Buletinul Ştiinţific al Universităţii “Politehnica” din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICATII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 51(65), Fascicola 1, 2006

TCP Identification of contactless measurement systems

Philipp Roebrock[1]

Abstract - This paper describes an algorithm to determine the robot tool transformation and tool center point (TCP) for contactless measurement systems. Valid types of sensors are those who provide metric information about one or multiple points within the sensors coordinate system. The reference objects are geometrical primitives (planes, spheres). The algorithm starts with estimated values for the tool transformation and the reference object definition and optimizes them in an iterative process. The optimisation result is tested for convergence with a simulation.

Keywords: sensor, robot tool transformation, TCP, tool center point, optimisation, Jacobian matrix

I. INTRODUCTION

In most applications with industrial robots it is common to have sensor systems with one or more sensors. The information gathered with a sensor system is mostly used for robot guidance or for quality assurance. If a sensor is part of the robot tool it is recommendable to identify its tool transformation and the origin of the sensor coordinate system, the tool center point (TCP). The tool transformation describes the transformation from the robot flange coordinate system to the tool coordinate system thus in this case the sensor coordinate system. There are some reasons for the determination of the tool transformation:

1. With a known tool transformation the robot can be moved in respect of the tool coordinate system and the tool center point. This makes teaching of robot programs much easier.
2. If the sensor has to be replaced due to a sensor malfunction it cannot be guaranteed that the replacement sensor can be mounted at exactly the same position the defect sensor was. If the robot programs were teached in respect of the tool coordinate system, the new tool transformation has to be determined and the problem is solved. Otherwise all robot programs have to be modified to fit to the new sensor position. Under normal circumstances this is a very time consuming and therefore expensive task.
3. With a known tool transformation the information provided by the sensor can referred to the base coordinate for each robot pose. This means the local sensor information becomes global information. In a multisensor system it may be a requirement to have a common reference for all sensor information.

The robot manufacturers provide software tools implemented in the robot control system to determine the tool transformation for common robot tools like grippers or welding tools. In the classical 4-point method (see e.g. [1], page 33ff) the TCP of the tool is moved to a fixed reference point from four different directions. The translative component of the tool transformation can be calculated together with an error estimation basing on four different positions and orientations of the robot flange. For most applications the knowledge of the translative component of the tool transformation is sufficient. But the rotational component of the tool transformation can be identified with similar methods. Either the robot has to be moved along the coordinate axes of the tool coordinate system or the tool coordinate system has to be aligned to the axes of the base coordinate system of the robot in a special way.

But these methods are not satisfactory to determine a sensors tool transformation. Sensors do not have a certain physical point as their TCP. The TCP lies somewhere in the measurement range of the sensor. It is difficult to use one of the methods described above to move the sensors TCP to the reference point even with the usage of software tools to display the current sensor information. The usage of special devices mounted at the sensor to mark the sensors TCP with a physical point may make the determination of the tool transformation easier but there is still a problem of the low accuracy of these methods. Mainly the determination of the rotational component of the tool transformation is quite inaccurate. The accuracy requirements for the TCP transformation may vary from application to application. But in most use cases the sensors TCP transformation has to be known in all six degrees of freedom with an accuracy that is not much lower than most inaccurate system component (sensor or robot).

This paper describes a method to optimize a given estimated TCP transformation determined by CAD data or by the methods provided by the robot manufactures. The algorithm is iterative and is based on sensitivity matrices and Jacobian matrices. A similar approach is described in [2]. This solution uses simple reference objects and free selectable robot poses too, uses similar error functions, but the optimisation is based on statistical methods (Bayesian

[1]Fachhochschule Gelsenkirchen, University of Applied Sciences, Faculty of Computer Science, Neidenburger Strasse 43, 45877 Gelsenkirchen, Germany, philipp.roebrock@informatik.fh-gelsenkirchen.de

search algorithms).

II. SYSTEM STRUCTURE

The total systems structure is shown in Figure 1. The sensor is attached to the end-effector of the robot. The sensor is a device that provides information about the position of a number of points within its field of view based on its sensor coordinate system. A simple example for a sensor would be a laser distance sensor which determines the position of a single point in its coordinate system (e.g. by triangulation or by time-of-flight). Other examples for sensor systems are laser stripe sensors (position of multiple points along a laser line), 3D cameras (position of multiple points of a matrix) or other intelligent camera systems. Important condition is that the sensor is calibrated and provides metric information. The sensor has a reference object within its field of view. The reference object is the representation of a geometrical primitive like a plane or a sphere. The position of the reference object is defined in the robots base coordinate system. Estimated values for the tool transformation and the reference object definition are known.

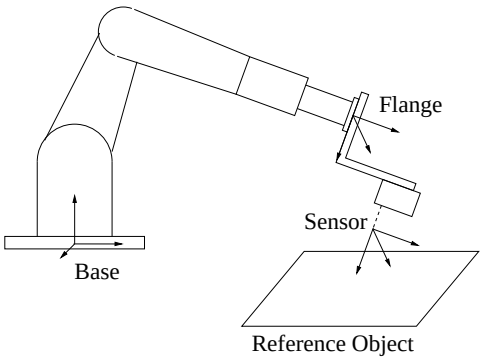

Figure 1: Basic System Structure

III. ERROR FUNCTION

First objective before optimizing the estimated values is the definition of an error function that allows the evaluation of a given tool transformation for the current robot pose. As defined above the sensor provides information about the position of a number of m points within its sensor coordinate system:

$$^{\text{Sensor}}\underline{p}_j \quad \text{with } j \in \{1, \ldots, m\} \tag{1}$$

These points have to be transformed into the base coordinate system in which the reference object is defined. This is the necessary transformation:

$$^{\text{Base}}\underline{\underline{T}}_{\text{Sensor}} = {}^{\text{Base}}\underline{\underline{T}}_{\text{Flange}} \cdot {}^{\text{Flange}}\underline{\underline{T}}_{\text{Sensor}} \tag{2}$$

And this transforms all points into the base coordinate system:

$$^{\text{Base}}\underline{p}_j = (^{\text{Base}}\underline{\underline{T}}_{\text{Sensor}})^{-1} \cdot {}^{\text{Sensor}}\underline{p}_j \tag{3}$$

Using an optimal reference object definition and an optimal tool transformation, all points $^{\text{Base}}\underline{p}_j$ measured by the sensor should be points of the reference object. But because of the inaccuracy of the estimated values there is a certain deviation. The sum of the deviation for every single point is a good error function. The error function is called "E".

If for example the reference object is a plane and the sensor a laser stripe sensor, the error function can be expressed as followed. The plane can be defined in the Hessian normal form:

$$\underline{n} \cdot \underline{x} - d = 0 \quad \text{with } |\underline{n}| = 1 \tag{4}$$

Because of

$$|\underline{n}| = \left| \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \right| = \sqrt{n_x^2 + n_y^2 + n_z^2} = 1 \tag{5}$$

it is possible to reduce one more degree of freedom in the plane description:

$$n_z = \sqrt{1 - n_x^2 - n_y^2} \tag{6}$$

The plane definition is now dependent from three factors: from n_x, n_y and d. The error function can be defined as the average distance of the distances of all points to the plane:

$$E = \frac{1}{m} \sum_{j=1}^{m} \left| \underline{n} \cdot {}^{\text{Base}}\underline{p}_j - d \right| (\tag{7}$$

Figure 2 shows the laser stripe sensor points transformed into the base coordinate system and their distances to the plane as reference object.

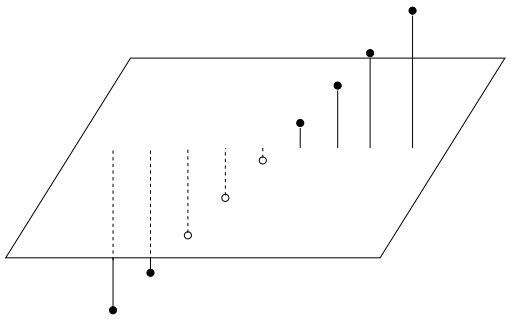

Figure 2: Laser Stripe Sensor Points

IV. OPTIMIZATION

The optimisation process is an iterative process that optimizes the tool transformation and (as a byproduct) the definition of the reference object. The number of factors to be optimized are up to six for the tool transformation (x, y, z, α, β, γ) and for the reference object definition usually three (plane or sphere with known radius), in total o factors. May

$$\underline{F} = (F_1, \ldots, F_o) \tag{8}$$

be a vector that contains all factors that should be optimized within the optimisation process and

$$\underline{f} = (f_1, \ldots, f_o) \tag{9}$$

current values for the optimisation factors (initially the estimated ones). Because of the high number of factors to optimize one single robot pose is not a sufficient base for the optimisation process. The base is a set of l different robot poses which have to vary in all degrees of freedom to provide enough information about the system structure:

$$^{\text{Base}}\underline{\underline{T}}_{i\ \text{Flange}} \quad \text{with } i \in \{1 \dots l\} \tag{10}$$

and a fixed set of m sensor points for each of the l poses:

$$^{\text{Sensor}}\underline{p}_{i,j} \quad \text{with } i \in \{1 \dots l\} \text{ and } j \in \{1 \dots m\} \tag{11}$$

Because the error function depends on the robot pose and a vector of optimisation values the error function becomes:

$$E = E(i, \underline{F}) \quad \text{with } i \in \{1 \dots l\} \tag{12}$$

For a given vector of optimisation factors $\underline{f}$ the error vector $\underline{e}$ is defined as

$$\underline{e} = (e_1, \dots, e_l) \quad \text{with } e_i = E(i, \underline{f}) \tag{13}$$

The sensitivity matrix $\underline{\underline{S}}$ describes how the result of the error function for a certain robot pose depends on a change of the optimisation factors in the near of the current optimisation factors f:

$$s_{i,k} = \left(\frac{\partial E(i, \underline{F})}{\partial F_k} \right)_{\underline{f}} \tag{14}$$

One single matrix element of $\underline{\underline{S}}$ can be calculated by a simple difference quotient:

$$\begin{aligned} s_{i,k} \quad = \quad & \frac{1}{2\delta_k} \cdot [E(i, (f_1, \dots, f_k + \delta_k, \dots, f_o)) - \\ & E(i, (f_1, \dots, f_k - \delta_k, \dots, f_o))] \end{aligned} \tag{15}$$

The Jacobian matrix J is the inverse of the sensitivity matrix and describes how the optimisation factors depend on a change of the result of the error function for a certain robot pose near the current optimisation factors f:

$$j_{k,i} = \left(\frac{\partial F_k}{\partial E(i, \underline{F})} \right)_{\underline{f}} \tag{16}$$

Because $\underline{\underline{S}}$ is rarely a square matrix, $\underline{\underline{J}}$ is calculated using the Moore-Penrose matrix inverse of $\underline{\underline{S}}$ (see [3]):

$$\underline{\underline{J}} = \underline{\underline{S}}^{+} = \left(\underline{\underline{S}}^{\mathrm{T}} \cdot \underline{\underline{S}} \right)^{-1} \cdot \underline{\underline{S}}^{\mathrm{T}} \tag{17}$$

If a vector of optimisation factors $\underline{f}$ is available and the corresponding error vector $\underline{e}$ and the Jacobian matrix $\underline{\underline{J}}$ have been determined, the improved vector of factors $\underline{f'}$ is calculated by:

$$\underline{f'} = \underline{f} - \left(\underline{\underline{J}} \cdot \underline{e} \right) \tag{18}$$

The complete optimisation algorithm is now:

1. Start with an initial vector of estimated optimisation factors $\underline{f}$ and with an fixed set of robot poses and sensor values.
2. Calculate error vector $\underline{e}$ for $\underline{f}$
3. Calculate Jacobian matrix $\underline{\underline{J}}$ for $\underline{f}$
4. Calculate new optimisation factors $\underline{f'}$
5. Stop if distance between $\underline{f}$ and $\underline{f'}$ is small enough
6. Set $\underline{f} = \underline{f'}$ and continue at step 2.

An alternative stop criterion would be a check if the current error vector $\underline{e}$ is small enough.

V. SIMULATION

To check the usability of the algorithm and the convergence behavior of the optimisation a simulation in MATLAB (see [4]) has been implemented. Part of the simulation is a module for the kinematics of an industrial robot. Mounted at the robots hand is a simulated laser stripe sensor with a plane as reference object in his field of view. The error function has been implemented as shown above (see (7)). With this simulation it is possible to test different robot poses with a correctly defined tool transformation and reference object ($E = 0$) and with slightly wrong tool transformations and reference objects ($E \neq 0$).

The simulated sensor provides position information about 10 points in its coordinate system, base of the optimisation are 20 robot poses. The factors to optimize are 9 in total:

$$\underline{F} = (x, y, z, \alpha, \beta, \gamma, n_x, n_y, d) \tag{19}$$

The sensitivity matrix is:

$$\begin{pmatrix} \frac{\partial E(1,\underline{F})}{\partial x} & \frac{\partial E(1,\underline{F})}{\partial y} & \cdots & \frac{\partial E(1,\underline{F})}{\partial d} \\ \frac{\partial E(2,\underline{F})}{\partial x} & \frac{\partial E(2,\underline{F})}{\partial y} & \cdots & \frac{\partial E(2,\underline{F})}{\partial d} \\ \vdots & \vdots & & \vdots \\ \frac{\partial E(20,\underline{F})}{\partial x} & \frac{\partial E(20,\underline{F})}{\partial y} & \cdots & \frac{\partial E(20,\underline{F})}{\partial d} \end{pmatrix} \tag{20}$$

And this is the Jacobian matrix:

$$\begin{pmatrix} \frac{\partial x}{\partial E(1,\underline{F})} & \frac{\partial x}{\partial E(2,\underline{F})} & \cdots & \frac{\partial x}{\partial E(20,\underline{F})} \\ \frac{\partial y}{\partial E(1,\underline{F})} & \frac{\partial y}{\partial E(2,\underline{F})} & \cdots & \frac{\partial y}{\partial E(20,\underline{F})} \\ \vdots & \vdots & & \vdots \\ \frac{\partial d}{\partial E(1,\underline{F})} & \frac{\partial d}{\partial E(2,\underline{F})} & \cdots & \frac{\partial d}{\partial E(20,\underline{F})} \end{pmatrix} \tag{21}$$

Table 1 shows an optimisation process of eight steps. The first row shows the deviation of the tool transformation from the correct tool transformation in all degrees of freedom. The other rows show the progress of the optimisation process. After eight steps the size of the deviation is smaller than the repeat accuracy of an common industrial robot. The values for n_x, n_y and d are skipped. Figure 3 and Figure 4 display the deviations over the optimisation steps.

Simulations with different random start deviations from optimal tool transformation have shown that a distance up to $10\,mm$ for x, y and z, a distance of $5°$ for α, β and γ and a distance of $0.1\,mm$ for n_x, n_y and d lead to certain convergence to the optimal transformation: the global minimum. If the deviation raises up to $15\,mm$, $7.5°$ and $0.2\,mm$ it is

x	y	z	α	β	γ
-6.983	-3.958	-2.433	3.537	0.936	3.600
-2.780	-2.798	-0.435	1.256	0.966	3.538
-0.042	-0.275	-0.025	0.026	0.123	3.541
0.068	-0.202	-0.096	-0.036	0.046	1.750
0.047	-0.093	-0.057	-0.026	0.018	0.870
0.039	-0.051	-0.030	-0.020	0.008	0.432
0.045	-0.029	-0.025	-0.019	0.004	0.212
0.030	-0.020	-0.016	-0.012	0.002	0.105
0.021	-0.009	-0.025	-0.007	0.001	0.053

Table 1: Simulation Result

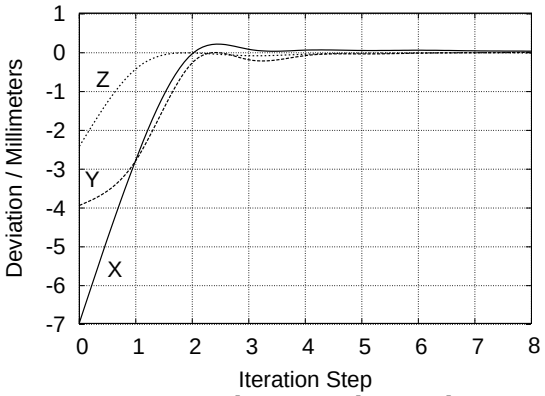

Figure 3: Simulation Result: Translation

not guaranteed that the optimisation process finds the optimal values. Result is a local minimum or a divergence. The convergence behavior shows that there is some slight overshooting. For speeding up the optimisation this can be compensated by introducing an attenuation factor $\lambda \leq 1$ applied when calculating the correction:

$$\underline{f'} = \underline{f} - \lambda \cdot \left(\underline{\underline{J}} \cdot \underline{e}\right) \tag{22}$$

The attenuation factor can be fixed or dynamically adapted.

The robot poses have to be chosen carefully. The set of robot poses has to contain variations for all degrees of freedom of the robot position to ensure a well-conditioned sensitivity matrix. In case of an ill-conditioned sensitivity matrix the Jacobian matrix can not be determined and the optimisation fails.

An additional note about the accuracy of the tool transformation that is result of the optimisation: the resulting tool transformation is an artificial entity that has no physical counterpart. The result depends on the selected robot poses and the chosen workspace where the robot poses are located. It depends on the all errors of the robot and the sensor system. It is the transformation that solves the given problem in an optimal way. If choosing different work spaces the resulting tool transformation varies.

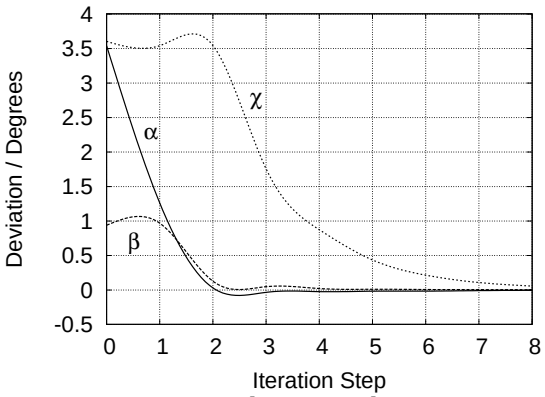

Figure 4: Simulation Result: Rotation

VI. MODIFICATION

If the sensor is not attached to the robot tool but fixed in the robot cell and its important to know the sensors position in respect of the robot its possible to use the algorithm from above with a slight modification. Figure 5 shows the changed system structure. Attached to the robots tool is the reference object which is defined in the robots flange coordinate system. This transforms all sensor points to the robots flange coordinate system:

$$^{\text{Flange}}\underline{p}_j = \left(^{\text{Base}}\underline{\underline{T}}_{\text{Flange}}\right)^{-1} \cdot {}^{\text{Base}}\underline{\underline{T}}_{\text{Sensor}} \cdot {}^{\text{Sensor}}\underline{p}_j \tag{23}$$

From here the solution is the same as for the base problem. The definition of the reference object in the flange coordinate system and the transformation from base to sensor are the factors to be optimized.

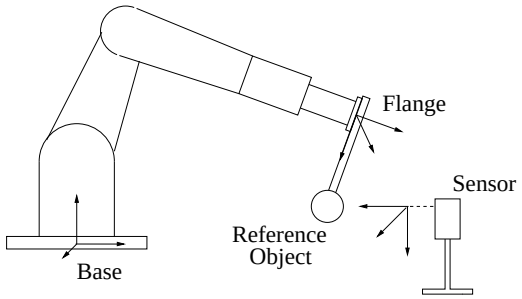

Figure 5: Modified System Structure

VII. CONCLUSION

This paper has demonstrated a method to determine a sensors tool transformation, a problem that occurs in practical industrial applications with robots. The method is based on an optimisation using Jacobian matrices. The convergence quality of the algorithm has been confirmed with a software simulation.

REFERENCES

[1] KUKA Roboter GmbH: *KRC2/KRC3 KUKA System Software 5.2: Start-Up*, Augsburg/Germany, 2003-09-26

[2] Mikko Sallinen, Tapio Heikkilä: *A simple Hand-Eye Calibration Method for a 3D Laser Range Sensor*, VTT Automation Finland

[3] Eric W. Weisstein: *Moore-Penrose Matrix Inverse*, From MathWorld, A Wolfram Web Resource (http://mathworld.wolfram.com/Moore-PenroseMatrixInverse.html)

[4] MATLAB, The MathWorks Company, Natik/MA USA (http://www.mathworks.com/products/matlab/)