Guaranteed in-the-workspace improved trajectory/suface/volume verification for parallel robots

J-P. Merlet
INRIA Sophia Antipolis, France
E-mail: Jean-Pierre.Merlet@sophia.inria.fr

Abstract— **The workspace of a parallel robot has a complex shape and is difficult to model. Hence verifying if an arbitrary trajectory lie entirely within this workspace is a complex issue. We present an algorithm that allow such verification for any trajectory such that the pose parameters are arbitrary analytical time function. Furthermore this algorithm allows one to deal with uncertainties in the followed trajectory and in the geometrical description of the robot**

Keywords— **workspace, trajectory verification, parallel robots, interval analysis**

I. Introduction

Workspace of parallel robots are usually relatively small in size and are characterized by a complex shape [1], [2], [3], [15]. Although possible [10] it is difficult to compute a full model of the workspace and hence it is difficult to verify if a given trajectory lie completely within the workspace of the robot although this is an essential step for trajectory planning. This important problem has been addressed in few papers [4], [13], [14] and mostly in view of singularity avoidance.

Let $\mathcal{X} = \{X_1, X_2, \ldots, X_l\}$ be the set of parameters that are used to describe a pose of the platform. We will assume in this paper that a trajectory $\mathcal{T}$ is defined by a set of arbitrary analytical time functions f_i such that

$$X_i = f_i(T) \qquad (1)$$

with the time T restricted to lie in the range [0,1]. For example for a 2-dof translational robot in the plane we will have $\mathcal{X} = \{x, y\}$ and a circular trajectory centered at (0,0) with radius 10 may be defined as:

$$\begin{aligned} x &= 10\sin(2\pi T) \\ y &= 10\cos(2\pi T) \end{aligned}$$

We will also assume that the workspace of the robot is defined as the set of poses $\mathcal{X}$ such that a set of m inequalities constraints:

$$F_i(\mathcal{X}) \le 0 \quad i \in [1, m] \qquad (2)$$

are satisfied. To give an example let us assume that we are dealing with a Gough platform having the lengths ρ of its extensible legs restricted to lie in the range $[\rho_{min}, \rho_{max}]$. Let $\mathcal{K}$ be the inverse kinematics equation of the robot that provides the square of leg length as function of the pose:

$$\rho^2 = \mathcal{K}(\mathcal{X})$$

The F inequality constraints defined by (2) will be the 12 inequalities defined by:

$$\mathcal{K}(\mathcal{X}) - \rho_{max}^2 \le \qquad \rho_{min}^2 - \mathcal{K}(\mathcal{X}) \le 0$$

for the 6 legs of the robot. Additional constraints limiting the workspace may also be considered. For example assume that the anchor point of the legs on the base (platform) are denoted A_i (B_i). The ball-and-socket joint at A_i may have a limited rotation capability i.e. the angle between the leg and the vertical direction may not exceed a given value δ. This can be written as:

$$\cos(\delta) \le \frac{\mathbf{A_iB_i}_z}{||\mathbf{A_iB_i}||}$$

where $\mathbf{A_iB_i}_z$ is the z-component of the vector $\mathbf{A_iB_i}$. As the vector $\mathbf{A_iB_i}$ may be expressed as function of $\mathcal{X}$ the above relation will provide an additional constraint F_i.

Singularity analysis can be added as well in this framework. Assume that at the starting point of the trajectory we have determined that the sign of the determinant of the inverse Jacobian matrix of the robot is negative. Then no singularity will occur on the trajectory if the sign of the determinant is negative for any T in [0,1]. Hence the determinant can be given as one of the F_i's.

Plugging in equation (1) into the inequalities (2) allows one to express the inequalities into function dependent only on the time T. The purpose of this paper is to propose an algorithm will verify if the trajectory $\mathcal{T}$ is fully enclosed in the workspace of the robot i.e. that for all T in [0,1] and i in [1,m] we have $F_i(\mathcal{X}(\mathcal{T})_\mathcal{T}) \le 0$. This algorithm will use interval arithmetics that is briefly explained in the next section

II. Interval arithmetics

Interval arithmetics is a well known method for computing bounds of a function, being given bounds on the unknowns appearing in this function [7], [12]. In other words if f is a function of some unknowns $a_1, a_2, \dots, a_k$ and if these unknowns are restricted to lie in the intervals $A_1, A_2, \dots, A_k$, then the interval arithmetics evaluation of $f(A_1, \dots, A_k)$ will be the interval $[\underline{f}, \overline{f}]$ such that if all a_i have a value in their range A_i, then $\underline{f} \leq f(a_1, \dots, a_k) \leq \overline{f}$. An appropriate implementation of interval arithmetics allows one even to deal with numerical round-off errors.

Such interval evaluation may be obtained simply by using the *natural evaluation*: in this process all the operators appearing in a function are substituted by their interval equivalent. For example consider the function $f(x) = x - \sin(x)$ for x in the range [-0.1,1]. We have

$$\begin{aligned} f([-0.1, 1]) &= [-0.1, 1] - sin([-0.1, 1] \\ &= [-0.1, 1] - [-0.099834, 0.841471] \\ &= [-0.941471, 1.099834] \end{aligned}$$

The value of $\underline{f}, \overline{f}$ may be overestimated (as in the previous example, but the amplitude of this overestimation will decrease with the width of the ranges. Note that an interval evaluation is sensitive to the analytical form of the equation. For example the expressions

$$\begin{gathered} x + x\sin(y) \\ x(1 + \sin(y)) \end{gathered}$$

although mathematically equivalent, will have in general a different interval evaluation (the later one having only one occurrence of the unknowns will lead to an exact evaluation of the lower and upper bound of the expression).

Assume that T is constrained to lie in a given interval $[a, b]$ included in [0,1] and let $[\underline{f_i}, \overline{f_i}]$ be the interval evaluation of $F_i([a, b])$. The following results hold:
1. if $\overline{f_i} \leq 0$, then $F_i \leq 0$ for any value of T in $[a, b]$
2. if $\underline{f_i} > 0$, then $F_i > 0$ for any value of T in $[a, b]$

III. Algorithm principle

The interval evaluation of a quantity Q will be denoted $\mathcal{B}(Q)$, the lower bound of this interval evaluation $\underline{\mathcal{B}(Q)}$ and its upper bound $\overline{\mathcal{B}(Q)}$. We will also use a *bisection process* for the range $\hat{T} = [T_1, T_2]$: the result of the bisection process applied on this range is the 2 new ranges $[T_1, (T_1 + T_2)/2], [(T_1 + T_2)/2, T_2]$.

We will use a list $\mathcal{S}$ of ranges for T with n ranges. This list will be initialized with the range $S_1 = [0, 1]$ and hence $n = 1$. During the algorithm we will consider the i-th element $\mathcal{S}_i$ of the list $\mathcal{S}$. We start with $i = 1$ and the algorithm proceeds along the following steps:

1. if $i > n$ return `VALID TRAJECTORY`
2. compute $\mathcal{B}(F_j(\mathcal{S}_i))$ for all j in [1,m]:
 (a) if there exists an j such that $\underline{\mathcal{B}(F_j(\mathcal{S}_i))} > 0$ then return `INVALID TRAJECTORY`
 (b) if for all j in [1,m] we have $\overline{\mathcal{B}(F_j(\mathcal{S}_i))} \leq 0$, then $i = i + 1$ and go to step 1
 (c) otherwise bisect $\mathcal{S}_i$ and add the resulting ranges at the end of $\mathcal{S}$. Then $i = i + 1$, $n = n + 2$ and go to step 1

Consider what will happen with the range $\mathcal{S}_1$. At step 2 we will compute the interval evaluation of the 6 leg lengths. At step 2(a) we have found that one of the inequality will never be satisfied for the range S_i i.e. the trajectory is not valid. At step 2(b) we find that the trajectory for the time range S_i satisfies all the constraints and hence this part of this trajectory is valid. At step 2(c) we find that at least one of the constraints has a positive upper bound while the other one are satisfied. But a positive upper bound does not mean that the constraints is not satisfied as the interval evaluation may be overestimated. Thus we will bisect the range $\mathcal{S}_1 = [0, 1]$ and start again with the range $\mathcal{S}_2 = [0, 0.5]$ and $\mathcal{S}_3 = [0.5, 1]$.

The purpose of the above algorithm is just to return a yes/no answer about the validity of the trajectory. If the trajectory is not valid we will also get one range for T on which the trajectory. A variant will allow one to get all the time ranges on which the trajectory is outside the workspace. For that purpose we modify step 2(a): a flag will be used to memorize that the trajectory is not valid and a list will be used to store all the time ranges for which the trajectory is outside the workspace.

Note also that the presented algorithm is only the most basic one: numerous tricks issued from interval analysis and constraint programming can be used to improve the efficiency. For example the derivative of the constraints may be used to improve the interval evaluation of the F_i, thereby reducing the number of bisection that will be used in the algorithm. Indeed let G_i be the time derivative of F_i and let $[\underline{G}, \overline{G}]$ be the interval evaluation of G for a given time range $[a, b]$. If $\underline{G} > 0$ or $\overline{G} < 0$, then G_i is monotonic with respect to T. Thereby the exact interval evaluation of $F_i([a, b])$ is $[F_i(a), F_i(b)]$ if $\underline{G} > 0$ or $[F_i(b), F_i(a)]$ if $\overline{G} < 0$. If the upper bound of this evaluation is positive, then part of the trajectory is outside the workspace.

The presented algorithm is numerically safe: if a trajectory is found to be valid then we can guarantee that it is indeed fully enclosed in the workspace. On the other hand it may happen that due to numerical roundoff errors we do not get $\underline{\mathcal{B}(F_j(\mathcal{S}_i))} > 0$ or

$\overline{\mathcal{B}(F_j(\mathcal{S}_i))} \leq 0$ even for a time range whose width is very small: in that case will state that the trajectory is not valid but we will also return a specific signal to indicate that this result is not guaranteed.

The algorithm may also be extended to deal with surface and volume defined by parametric analytical equations [11]. In that case instead of having to deal with only one parameter in the bisection process we will have 2 or 3 parameters.

IV. Implementation

The algorithm is implemented basically in C++ using the interval arithmetics package `Bias/Profil` [1]. and high level interval analysis modules of our `ALIAS` library[2]. We may have implemented the constraints also in C++ but this will have implied to compile the trajectory verifier for each new trajectory. To allow for more flexibility we have used a parser that is provided with the `ALIAS` library.

A. The parser

Our parser is a C++ program that takes as input:

- the name of a file, called the *formula file*, that contains the analytical form of the function(s) that have to be evaluated in Maple format
- the ranges and name for each unknowns appearing in the function(s)

and it will return the interval evaluation of the function(s). An example of a formula file is:

```
eq=(T^2-1.23)*T+(2.34*sin(2*Pi*T)-2)*T
eq=2+(-10*T+log(T+10))*sin(2*Pi*T)
```

each line representing a constraint F_i. The parser is also able to recognize ranges which are written using Maple syntax (e.g. `INTERVAL(0.1..0.2)` for the range [0.1,0.2]), constant parameters whose values will be read in a parameter file, and intermediate variables that will allow to decrease the computation time of the evaluation (in the example we may assign `sin(2*Pi*T)` to an intermediate variable so that to avoid evaluating twice the same term).

Creating a formula file may be done by using a text editor. Alternatively it is possible to use a specific Maple package that produces automatically the formula file, but also performs some heuristic simplifications on the function in order to improve both the evaluation and its computation time. This package allows one to deal with any type of robot (including non parallel one) and any type of constraints that influence the workspace.

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

B. The kernel

In the algorithm only the interval evaluation of the constraints is trajectory dependent. With the parser this part is defined by the external formula file and hence the core of the algorithm is not trajectory or robot dependent. We have designed a C++ kernel that implements the algorithm that may be used independently of the robot and of the trajectory specifications.

C. Examples

Consider the *parallel robot*, called a Gough platform [6], described in figure 1. In this robot a base

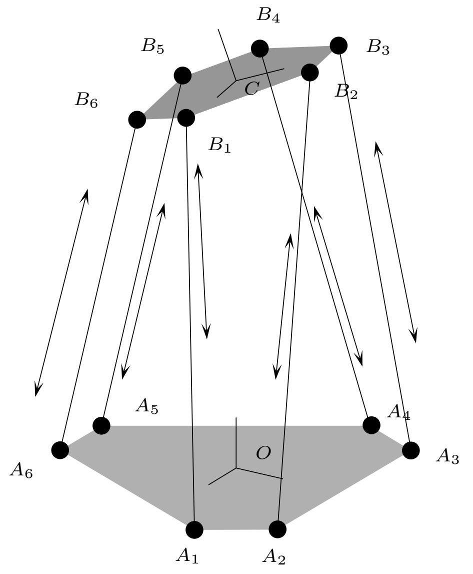

Fig. 1. The classical Gough-type parallel robot

and a platform are connected through 6 legs which have a ball-and-socket joint at each extremity A_i, B_i. Linear actuators enable one to change the leg lengths, which in turn enables one to control the position and orientation of the platform.

We define a reference frame $O, (\mathbf{x}, \mathbf{y}, \mathbf{z})$ which is attached to the base and a mobile frame $C, (\mathbf{x_r}, \mathbf{y_r}, \mathbf{z_r})$ which is attached to the platform. The parameters of of a pose are the coordinates of point C in the reference frame, together with the Euler's angles ψ, θ, ϕ that allow to define the orientation of the platform. For physical reasons the leg lengths ρ of the robot are constrained to lie in a given range:

$$\rho_{min} \leq \rho_i \leq \rho_{max} \quad \forall i \in [1, 6] \tag{3}$$

For a Gough platform the length of a leg is simply the norm of the vector $\mathbf{AB}$ which may be written as

$$\mathbf{AB} = \mathbf{AO} + \mathbf{OC} + \mathbf{CB} \tag{4}$$

and the square of the leg length ρ is given by:

$$\rho^2 = ||\mathbf{AB}||^2 \tag{5}$$

For a given trajectory we may obtain with Maple an analytical expression of $\rho_i(T)$. Consider for example the planar trajectory shown in figure 2. Using the

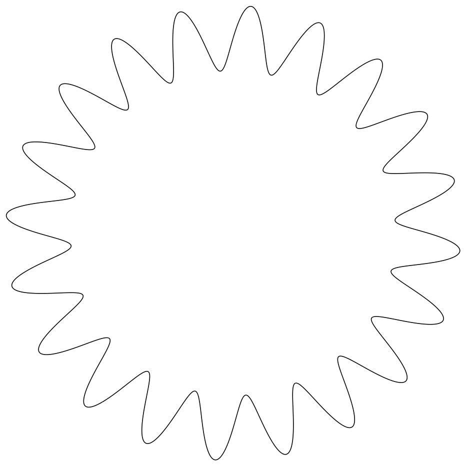

Fig. 2. An example of complex trajectory: the gear trajectory

convention x_C =x, y_C =y, z_C =z, ψ =p, θ =t, ϕ =h the trajectory may be defined as:

```
p:=0:t:=0:h:=0:
x:=(3+0.5*sin(40*Pi*T))*sin(2*Pi*T):
y:=(3+0.5*sin(40*Pi*T))*cos(2*Pi*T):
z:=56:
```

Then the Maple procedure will produce the formula file corresponding to the 12 F_i For a given robot the square of length of the first leg is:

$$\begin{aligned}\rho_1^2 \quad = \quad & \frac{12741}{4} + 3\,\sin(40\,\pi\,T) + 36\,\sin(2\,\pi\,T) - \\ & 1/4\,\left(\cos(40\,\pi\,T)\right)^2 + 6\,\sin(2\,\pi\,T)\sin(40\,\pi\,T) \\ & -12\,\cos(2\,\pi\,T) - 2\,\cos(2\,\pi\,T)\sin(40\,\pi\,T)\end{aligned}$$

On a SUN Blade the verifier is able to determine that the trajectory is invalid (figure 3) in a computation time of less than 1 second. Another example is a trajectory that simulate the manufacturing of a conic lens [8], figure 4, which may be described by the equations:

```
x:=10*T*sin(20*Pi*T):
y:=10*T*cos(20*Pi*T):
z:=58-3*T:
```

With the constant orientation defined by $\psi = \theta = \phi = 0$ the trajectory is inside the workspace of the robot. Indeed as shown in figure 5 all the 6 leg lengths lie within the allowed limit [55,60]. However we may wish

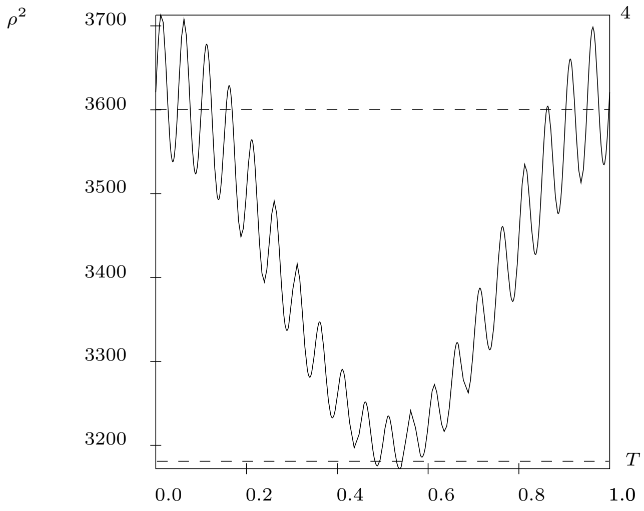

Fig. 3. The square of one the leg lengths for the gear trajectory. The dashed lines represent the minimal and maximal value of this leg length. It may be seen that the trajectory is invalid and the motion verifier indicates that the trajectory is invalid at least in the time range [0,0.015625]

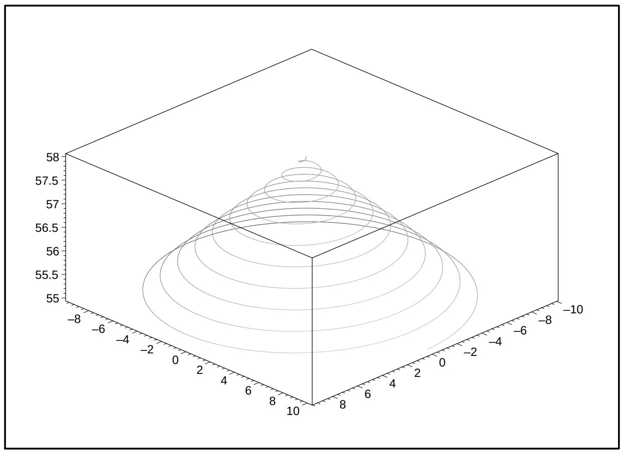

Fig. 4. A trajectory for the manufacturing of a lens

to have the normal of the platform roughly normal to the lens. This may be obtained by setting the rotation angle as follows:

```
p:=21*Pi*T:h:=-p: t:=10*Pi*T/180:
```

In that case the algorithm determines that the trajectory is outside the workspace at least for T in [0.75,0.753]. Indeed the leg length of link 5 is at this time lower than its minimal limit 55 (figure 6). Orientation motion may be checked as well. Assume for example that we are using a Gough platform for tracking celestial object with a telescope. The telescope should be able to modify its orientation in order to explore some sector in space using a spiral motion

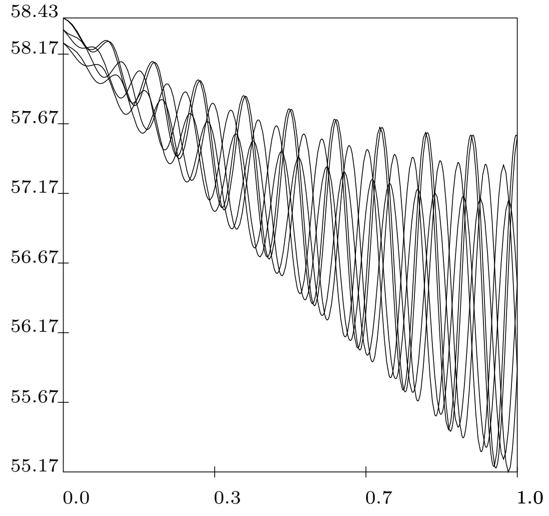

Fig. 5. On the trajectory all the leg lengths are within their limit [55,60]

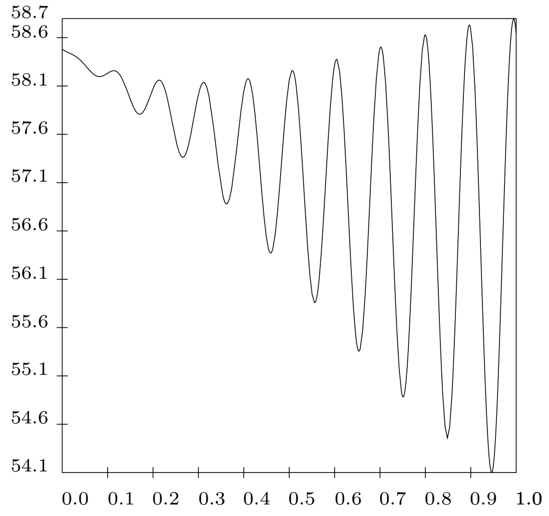

Fig. 6. A leg length is lower than its lower limit 55

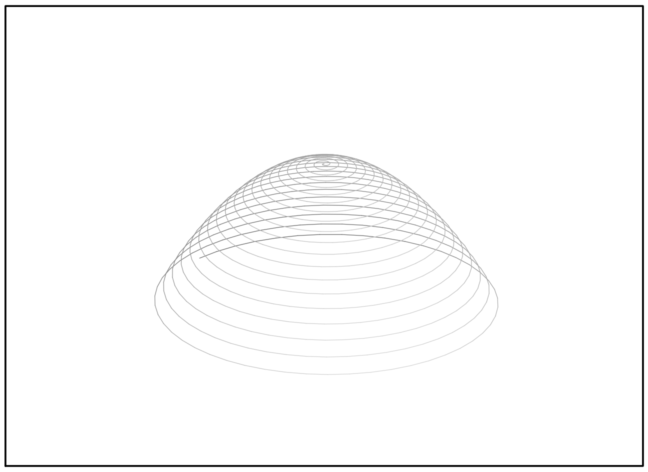

Fig. 7. Trajectory of the tip of a telescope when exploring a spatial sector

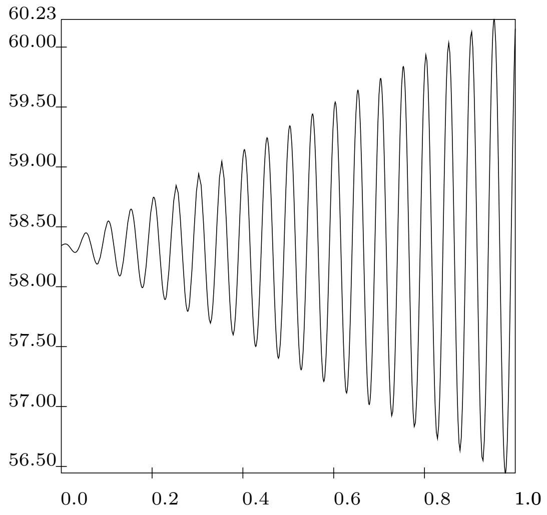

Fig. 8. For the telescope motion a leg length exceed its upper limit 60 toward the end of the trajectory

while the center of the telescope is not moving [5]. Such motion may be described by:

```
p:=40*Pi*T:
h:=-p:
t:=10*Pi*T/180:
```

while the location of C remains invariant. Figure 7 shows the motion of the tip of the telescope. The motion verifier determine that the trajectory is valid in about 1 second. But if we increase the tilt of the platform from 10 degrees to 15 degrees then the trajectory is no more valid as the actuators exceed the upper limit (figure 8). By increasing incrementally the maximal tilt angle in the formula file we are able to determine that the tilt angle should not exceed 12.5 degrees.

V. Uncertainties

Up to now we have assumed that the robot will follow exactly the specified trajectory. But in practice control errors and errors in the geometrical model of the robot may cause the real trajectory to be a little bit different from the specified one. We still are able to deal with these uncertainties.

A. First approach

In this approach we will assume that the errors are small and that each of them may be attributed a range. Theses ranges will appear then in the expression of the F_i's. But remember that the parser is able to deal with expression involving interval coefficients and hence we may still calculate an interval evaluation of the F_i's. For example in the lens case with a constant orientation a trajectory error of $\pm$ 0.05 will still keep the trajectory within the workspace.

But in some case we may end up with a deadlock. Indeed let us assume that one of the F_i is written as:

$$F_i = -0.999(T-1) + [-0.01, 0.05]$$

the intervals being due to the uncertainties. At time $T = 1$ we will not be able to determine if the F_i is indeed negative.

B. Second approach

In the second approach each parameter that may have an error is considered as a new variable. Beside the variable T we will have now to consider also these new variables $\{P_1, \ldots, P_k\}$. A *box* will be defined as a set of ranges for $T = P_0$ and the variables P_i. The initial box is known and the bisection process will be applied on the boxes. In that case we need a method to determine which variable in the box will be bisected. There are various possibilities:

• always bisect the variable having the largest range. This method is simple but has the drawback that if the constraints are not very sensitive with respect to some unknowns we will still bisect them, thereby inducing unnecessary calculation

• if the constraints are differentiable use the *smear function* [9]. Let G be the Jacobian matrix of the constraint equation and $\hat{G}$ be its interval evaluation for a given box. The smear function for a variable P_l is defined as $\text{Max}(|\hat{G}_{jl}|w(P_l))$ for all j in $[1,m]$, where $w(P_l)$ is the width of the range for P_l. It can be seen as the influence of P_l on the variation of the constraint equation, weighted by the width of the range of the variable. The bisected variable will be the one having the largest smear function

The output of the algorithm will be similar:

• either a box such that at least one constraint will never be fulfilled. This will mean that whatever are the errors in the box ranges and whatever is the time in the time range the trajectory will be outside the workspace

• or all the boxes for each at least one one constraint will never be fulfilled. This output contains a lot of information: for example it will be possible to check if a parameter in the geometry of the robot has to have a smaller clearance

C. Dealing with pose-dependent errors

Note that in this approach we are able to take into account errors that are pose-dependent, *as soon as an analytical form of these errors can be calculated.*

Unfortunately in some cases such expression cannot be calculated or are too complex to be used. For example for a Gough platform the influence of the leg lengths measurements errors on the positioning of the platform is pose dependent but is extremely difficult to calculate as it implies the use of the Jacobian matrix J while only J^{-1} has a simple analytical expression. This problem will be addressed in a future paper.

VI. Conclusion

The proposed algorithm allows to check very quickly the validity of almost any trajectory (or of a suface/volume) in a guaranteed manner. It is also able to deal with uncertainties both on the control and on the robot's geometry parameters. It may also provides information on the part of the trajectory that is outside the workspace of the robot.

Prospective work will be to design a strategy for proposing modifications of the trajectory (or of the robot itself) so that a non valid trajectory will become valid.

References

[1] Adkins F.A. and Haug E.J. Operational envelope of a spatial Stewart platform. *ASME J. of Mechanical Design*, 119(2):330–332, June 1997.

[2] Agrawal S.K. Workspace boundaries of in-parallel manipulator systems. *Int. J. of Robotics and Automation*, 7(2):94–99, 1992.

[3] Bonev J., I.A.and Ryu. Workspace analysis of 6-PRRS paralllel manipulators based on the vertex space concept. In *ASME Design Engineering Technical Conference*, Las Vegas, September, 12-15, 1999.

[4] Dasgupta B. and Mruthyunjaya T.S. Singularity-free path planning for the Stewart platform manipulator. *Mechanism and Machine Theory*, 33(6):711–725, August 1998.

[5] E. McInroy J. and Hamann J.C. Design and control of flexure jointed hexapods parallel manipulator. *IEEE Trans. on Robotics and Automation*, 16(4):372–381, August 2000.

[6] Gough V.E. and Whitehall S.G. Universal tire test machine. In *Proceedings 9th Int. Technical Congress F.I.S.I.T.A.*, volume 117, pages 117–135, May 1962.

[7] Hansen E. *Global optimization using interval analysis.* Marcel Dekker, 1992.

[8] Hesselbach J., Plitea N., Frindt M., and Kusiek A. A new parallel mechanism to use for cutting convex glass panels. In *ARK*, pages 165–174, Strobl, June 29- July 4, 1998.

[9] Kearfott R.B. and Manuel N. III. INTBIS, a portable interval Newton/Bisection package. *ACM Trans. on Mathematical Software*, 16(2):152–157, June 1990.

[10] Merlet J-P. Determination of 6d workspaces of Gough-type parallel manipulator and comparison between different geometries. *Int. J. of Robotics Research*, 18(9):902–916, October 1999.

[11] Merlet J-P. A generic trajectory verifier for the motion planning of parallel robots. *ASME J. of Mechanical Design*, 123:510–515, December 2001.

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

[13] Nenchev D.N. and Uchiyama M. Singularity-consistent path planning and control of parallel robot motion through instantaneous-self-motion type. In *IEEE Int. Conf. on Robotics and Automation*, pages 1864–1870, Minneapolis, April, 24-26, 1996.

[14] Nguyen C.C. and others . Trajectory planning and control of a Stewart platform-based end-effector with passive compliance for part assembly. *J. of Intelligent and Robotics Systems*, 6(2-3):263–281, December 1992.

[15] Pernkopf F. *Workspace analysis of Stewart-Gough platforms.* Ph.D. Thesis, Baufakultät, University of Innsbruck, September, 11, 2003.