Computing the worst case accuracy of a PKM over a workspace or a trajectory

Merlet, J-P.

Abstract

Accuracy is clearly an important feature of PKM. Still it is difficult to evaluate what will be the influence of the sensor errors on the positioning errors except at one particular pose. We present a method that allow us, provided an analytical form of the inverse jacobian matrix of the robot, to estimate safely if the positioning errors may exceed some given thresholds over a given workspace or trajectory. This method is generic, i.e. it may be used for any PKM or any type of workspace or n-dimensional parametric trajectory (i.e. a surface, a volume).

1 Introduction

Determining the accuracy of a parallel robot is a problem that is of high practical interest.

Inaccuracy of parallel robots are due to

- *geometrical errors* in the robot modeling: the influence of the manufacturing tolerances on the positioning errors have been studied [7, 10, 17, 20, 22, 24], although the study is usually reduced to a few poses in the workspace. More complete analysis has been performed for specific robot such as the 3-UPU robot [5, 16, 25]. A general approach to evaluating the positioning errors in one pose has been proposed by Pott [18]: it relies however on a numerical evaluation that requires solving the direct kinematics, and is thus computer intensive. The influence of such errors may be decreased by an appropriate calibration and this issue will not be addressed in this paper.

- *thermal effect*: this cause seems to have only a real influence for high accuracy positioning devices [3, 15]. It was mentioned for heavy duty robots, although

few works substantiate the claim that they have a significant effect [23]. Sellgren [21] proposes using thermal sensors to correct this effect, and poses the location of these sensors as a design problem. Pritschow [19], however, states that cooling is the most effective measure, as the thermal model of parallel robots is complex.

- *joint sensor errors*: this is usually the largest source of error and the one we will address in this paper

The errors $\Delta\boldsymbol{\Theta}$ in the sensor measurements are linearly related to the positioning errors $\Delta\mathbf{X}$ through the inverse jacobian, denoted $\mathbf{J}^{-1}$, that is pose dependent:

$$\Delta\boldsymbol{\Theta} = \mathbf{J}^{-1}(\mathbf{X})\Delta\mathbf{X} \tag{1}$$

Many papers have addressed the determination of the inverse jacobian $\mathbf{J}^{-1}$ of parallel robots and for most parallel robots this matrix may be determined in closed-form (for most parallel robots $\mathbf{J}^{-1}$ has as rows the Plücker vectors of well-defined lines.). On the opposite the jacobian matrix cannot usually be established in closed-form or this closed-form is so complex that it is useless.

After having determined $\mathbf{J}^{-1}$ it is necessary to determine accuracy indices that allow to quantify the performance of the robot. A possible error amplification factor for the system (1) expresses how a **relative** error in $\boldsymbol{\Theta}$ gets multiplied and leads to a **relative** error in $\mathbf{X}$. It characterizes in some sense the dexterity of the robot and has been proposed as a performance index and is known as the inverse of the *condition number*. It is often claimed that the condition number of $\mathbf{J}^{-1}$ is a good index to qualify the accuracy of parallel robots [1, 2, 4]. However this index has to be well understood to be used efficiently. First of all there is not a unique definition for the condition number as its value depends on the choice of the matrix norm that is used for its calculation. Second a proper definition of the condition number is difficult as soon as the d.o.f. of the robot have not the same nature (i.e. translation or rotation). Third this amplification factor is valid only for *relative* errors and not for *absolute* one, although the accuracy analysis should focus on this absolute error. To illustrate that we have considered in a recent paper [11] three different poses for a given parallel robot of the Gough type and have computed the maximal value of each component of $\mathbf{X}$ for given bounded sensor errors. The poses $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3$ were chosen so that they can be sorted according to the accuracy: each positioning error for $\mathbf{X}_1$ was lower than the one in $\mathbf{X}_2, \mathbf{X}_3$ while 5 out of 6 errors were lower for $\mathbf{X}_2$ than for $\mathbf{X}_3$ and the sixth one was very close. Various condition numbers have

then be calculated for each pose and it was shown that most of the time the ranking according to the condition number was not the same than the accuracy ranking. The ranking was not always even coherent between two different condition numbers.

Finally we should mention that the condition number is difficult to manage as soon as we are interested in its value not only at a given pose but, for example, over a workspace or on a trajectory, as there is no explicit general formula that allows to compute it.

The problem we will address in this paper is a *verification* problem: being given a robot geometry (possibly with uncertainties), bounds on the sensor errors and a given workspace (i.e. here a motion variety of degree 1 to n for a n d.o.f. robot) we will check that the maximum of the positioning errors does not exceed some thresholds.

An important point is that the result of the algorithm should be *certified* even with respect to numerical round-off errors. Our algorithm is certified and will provide three different types of result:

- *workspace is safe*: for any pose within the prescribed workspace the positioning will be lower than the thresholds whatever are the uncertainties
- *workspace is unsafe*: for some poses in the workspace some positioning errors will be larger than the thresholds
- *unable*: the algorithm is unable to assert the accuracy constraint. This may occur because the computer numerical errors do not allow to establish the result or because at some poses the positioning errors are very sensitive to the uncertainties: for some values of these uncertainties the positioning errors may be lower than the threshold while for some other values they will be larger

This certification constraint usually rules out the use of an optimization procedure, especially as we will assume uncertainties on the geometrical model of the robot. Furthermore using an optimization procedure may be an overkill: in the verification step our objective is **not** to calculate the maximal positioning errors but just to verify that the maximal positioning errors are lower than some thresholds. We will see that our algorithm is efficient because it will almost never try to calculate exactly the maximum.

2 A maximal error calculation scheme

First of all we notice that (1) implies that the maximal positioning errors will be obtained when the sensor errors have for values either the lower or upper bound of their respective range. Without lack of generality we will assume here that the sensors have all the same error ranges.

We consider here a workspace W defined as a closed region in the 6-dimensional generalized coordinates space with 3 parameters describing the location of the center of the platform and 3 angular parameters allowing to describe the orientation of the platform (robot with less than 6 d.o.f. may be treated as well).

2.1 Workspace and boxes

We define a *box* as a set of 6 ranges, one for each of the parameter that allows to describe the pose of the platform. Hence a box defines a 6D workspace for the robot. Reciprocally almost any workspace type may be approximated by a set of boxes.

In the sequel we will use an operator for a box called *bisection*. This operator takes as input a box and outputs two new boxes whose union is the initial box. These new boxes are obtained by splitting in half one of the 6 ranges $I_1, \ldots, I_6$ of the initial box. For example if the initial box is defined by the following ranges:

$$[-10, 10], [-10, 10], [40, 50], [-10, 10], [-10, 10], [-10, 10]$$

and if we bisect the first range, then the output of the bisection will be the 2 boxes defined by:

$$[-10, 0], [-10, 10], [40, 50], [-10, 10], [-10, 10], [-10, 10]$$
$$[-10, 0], [-10, 10], [40, 50], [-10, 10], [-10, 10], [-10, 10]$$

2.2 Maximal positioning errors calculation in a workspace

We consider a box $\mathcal{B}$ and we assume that we have an algorithm $\mathbf{A}(\mathcal{B})$ which is able to determine a box $\mathcal{D}$ that enclose all solutions in $\Delta\mathbf{X}$ of the set of linear systems:

$$\mathbf{J}^{-1}(\mathcal{B})\Delta\mathbf{X} = \Delta\mathbf{\Theta} \tag{2}$$

This equation defines a set of linear systems as the matrix $\mathbf{J}^{-1}$ is pose dependent: hence for each pose in the box $\mathcal{B}$ a different system is obtained. Furthermore the

right-hand side term of this equation will be constituted of all possible combinations of extremal sensor errors (for a 6 d.o.f. robot there will $2^6 = 64$ such combination). The box $\mathcal{D}$ is defined as a set of ranges $[\underline{d_i}, \overline{d_i}]$, one for each element of $\Delta\mathbf{X}$. Note that these ranges may be overestimated but should be exact if the box is reduced to one pose (up to numerical round-off errors). The prescribed threshold for the i-th component of $\Delta\mathbf{X}$ will be denoted ϵ_i.

The algorithm will return:

- 1 if for all i $[\underline{d_i}, \overline{d_i}]$ is included in $[-\epsilon_i, \epsilon_i]$
- -1 if for at least one i we have $\underline{d_i} > \epsilon_i$ or $\overline{d_i} < -epsilon_i$
- 0 if for at least one i we have $[-\epsilon_i, \epsilon_i]$ included in $[\underline{d_i}, \overline{d_i}]$

In case 1 we can guarantee that for any pose in the box $\mathcal{B}$ the robot satisfy the accuracy constraint. This is the opposite for case -1: for all poses in $\mathcal{B}$ the robot violates at least one accuracy constraints. Case 0 will occur because of the overestimation implied by the method we will use for the **A** algorithm.

The purpose of this section is to show that then we are able to verify the accuracy constraint for any type of workspace and any type of robot.

Let assume that we want to check the accuracy constraint in a workspace W defined by a 3D volume V describing the possible location of the center of the end-effector and three ranges I_ψ, I_θ, I_ϕ which describe the possible values of the orientation angles. Clearly V may be approximated by a set of boxes.

Let us assume now that we are able to design a test algorithm $\mathbf{T}(\mathcal{B})$ which enable to determine if for the box $\mathcal{B}$:

1. the location part of $\mathcal{B}$ is fully inside V
2. the location part of $\mathcal{B}$ is fully outside V
3. the location part of $\mathcal{B}$ is partially inside V

The algorithm **T** will return an integer which may 1, 2 or 3 according to the status of the location part with respect to V.

Using the algorithms **A** and **T** we are able to design an algorithm which enable to check the accuracy constraint within W. The algorithm start by computing a list S=$\{\mathcal{B}_0, \mathcal{B}_1, \ldots \mathcal{B}_{n-1}\}$ of n boxes such that the union of the location part of the boxes in the list is an approximation of the volume V, strictly including V, while the orientation part of each box is simply defined as the range I_ψ, I_θ, I_ϕ. During the process

new boxes may be added to the list but n will always denote the total number of boxes in the list.

We may now describe the different steps followed by the algorithm at iteration k, the algorithm starting at iteration 0:

1. if $k > n - 1$ return **WORKSPACE IS SAFE**
2. if $\mathbf{T}(\mathcal{B}_k) = 2$ then $k = k + 1$ and reiterate
3. if $\mathbf{A}(\mathcal{B}_k) = 1$ then $k = k + 1$ and reiterate
4. if $\mathbf{A}(\mathcal{B}_k) = -1$ and $\mathbf{T}(\mathcal{B}_k) = 1$ then return **UNSAFE WORKSPACE**
5. if $\mathbf{T}(\mathcal{B}_k) = 3$ or $\mathbf{A}(\mathcal{B}_k) = 0$ bisect $\mathcal{B}_k$ and put the new boxes at the end of the list S whose number of elements n is updated, $k = k + 1$ and reiterate

Basically this algorithm just consider each box of the list sequentially. Boxes that are outside the workspace or satisfy the accuracy constraint are discarded.

During the iteration we will encounter boxes which are only partially inside W, or fully inside W but for which we cannot ascertain that all poses in the box satisfy the accuracy constraint. In that case we just bisect the box (the choice of the bisected range has a large influence on the computation time but this issue will not be addressed in this paper) and the resulting boxes are added to the list.

Finally the algorithm stops either if a box that violate the accuracy constraint is detected or when all the elements of S have been considered, in which case we can state that the accuracy constraint is satisfied over W.

Note that we have exposed the verification scheme for the workspace case. If it has to be used for a motion variety with a dimension m lower than the number of d.o.f. of the robot. we may safely assume that the pose parameters may be described as explicit function of m parameters. For example a trajectory (variety of dimension 1) will be defined by a single parameter (the time T) that we may assume to lie in the range [0,1].

The element of $\mathbf{J}^{-1}$may then be calculated as function of the m motion parameters and the boxes described in the algorithm will be m-dimensional boxes.

Note that the algorithm may be easily extended not to verify the accuracy constraint but to determine what is the maximal positioning errors up to a pre-defined accuracy α_i. For that purpose we will compute the positioning errors at the center of each box. This will allow to update a current value M_i for the maximum. All boxes

for which the **A** algorithm provides ranges $[\underline{d_i}, \overline{d_i}]$ that are all included in the range $[-M_i - \alpha_i, M_i + \alpha_i]$ will be discarded.

We may also extend further the algorithm by including the design parameters as unknowns. A two step algorithm may be used to determine ranges for these parameters so that the accuracy at some given poses is better than fixed threshold (for further details see [12]).

3 Implementation

Clearly the key point of the above accuracy verification scheme is the algorithm **A**. A possible approach for designing this algorithm is to consider the problem as a constrained optimization problem which consists in finding the maximum and minimum of the positioning errors. But with this approach there is no guarantee that the global optimum will be determined and this may cause a wrong result for the verification problem.

A better approach is to use interval analysis [6, 13]. For a given box interval analysis provides ranges for each element of the matrix $\mathbf{J}^{-1}$ that are guaranteed to includes the minimum and maximum value of the element. Thus we end up with a *linear interval system* of the form

$$\mathbf{YZ} = \mathbf{B}$$

where $\mathbf{Y}$ is an interval matrix and in our case $\mathbf{B}$ is a vector with scalar values. Finding a box enclosure of such system (i.e. finding a box that includes all solutions) is a classical problem in interval analysis [8, 14], whose complexity may be high [9]. The algorithms proposed for solving such systems are basically interval variant of classical linear system solvers (such as Gauss elimination).

3.1 The influence of uncertainties

Uncertainties are inherent part of a real system such as a robot. They occur at the modeling level: the geometry of the real robot differs from its theoretical model due to the manufacturing tolerances (for example for the Gough platform the locations of the anchor points of the legs on the base and platform are known only up to a known accuracy). Uncertainties are also due to control: there will be a deviation of the robot motion from the theoretical trajectory. An important point is that all these uncertainties are bounded.

An ideal accuracy verification scheme should be able to determine if the accuracy constraint is satisfied or violated in spite of these uncertainties. To deal with these uncertainties we may either just add them as intervals in the elements of $\mathbf{J}^{-1}$or we may add the uncertainties as additional unknowns, thereby increasing the dimension of the boxes. The best strategy depends on the size of the uncertainty ranges. For small one we may try to use the minimal set of unknowns and examine the behavior of the algorithm. The worst case occurs when it start bisecting again and again the same box, which indicates that the positioning errors may be very sensitive to these uncertainties: in that case it will be necessary to add them as unknowns.

4 Various improvement methods

The classical method for solving linear interval systems do not take well into account that our matrix $\mathbf{Y}$ is *parametric*, i.e. that the values of its elements are not independent. Hence it may be interesting to improve our algorithm by using methods that use in a better way the structure of $\mathbf{Y}$.

4.1 Pre-conditioning

A classical approach in interval analysis for solving linear interval systems is to pre-condition the matrix by multiplying it by a real matrix $\mathbf{K}$, usually the inverse of the *mid-matrix*, i.e. the matrix whose components are the mid-point of each range of the components of $\mathbf{Y}$. Indeed the system

$$\mathbf{KYZ} = \mathbf{KB}$$

has the same solution than the system (2). The purpose of this strategy is to get $\mathbf{S} = \mathbf{KJ^{-1}}$ close to the identity matrix. In term of solutions the domain may be indeed sharper than the one obtained with a direct solving of (2), but is still not satisfactory. Indeed numerical conditioning does not take into account the dependency between the elements of $\mathbf{J}^{-1}$.

We propose another method which consists first to compute *symbolically* the matrix $\mathbf{S}$, using symbolic k_{ij} as components of $\mathbf{K}$ and then plugging in the numerical values. For example the first column of $\mathbf{J}^{-1}$ for a Gough platform may be written as $x + F_i$, where x represent the location of the center of the platform along the x reference axis. Hence the first element S_{11} of the matrix product $\mathbf{S}$ will be numerically calculated as $\sum_{j=1}^{j=6} xK_{1j} + K_{1j}F_j$ where K_{1l} is the l-th element

of the first row of matrix $\mathbf{K}$. Using symbolic calculation S_{11} may be written as $x\sum_{j=1}^{j=6} K_{1j} + \sum_{j=1}^{j=6} K_{1j}F_j$. The number of occurrence of x is hence reduced to one and this will lead to a sharper interval evaluation of the element.

Note that both a left and right pre-conditioning matrix may be used.

4.2 Improvement of the Gaussian elimination scheme

It must be noted that the interval evaluation of the elements of $\mathbf{J}^{-1}$ may be improved by using the derivatives of the elements with respect to the unknowns. Indeed we may calculate the interval evaluation of these derivatives and if the lower bound is positive or the upper bound is negative, then the element is monotonic with respect to this unknown. Hence to compute the lower and upper bound of the interval evaluation of the element we will fix this unknown to its lower or upper bound. This procedure should be used recursively as the interval evaluation of a derivative that was not of a constant sign initially may change if one of the variable is fixed to its extremal value.

A similar procedure may be used for improving the Gaussian elimination scheme. We first compute an interval evaluation $\mathbf{Y}^{(0)}$ of $\mathbf{Y}$. The Gauss elimination scheme may be written as [14]

$$Y_{ik}^{(j)} = Y_{ik}^{(j-1)} - Y_{ij}^{(j-1)}Y_{jk}^{(j-1)}/Y_{jj}^{(j-1)} \quad \forall\, i \text{ with } j > k \tag{3}$$

$$b_i^{(j)} = b_i^{(j-1)} - Y_{ij}^{(j-1)}b_j^{(j-1)}/Y_{jj}^{(j-1)} \tag{4}$$

The enclosure of the variable Z_j can then be obtained from $Z_{j+1}, \ldots, Z_n$ by

$$Z_j = (b_j^{(j-1)} - \sum_{k>j} Y_{jk}^{(j-1)}Z_k)/Y_{jj}^{(j-1)} \tag{5}$$

Note that the elements at iteration j can be computed only if the interval $Y_{jj}^{(j-1)}$ does not include 0. To improve the efficiency of the procedure we note that at iteration j an interval evaluation of the derivatives of Y_{ik}, b_i with respect to $\mathbf{X}$ may be deduced for the derivatives of the elements computed at iteration $j-1$ by using the chain rule. As we have the derivatives of the elements at iteration 0 we may then deduce the derivatives of the elements at any iteration and use these derivatives to improve the interval evaluation of these elements.

5 Validation and examples

As may be seen with the above theoretical sections the proposed accuracy constraint check is relatively complex and it is difficult to perform a theoretical complexity analysis. Hence numerical validation is necessary to determination which combination of regularity checks are efficient.

For the examples we will consider one of the most difficult case: the Gough platform (figure 1). We define a reference frame $(O, \mathbf{x}, \mathbf{y}, \mathbf{z})$. The attachment points

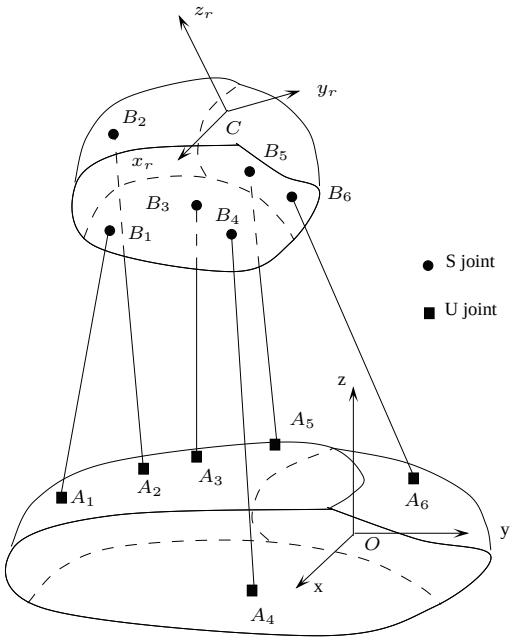

Figure 1: The Gough platform

of the leg i on the base will be denoted by A_i. The attachment points on the platform will be denoted by B_i and it is well known that the coordinates of B_i in the reference frame can be obtained as function of the pose parameters. The inverse jacobian matrix is then constituted of the normalized Plücker vectors of the line associated to

each leg:

$$\mathbf{J}^{-1} = ((\frac{\mathbf{A_i B_i}}{||\mathbf{A_i B_i}||} \quad \frac{\mathbf{OA_i} \times \mathbf{OB_i}}{||\mathbf{A_i B_i}||})) \tag{6}$$

The uncertainties in the geometry lie in the location of the anchor point A_i, B_i.

The interval analysis part is implemented using the `BIAS/Profil` package[1] and our interval analysis library `ALIAS`[2] that offer high-level modules that are combined for implementing the calculation of the linear system solver. Furthermore this library allows one to use a distributed implementation: a master manages the list of boxes and send the box on top of the list to a slave computer that performs a few iterations of the algorithm and send the result back to the master.

A 6D workspace $\mathcal{W}$ is defined with the ranges x, y in [-15,15], z in [45,50] and the three Euler angles having the ranges [-15,15] degree. The computation time on a Dell D400 laptop (1.7 Ghz) is established as follows:

- 6D workspace without uncertainty: for $\mathcal{W}$ it is established that the accuracy constraint is satisfied in a computation time of about 10 mn. If the orientation ranges of $\mathcal{W}$ is extended to [-40,40] degree poses that violates the accuracy constraint is detected in less than 10s.
- 6D workspace with uncertainties: for a $\pm$ 0.05 uncertainty on each coordinates of the A_i, B_i points the constraint accuracy in $\mathcal{W}$ in about 23mn. For an uncertainty of $\pm$ 0.1 the computation time establishes respectively at 42mn.

6 Conclusion

We have proposed an accuracy validation scheme that may be used to determine if a given robot (possibly with an uncertain modeling) is such that the maximal positioning errors over a given workspace (that may also include uncertainties) are lower than given thresholds. This algorithm may easily be extended to determine what is the maximal positioning errors over the given workspace and even to find ranges for design parameters so that the accuracy at given poses is better than fixed thresholds.

We have shown that the extremal positioning errors is a better index than the usual condition number to qualify a robot in terms of accuracy. However it may not be sufficient to compare two possible design: mean value and variance will also be needed. We intend to address these issues in the future.

[1] http://www.ti3.tu-harburg.de/Software/PROFILEnglisch.html
[2] www.inria-sop.fr/coprin/logiciel/ALIAS/ALIAS.html

References

[1] Bhattacharya S., Hatwal H., and Ghosh A. On the optimum design of a Stewart platform type parallel manipulators. *Robotica*, 13(2):133–140, March - April , 1995.

[2] Chablat D., Wenger P., and Merlet J-P. A comparative study between two three-dof parallel kinematic machines using kinetostatic criteria and interval analysis. In *11th World Congress on Theory of Machines and Mechanisms*, pages 1209–1213, Tianjin, April, 1-4, 2004.

[3] Clavel R. and others . High precision parallel robots for micro-factory applications. In *2nd Int. Colloquium, Collaborative Research Centre 562*, pages 285–296, Braunschweig, May, 10-11, 2005.

[4] Fattah A. and Hasan Ghasemi A.M. Isotropic design of spatial parallel manipulators. *Int. J. of Robotics Research*, 21(9):811–824, September 2002.

[5] Han C. and others . Kinematic sensitivity analysis of the 3-UPU parallel manipulator. *Mechanism and Machine Theory*, 37(8):787–798, August 2002.

[6] Jaulin L., Kieffer M., Didrit O., and Walter E. *Applied Interval Analysis*. Springer-Verlag, 2001.

[7] Jelenkovic L. and Budin L. Error analysis of a Stewart platform based manipulators. In *Int. Conf. on Intelligent Engineering Systems (INES)*, Opatija, May, 26-28, 2002.

[8] Kreinovich V. Optimal finite characterization of linear problems with inexact data. Research Report CS-00-37, University of Texas at El Paso, 2000.

[9] Kreinovich V., Lakeyev A., Rohn J., and Kahl P. *Computational complexity and feasibility of data processing and interval computations*. Kluwer, 1998.

[10] Masory O., Wang J., and Zhuang H. Kinematic modeling and calibration of a Stewart platform. *Advanced Robotics*, 11(5):519–539, 1997.

[11] Merlet J-P. Jacobian, manipulability,condition number, and accuracy of parallel robots. *ASME J. of Mechanical Design*, 128(1):199–206, January 2006.

[12] Merlet J-P. and Daney D. Dimensional synthesis of parallel robots with a guaranteed given accuracy over a specific workspace. In *IEEE Int. Conf. on Robotics and Automation*, Barcelona, April, 19-22, 2005.

[13] Moore R.E. *Methods and Applications of Interval Analysis*. SIAM Studies in Applied Mathematics, 1979.

[14] Neumaier A. *Interval methods for systems of equations*. Cambridge University Press, 1990.

[15] Niaritsiry F-T., Fazenda N., and Clavel R. Study of the source of inaccuracy of a 3 dof flexure hinge-based parallel manipulator. In *IEEE Int. Conf. on Robotics and Automation*, pages 4091–4096, New Orleans, April, 28-30, 2004.

[16] Parenti-Castelli V. and Di Gregorio R. Influence of manufacturing errors on the kinematic performance of the 3-UPU parallel mechanism. In *2nd Chemnitzer Parallelkinematik Seminar*, pages 85–99, Chemnitz, April, 12-13, 2000.

[17] Patel A.J. and Ehmann K.F. Volumetric error analysis of a Stewart platform based machine tool. *Annals of the CIRP*, 46/1/1997:287–290, 1997.

[18] Pott A. and Hiller M. A new approach to error analysis in parallel kinematic structures. In *ARK*, Sestri-Levante, June 28- July 1, 2004.

[19] Pritschow G., Eppler C., and Garber T. Influence of the dynamic stiffness on the accuracy of PKM. In *3rd Chemnitzer Parallelkinematik Seminar*, pages 313–333, Chemnitz, April, 23-25, 2002.

[20] Ryu J. and Cha J. Volumetric error analysis and architecture optimization for accuracy of HexaSlide type parallel manipulators. *Mechanism and Machine Theory*, 38(3):227–240, March 2003.

[21] Sellgren U. Modeling of mechanical interfaces in a systems context. In *Int. ANSYS Conf.*, Pittsburgh, April 2002.

[22] Tischler C.R. and Samuel A.E. Predicting the slop of in-series/parallel manipulators caused by joint clearances. In *ARK*, pages 227–236, Strobl, June 29-July 4, 1998.

[23] Tönshoff K. and others . Modelling of error effects on the new hybrid kinematic DUMBO structure. In *3rd Chemnitzer Parallelkinematik Seminar*, pages 639–653, Chemnitz, April, 23-25, 2002.

[24] Wang S.M. and Ehmann K.F. Error model and accuracy analysis of a six-dof Stewart platform. *ASME Journal of Manufacturing Science and Engineering*, 124(2):286–295, May 2002.

[25] Wolf A. and Shoham M. Investigation of parallel manipulators using linear complex approximation. *ASME J. of Mechanical Design*, 125(3):564–572, September 2003.