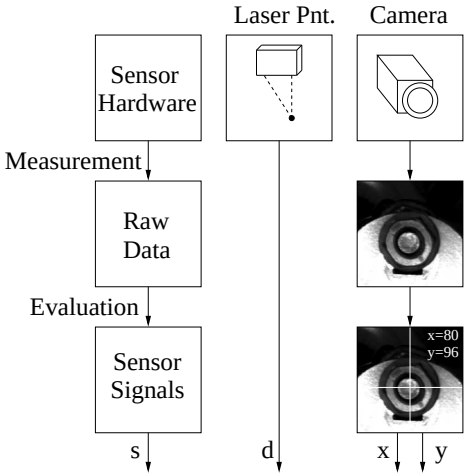

Fig. 2: Sensors and sensor signals

The mounting position of the sensors is completely free. In our system the sensors measure some geometry that has to change when the robot is moved and thats it. For convenience most sensors are mounted at the robots tool or at an extension of this tool. The latter one is called a *sensor tree*. There are sensors which signals change when the sensors mounting position is changed (like a distance sensor) and some are indifferent to change of position (like a camera measuring a gap width between two metal sheets). This is important to consider e.g. when you have to exchange faulty sensors and think about recalibration of the system.

4.2 Calibration

Let $m \in \mathbb{N}$ be the number of DOF of the robot that should be controlled (up to a number of 6 DOF). A robot status vector contains a number of coordinates in a fixed robot coordinate system (e.g. the robot base coordinate system or the work object coordinate system):

$$\mathbf{r} = \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix} \in \mathbb{R}^m \tag{1}$$

Let $n \in \mathbb{N}$ be the number of sensors signals used to control the robot. A sensor status vector contains the status of all sensor signals at a certain time:

$$\mathbf{s} = \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} \in \mathbb{R}^n \tag{2}$$

To control a robot on the basis of sensor information the knowledge of the unknown function

$$f : \begin{cases} \mathbb{R}^n & \longrightarrow & \mathbb{R}^m \\ \mathbf{s} & \longrightarrow & \mathbf{r} \end{cases} \tag{3}$$

would be helpful which describes the relationship between sensor signals and robot position. During the system (self-) calibration - called *training* - a linearized approximate solution for f in a certain working point is determined.

Before the actual training we pose the robot relative to a reference work object in the nominal position that the system should reproduce at different work objects during production. We save the nominal robot position $\mathbf{r}_0$ and the nominal sensor values $\mathbf{s}_0$. From now on all robot positions and sensor values are taken relative to these nominal values:

$$\Delta\mathbf{r} = \mathbf{r} - \mathbf{r}_0 \tag{4}$$

$$\Delta\mathbf{s} = \mathbf{s} - \mathbf{s}_0 \tag{5}$$

Now the robot performs a series of moves within a training range for each of the m DOFs to be controlled. It moves into the negative and positive direction of that particular DOF keeping all other DOFs constant and stopping at the nominal position again. During this movement the system continuously saves tuples of robot position and related sensor values. The set $\mathcal{T}$ containing all tuples $(\Delta\mathbf{r}_i, \Delta\mathbf{s}_i)$ with $i \in [1 \dots k]$ is called a *trace*, $k \in \mathbb{N}$ is the number of samples in the trace. The robot movements for each DOF are orthogonal in the robot position

vector space to get as much information about f in the area around the working point $\mathbf{r}_0$ as possible. Fig. 3 shows the result of an example trace in the X direction for three sensor signals plotted into a diagram. You can see that all signals change linearly when the robot moves in X between $-5\,\mathrm{mm}$ and $+5\,\mathrm{mm}$. Signal 3 shows the most significant change, signal 2 remains nearly zero.

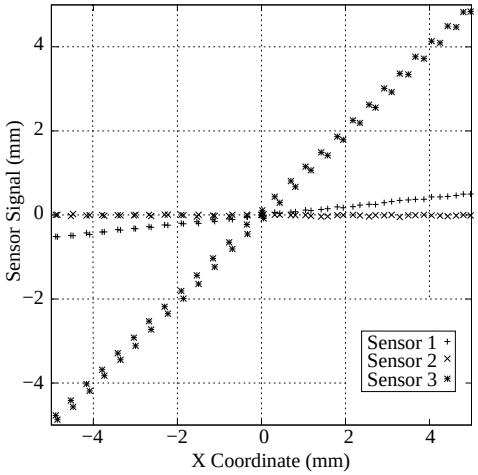

Fig. 3: Example trace for one DOF and three sensor signals

We see a noticeable gap between the trace data points of Signal 3 of the forward and the backward movement of the robot. There are two reasons for this behavior. One reason is the hysteresis as a mechanical property of the robot caused by effects like gear slackness and elasticities. The other reason is the delay between the sampling of the sensor data and the determination of the robot position which is not completely avoidable since the trace is recorded during robot movement and the involved components have only limited real time capabilities. For signal 3 this hysteresis has a size of about $0.15\,\mathrm{mm}$ measured in the sensor space.

We arrange the result of a trace as two matrices: The first matrix $\mathbf{R}$ has the size $k \times m$ and contains all trace robot positions as row vectors, the second matrix $\mathbf{S}$ has the size $k \times n$ and contains all trace sensor values as row vectors. The row-ordering of $\mathbf{R}$ and $\mathbf{S}$ is free, but the robot position of the i-th row vector of $\mathbf{R}$ has to correspond with the sensor values in the i-th row vector of $\mathbf{S}$ for all $i \in [1 \dots k]$. To get a linear solution for f we need to solve the overdetermined linear system

$$\mathbf{S} \cdot \mathbf{J} = \mathbf{R} \tag{6}$$

where $\mathbf{J}$ is the parameter matrix with a size of $n \times m$. Because the elements of $\mathbf{J}$ can be comprehended as partial derivatives of a single DOF with respect of a single sensor signal

$$\mathbf{J} = \left(\frac{\partial r_j}{\partial s_i} \right)_{1 \le i \le n, 1 \le j \le m} \tag{7}$$

the parameter matrix is a Jacobian matrix. In visual servoing this matrix is often referred as *inverse feature Jacobian.*

The solution for the linear system is calculated by using the pseudo inverse of $\mathbf{S}$ (see [13] and [14])

$$\mathbf{J} = \mathbf{S}^{+} \cdot \mathbf{R} \tag{8}$$

because $\mathbf{S}$ is hardly ever a square matrix. For calculation of the pseudo inverse the *singular value decomposition* (*SVD*) is quite applicable, see [6] or [15] for more information. The calculated solution is *least-squares* optimal which means it is optimized for a minimal residual square sum. The result of a training are the nominal values as well as the Jacobian. The Jacobian remains untouched during normal system operation.

Fig. 4 displays an example configuration of robot, sensors and work object. On the left side you see the base coordinate system of the robot, right besides it is a simple work object: A box. The robot moves a sensor tree with four sensors all measuring distances to the work object. We record a trace with the robot moving $\pm 5\,\mathrm{mm}$ in X, Y and Z containing $k = 2750$ samples in total. We first ignore sensor 4 and get the following parameter matrix:

$$\mathbf{J}_1 = \begin{pmatrix} 0.1015 & 0.9950 & 0.0002 \\ 0.0026 & -0.0131 & 0.9972 \\ 0.9966 & -0.1042 & 0.0012 \end{pmatrix} \tag{9}$$

Each row of the Jacobian reflects how one DOF is calculated from the sensor signals. So for the X direction the sensor signal of sensor 3 is used, for the Y direction signal 1 and for the Z direction signal 2 just like you would suppose because of the schematic. The signs are correct too because if the robot moves in positive direction, the measured distance increases for all sensors. Adding the fourth signal we get the following Jacobian:

$$\mathbf{J}_2 = \begin{pmatrix} 0.7876 & 0.3749 & -0.1549 \\ 0.0013 & -0.0120 & 0.9975 \\ 0.9952 & -0.1029 & 0.0015 \\ -0.6907 & 0.6242 & 0.1561 \end{pmatrix} \tag{10}$$

Now the Y direction is represented by signals 1 and 4 (see second row), but at a different rate (signal 1:

0.37, signal 4: 0.62). The reason for this is, that the Jacobian is result of a least-squares solution. Since the analysis of the signal 1 in the trace showed a slightly higher noise rate than signal 4, signal 4 is favoured. This is quite a nice feature because when two signals carry a similar information but one signal has a higher signal-noise ratio, the signal with higher noise rate is suppressed automatically because the solution is least-squares optimal. Another conspicuous fact are the high rates for signal 1 and 2 for the X direction with different sign each (signal 1: 0.79, signal 4: -0.69). These factors compensate for the sensitivity of signal 3 for the Y direction.

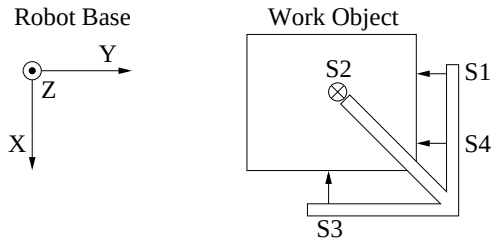

Fig. 4: Robot and sensor example configuration

A sufficient density of samples together with the way the training movements are executed are a precondition for getting enough information about the connection of robot movements and sensor value changes. But a well-conditioned solution requires a sensor configuration that is able to detect changes in every DOF that has to be controlled: If a movement in a certain DOF does not lead to significant sensor changes, this DOF can not be controlled with this sensor configuration. To get an idea of the quality of a given solution, we use the *coefficient of determination* (see e.g. [17] chapter 9.5). First we calculate the residual matrix $\mathbf{E}$ by

$$\mathbf{E} = \mathbf{S} \cdot \mathbf{J} - \mathbf{R}. \tag{11}$$

Each row of $\mathbf{E}$ contains the difference vector between the robot position estimated on the basis of the linear model of $\mathbf{J}$ and the true robot position. The coefficient of determination for each DOF is then defined by the term

$$R_j^2 = 1 - \frac{|\mathbf{e}_j|^2}{k \cdot \sigma_{\mathbf{r}_j}^2} \tag{12}$$

where $\mathbf{e}_j$ is the j-th column vector of $\mathbf{E}$, $\mathbf{r}_j$ the j-th column vector of $\mathbf{R}$ and $j \in [1 \dots m]$. Or written in detail

$$R_j^2 = 1 - \frac{\sum_{i=1}^{k} \mathbf{E}_{ij}^2}{\sum_{i=1}^{k} \mathbf{R}_{ij}^2 - \frac{1}{k}\left(\sum_{i=1}^{k} \mathbf{R}_{ij}\right)^2}. \tag{13}$$

The quality of the complete solution R^2 is the product of the coefficients of determination of each DOF

$$R^2 = \prod_{j=1}^{m} R_j^2. \tag{14}$$

R^2 is a value between 0 and 1 and gives an idea on how many percent of the output change (sensor signals) can be explained with the input changes (robot movements) under the given solution. So a value of $R^2 = 1$ indicates an ideal solution, $R^2 = 0$ a totally unusable solution. Reasons for a low quality can be among others a badly-conditioned sensor configuration (e.g. a change of a certain DOF does not lead to significant sensor value changes) or non-linearities (e.g. distance sensor moves over a step-shaped object contour).

Calculating the coefficients of determination for the example above and the Jacobian in (9) we get:

$$0.99825 \cdot 0.99930 \cdot 0.99917 = 0.99673. \tag{15}$$

After adding the fourth sensor signal and calculating the Jacobian in (10) the qualities are slightly increased:

$$0.99834 \cdot 0.99938 \cdot 0.99918 = 0.99690. \tag{16}$$

Even though the underlying data is real world data, the quality results of these trainings are very good. If we include a rotation around the Y axis of $\pm 0.5\,^\circ$ in the trace, the result becomes:

$$0.99669 \cdot 0.99919 \cdot 0.99912 \cdot 0.05321 = 0.052944. \tag{17}$$

A look on Fig. 4 shows that a rotation around the Y axis is hardly to detect with the given sensor configuration.

The system saves the complete trace data for later use. If a sensors fails during system operation the Jacobian can be recalculated without the missing sensor offering the user a continued operation at a certain loss of accuracy. This loss can be minimized when using redundant sensors.

4.3 Robot Control

Fig. 5 shows an overview over the control structure of the system. The nominal sensor values $\mathbf{s}_0$ acquired during training are used in connection with the current sensor values $\mathbf{s}$ to get the sensor value difference $\Delta\mathbf{s}$. This is the basis for calculating the current robot correction $\Delta\mathbf{r}$ using the Jacobian:

$$\Delta\mathbf{r} = \mathbf{J}^{\mathsf{T}} \cdot \Delta\mathbf{s}. \tag{18}$$

This correction is multiplied with a gain K_p (simple proportional controller) and sent to the robot controller. The robot controller moves the robot which changes the geometry between robot tool and work object. This leads to new sensor values that are evaluated in the next control step. The stop criterion stops the control process when the current correction $\Delta\mathbf{r}$ is small enough over a certain time (to avoid precipitate control stops on zero-crossings).

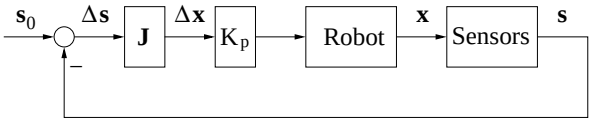

Fig. 5: Control flow diagram

Classifying the system after Hutchinson (see [9]) it is a *dynamic look-and-move* system because it does not directly control the robot but sends corrections to the original robot controller. It is further an *image-based control* system since the nominal values are sensor values or feature values. The sensors directly measure geometrical relationships between robot tool and work object so the system is an *endpoint closed-loop* (*ECL*) system.

The theory in this paper has been tested with robots from *KUKA Robotics Corporation*. Together with the *KRC2* controllers they provide a system that is called *Robot Sensor Interface* (*RSI*, see [10]) and is a real-time interface for external robots. It is part of the *KUKA Robot Language* (*KRL*) and consists of a toolbox for accessing sensors that are connected with I/O components of the robot controller. An extension developed by the company Amatec (see [1]) allows the access of the robot interface via Ethernet from external computer systems. This is the way the robot can be controlled from outside: The robot program moves the robot into the nominal position and passes control to the positioning system, this returns control to the robot after the positioning task. Other robot manufacturer are developing comparable interfaces, but none seems to be ready for the market yet.

The RSI/Corob robot interface accepts new correction values with the same frequency as the internal interpolator, every $12\,\mathrm{ms}$. If the sensor subsystem and the correction calculation is fast enough, the robot controller can be provided with new corrections at that speed. If the sensor subsystem is slower (and this is first of all the case if the evaluation of the sensor raw data is a time-consuming task, e.g. the feature extraction of camera pictures) there is not necessarily a problem because the electrical and mechanical components of robot are unable to adapt new corrections at that high speed. Additionally the speed of the robot can be controlled with the gain K_p. If the gain is too small, the robot moves very slowly towards the target. A gain that is too high lets the robot reach the area around target position very fast but might lead to an oscillation around the target position without reaching it in stable way.

Now we take a look at an example control process done with the setup described in the example in Fig. 4. The system controlled the directions X, Y and Z with three sensor signals, Fig. 6 shows the current actuation values $\Delta\mathbf{x}$ sent to the robot over the control time. The accuracy chosen as stop criterion was $0.01\,\mathrm{mm}$ for all DOFs over a time of $120\,\mathrm{ms}$, the control gain was $K_p = 0.2$, the resulting total control time was around $3.2\,\mathrm{s}$. Fig. 7 shows the sensor deviations from the nominal position $\Delta\mathbf{s}$ over the time. The data in both figures has been thinned out to make the single signals distinguishable.

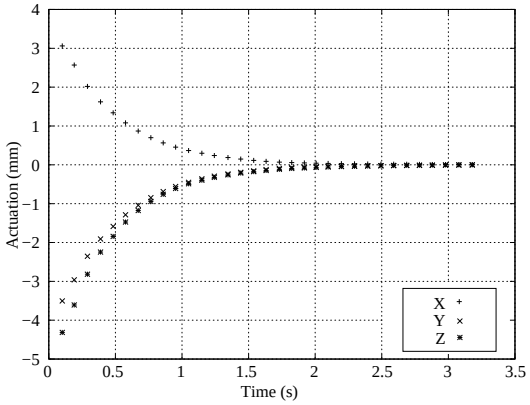

Fig. 6: Example control process: Actuation

5 CONCLUSIONS AND FUTURE WORKS

5.1 Conclusions

The system presented in this paper meets the quality requirements described in the introduction. Because

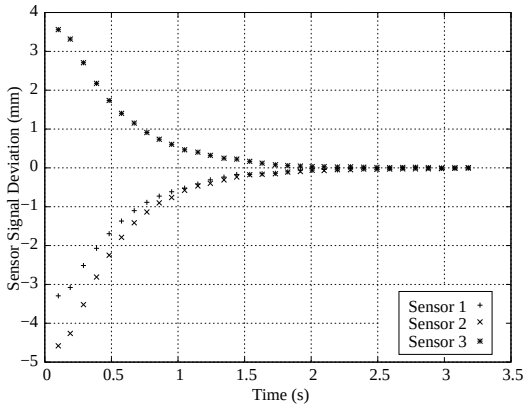

Fig. 7: Example control process: Signal Deviations

of the abstract sensor concept the system is flexible enough to handle different kinds of sensors to obtain the best results. The setup process involves an automated self-calibration process which makes the system easy to install and easy to maintain. With the data collected in the calibration the system is able to reconfigure even on-the-fly within the control process if single sensors fail which makes the system quite robust. Since the system uses a closed-loop control in connection with the real-time robot interface it obtains high accuracies. So the relative positioning to the work object is better even than the repeat accuracy of the robot. The robot in the control process example above had a repeat accuracy of $\pm 0.15\,\mathrm{mm}$ but it was possible to position it relative to the work object with a remarkable accuracy of $\pm 0.01\,\mathrm{mm}$ in a stable way. This makes the usage of the system interesting in connection with applications with high accuracy positioning demands.

5.2 Future Works

A nice future extension of the system would be the ability to integrate new sensor signals into the system without running a new training. When adding a new signal the control process is executed using the old signals, but the values of all signals and the robot position during control are logged. With this logging information, lets call it *control trace*, it should be possible to identify the properties of the new signal and integrate it without the need of a new training. This would make sensor addition and replacement fairly easy.

Another subject that needs further attention is the evaluation of the non-linearity-effects that occur in the traces caused by the robot mechanics. For one thing there is the robot hysteresis and for another thing some gear effects that cause the appearance of saw-shaped signals in the trace diagram. Furthermore it would be interesting to develop a method to automatically detect the linearity range for each DOF using the trace data before actually calculating the Jacobian.

References:

[1] Amatec Robotics Corporation, *RSI-Ethernetinterface: Corob*, Version 3.6.0

[2] F. Chaumette, S. Hutchinson, *Visual Servo Control Part I: Basic Approaches*, IEEE Robotics & Automation Magazine, Vol. 13, Issue 4, Pages 82-90, Dec. 06

[3] F. Chaumette, S. Hutchinson, *Visual Servo Control Part II: Advanced Approaches*, IEEE Robotics & Automation Magazine, Vol. 14, Issue 1, Pages 109-118, Mar. 07

[4] P. I. Corke, *Visual control of robot manipulators - A review*, Visual Servoing K, Hashimoto Ed. Singapore: World Scientific, Pages 1-31, 1993

[5] K. Feldmann, U. Schonherr, J. Zeller, *Multisensor integration for sensor guided robots*, Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems '94 (IROS 1994), Volume 3, Pages 1730-1735, Sept. 1994

[6] G. H. Golub, C. F. Van Loan, *Matrix Computations*, Chapter 8.6: "Computing the SVD", The Johns Hopkins University Press, 1996

[7] J. Hill, W. T. Park, *Real time control of a robot with a mobile camera*, Proceedings of the 9th International Symposium on Industrial Robotics '79 (ISIR 1979), Pages 233-246, Mar. 1979

[8] B. Huang, V. Milenkovic, *Method and system for correcting a robot path*, U.S. Patent Number 4945493, applied for 1988-09-26, received 1990-07-31

[9] S. A. Hutchinson, G. D. Hager, P. I. Corke, *A tutorial on visual servo control*, IEEE Trans. Robotics and Automation, Volume 12, No. 5, Pages 651-670, Oct. 1996

[10] KUKA Robotics Corporation, *Robot Sensor Interface (RSI)*, Release 2.0

[11] F. Lange, G. Hirzinger, *Stability Preserving Sensor-Based Control for Robots with Positional Interface*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2005), Pages 1700-1705, Apr. 2005

[12] J. T. Lapreste, F. Jurie, M. Dhome, F. Chaumette, *An Efficient Method to Compute the Inverse Jacobian Matrix in Visual Servoing*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2004), Volume 1, Pages 727-732, Apr. 2004

[13] E. H. Moore, *On the reciprocal of the general algebraic matrix*, Bulletin of the American Mathematical Society 26, Pages 394-395, 1920

[14] R. Penrose, *A generalized inverse for matrices*, Proceedings of the Cambridge Philosophical Society 51, Pages 406-413, 1955

[15] W. H. Press, A. S. Teukolsky, W. T. Vetterling, *Numerical Recipes in C*, Chapter 2.6: "Singular value decomposition", Cambridge University Press, 01/1993

[16] P. Roebrock, Kay Böhnke, *Offline Path Correction System for Industrial Robots*, 9th WSEAS Int. Conf. on Automatic Control, Modelling & Simulation (ACMOS'07), Istanbul Turkey May 2007

[17] S. M. Ross, *Introduction to Probability and Statistics for Engineers and Scientists*, Academic Press, 3rd Edition, July 2004

[18] Y. Veryba, L. Kourtch, B. Verigo, V. Murashko, V., *Method and system for robot end effector path correction using 3-D ultrasound sensors*, Proceedings of the 3rd World Congress on Intelligent Control and Automation 2000, Volume 2, Pages 1240-1243, 2000

[19] L. Weiss, *Dynamic Visual Servo Control of Robots: An Adaptive Image-Based Approach*, PhD Thesis, CMU-RI-TR-84-16, Carnegie Mellon University, Apr. 1984