A Local Motion Planner for Closed-loop Robots

Jean-Pierre Merlet

A Local Motion Planner for Closed-loop Robots

J-P. Merlet

INRIA

BP 93, 06902 Sophia-Antipolis Cedex, France

Jean-Pierre.Merlet@sophia.inria.fr

Abstract— **Global motion planners have been proposed for closed-loop robot based on the same paradigm than has been proposed for serial chains. First a sparse representation of the configuration space of the robot is constructed as a set of nodes. This is somewhat more complicated than for serial chain as the closure equations of the mechanism should be satisfied. Then a motion planning query consists simply in connecting the start and goal points through an appropriate set of nodes (usually minimizing the length of the trajectory). But such motion planner should be complemented by a local motion planner that addresses the following issues:**

1) **ensure that two successive nodes belong to the same robot kinematic branch (otherwise connecting these nodes will require to disassemble the robot)**
2) **verify that all poses between nodes satisfy the robot constraints (if possible taking into account the uncertainties in the robot modeling)**
3) **eventually try to shorten the trajectory length**

We present such a local motion planner that addresses all three issues and illustrates its use on a Gough parallel robot.

I. INTRODUCTION

A. Global motion planner

Motion planning is a classical problem in robotics and has been largely addressed for serial chains [2], [9], [10]. Among the most successful method we may mention the *roadmap* approach: a representative, but limited, set of reachable poses, called the sampling tree, is pre-computed and a planning query consists in connecting the reachable poses so that, for example, the length of the trajectory is minimal. An interest of this approach is that the construction of the sampling tree may be done once off-line, while finding a trajectory relies on determining a shortest path within a graph, a task for which there are efficient and fast algorithms. For serial chains the sampling is performed usually in the joint space (a point in this space leads to a unique configuration of the robot). Unfortunately adapting this approach to closed-loop chains is not easy as the joint variables should satisfy the *closure equations* and cannot be arbitrary chosen.

B. Closed-loop robots and kinematic constraints

The planning problem for serial chains is usually related to avoiding obstacles and self-collision, while for closed-chains *kinematic constraints* become preponderant.

A specific kinematic constraint for closed-loop structure is that their operational configuration space may have different components that are not connected. This may be illustrated simply on a four-bar mechanism (figure 1).
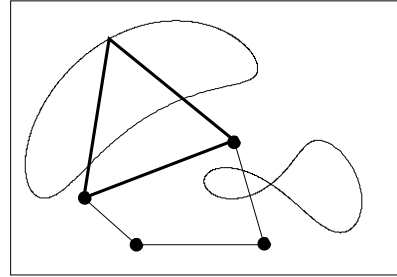

Fig. 1. A 4-bar mechanism with an operational space having two not connected components. Going from a pose in a component to another in the second component requires to disassemble the mechanism.

A consequence is that even if a sampling method is able to determine poses that satisfy the closure equations, they may still belong to different *kinematic branches* and therefore cannot be connected by a trajectory.

Other kinematic constraints are joint limits, collision avoidance between the robot's bodies and the absence of singularity on the trajectory. All these constraints are functions of the pose of the robot and of parameters that describe the geometry of the robot and must be satisfied although there are *uncertainties* on these parameters.

The complexity of the kinematic constraints and the uncertainties leads to complicated configuration space. In some cases this configuration space can be constructed either analytically [17] or numerically [12] but manipulating these representations for motion planning is a complicated task. One of the most successful motion planning method for closed-chains is called the *Probabilistic Roadmap* [4], [18], [19]. Here the sampling poses are chosen at random although general geometric algorithms are used to guide the sampling so that the poses satisfy the closure equations.

C. Local planner

In all cases these global motion planners must be complemented by a *local motion planner* that addresses the following issues:

1) ensure that two successive nodes in the trajectory belong to the same *kinematic branch* of the robot.
2) find a path that connect two successive nodes of the trajectory, such that any pose on the path are guaranteed to satisfy the kinematic constraints
3) eventually improve the trajectory in terms of some optimality criteria (e.g. its length) and provide a solution that is "close" to the global optimum of the criteria (we will precise this closeness issue later on)

II. BASIC INGREDIENTS

The local planner we have developed relies on the following assumptions:

- being given a pose of the end-effector we are able to state if the kinematic constraints are satisfied
- all the kinematic constraint may be written as $\mathcal{C}(\mathbf{X}, \mathbf{\Theta}) \leq 0$, where $\mathbf{X}$ is a n-dimensional vector constituted of the parameters that describes the end-effector pose and $\mathbf{\Theta}$ the joint variables (active and/or passive)
- the workspace of the robot is bounded

On the other hand we don't impose any particular form for $\mathcal{C}$ or any choice on the parameters $\mathbf{X}$.

The output of the planner will be a list of *way points* in the operational space, the start point S being the first element of the list while the goal point G will be the last element (S, G will be provided by the global planner). A *trajectory* between two way points $\mathbf{W_j}, \mathbf{W_{j+1}}$ will be defined by a set $\mathbf{F}$ of n continuous, analytical functions F_k such that any pose $\mathbf{X_t}$ on the trajectory may be written as $\mathbf{X_t} = \mathbf{F}(\mathbf{W_j}, \mathbf{W_{j+1}}, t)$, where t is the time and lie in the range [0,1] and such that $\mathbf{F}(\mathbf{W_j}, \mathbf{W_{j+1}}, 0) = \mathbf{W_j}$, $\mathbf{F}(\mathbf{W_j}, \mathbf{W_{j+1}}, 1) = \mathbf{W_{j+1}}$. At time t the k-th component of $\mathbf{X_t}$ will be obtained as $F_k(\mathbf{W_{j_k}}, \mathbf{W_{j+1_k}}, t)$. Any choice of F_k is possible as soon as it is possible to formulate analytically the optimality criteria for the trajectory (if any is used). For the sake of simplicity we will use here linear functions i.e. $\mathbf{X_t} = \mathbf{W_j} + t(\mathbf{W_{j+1}} - \mathbf{W_j})$.

The number of way points may not be defined in advance. If an optimality criteria is used, the final number of way points of the trajectory will be such that adding a new way point will not improve significantly the value of the optimality criteria. Note however that this local planner is only able to provide a limited number of way points (typically 3). But requiring more way points means that the global planner has performed poorly.

Any pose belonging to the trajectory between two way points will be guaranteed to satisfy the kinematic constraints. This is evidently a key issue that is addressed in the next section.

A. Checking a trajectory between two way points

We have already proposed a method to check if a given trajectory, defined by analytical function of the time, satisfies the kinematic constraints [11] and we will just outline its principle. We rely on *interval analysis* to determine if a kinematic constraint $\mathcal{C}_i$ is satisfied or violated over a given time range. Being given an analytical function $f(x_1, \ldots, x_m)$ and ranges $[\underline{x_i}, \overline{x_i}]$ for the unknowns x_i, interval analysis allows to determine simply an interval $[a, b]$, called an *interval evaluation* of f, such that for all x_i in $[\underline{x_i}, \overline{x_i}]$ we have $a \leq f(x_1, \ldots, x_m) \leq b$ i.e. a, b are lower and upper bound of the minimum and maximum of f over the ranges. Usually a, b overestimate the minimum and maximum but the overestimation decreases with the width $\overline{x_i} - \underline{x_i}$ of the range. A major interest of interval analysis is its robustness with respect to round-off errors: even if such errors occurred, the range $[a, b]$ is still guaranteed to include the real minimum and maximum of f.

Being given a time interval $[t_0, t_1]$ this method will be used to determine ranges for the pose parameters along the trajectories defined by the set $\mathbf{F}$. In turn these ranges allows one to determine a range $[a_i, b_i]$ for the value of each kinematic constraint $\mathcal{C}_i$. If $a_i > 0$, then the constraint is violated at any time in $[t_0, t_1]$, while if $b_i < 0$, then the constraint is satisfied all over the time range. It may occur that due to the overestimation of interval analysis we get $a_i < 0, b_i > 0$, so that we cannot state if the constraint is violated or satisfied. In that case we will bisect the time range in two ranges $[t_0, (t_0 + t_1)/2], [(t_0 + t_1)/2, t_1]$ that will be stored in a list. All ranges in this list will be processed until we may determine for all boxes in the list that either all constraints are satisfied or that for some time range a constraint is violated. It may also occur that for a given time interval analysis does not allow to determine if one (or more) kinematics constraint is satisfied or violated because round-off errors do not allow to determine the sign of the constraints.

It must be noted that although we have assumed an analytical form for $\mathcal{C}$, this is not strictly necessary. Indeed our approach requires only to be able to calculate an interval evaluation of an index stating if the constraint is satisfied or violated. For example for closed-chain robot a singularity occurs if the Jacobian matrix is singular. To detect such occurrence we use the sign of the determinant without having its analytical form: only analytical forms of the components of the Jacobian are required [14].

A first extension of this algorithm is to consider uncertainties in the geometric modeling of the robot. In mechanical engineering they correspond to the bounded manufacturing tolerances. We use a worst case scenario by assuming that the real value of the geometrical parameters may be any value within known ranges. If there width are relatively small their ranges are used as it for the interval evaluation of the kinematic constraints. In the worst case it may happen that for a given fixed time that the interval evaluation of one (or more) kinematic constraint has a lower negative bound and positive upper bound, thereby not allowing to determine if the constraint is satisfied: in that case the parameters uncertainties are added as new variables and submitted to the bisection process. Note that the same process may be used if we assume *control errors*, i.e. the robot will not follow exactly the planned trajectory, the differences between the trajectory and the robot pose being still bounded.

At this point we are thus able to design an algorithm $\mathcal{A}_P(\mathbf{W_j}, \mathbf{W_{j+1}})$ that returns 1 if the trajectory between these two way points satisfies the kinematic constraints, -1 if it violate them or 0 if the trajectory is unsafe (i.e. at some time we cannot determine if a kinematic constraints is satisfied or violated).

We may extend this algorithm to deal with set of way points defined by a *box* in the operational workspace. This new algorithm, $\mathcal{A}_{\mathcal{P}}(\mathcal{W}_j, \mathcal{W}_{j+1})$, takes as input two boxes

of the operational workspace that include the potential way points $\mathbf{W_j}, \mathbf{W_{j+1}}$ and returns 1 if the trajectory between any poses in the boxes $\mathcal{W}_j, \mathcal{W}_{j+1}$ satisfies the constraint, -1 if all trajectories violate them. Furthermore the algorithm returns 0 if it is not able to complete its task after a limited number of iterations.

B. The optimality criteria

We will assume that we can calculate an interval evaluation of the eventual optimality criteria $\mathcal{H}$ (without loss of generality we will assume that $\mathcal{H}$ should be positive and minimal). Our algorithm will determine a trajectory with a criteria that is at most ϵ, that is defined by the end-user, away from its minimal value.

Now assume that we have determined a trajectory for which the value of $\mathcal{H}$ is l: using the interval evaluation of the criteria we are able to design an algorithm $\mathcal{O}(\mathcal{W}_{1,\ldots,m})$ that takes as input a set of boxes for the m way points and returns 1 if the lower bound of the interval evaluation of $\mathcal{H}$ is lower than $l - \epsilon$ (i.e. there may be trajectories having a value of $\mathcal{H}$ that is lower than $l - \epsilon$ with way points in the boxes), -1 if the lower bound of the interval evaluation of $\mathcal{H}$ is greater than $l - \epsilon$ (i.e. even if there are valid trajectories with way points in the boxes, they will have a higher value of $\mathcal{H}$ than the current trajectory). Otherwise the algorithm will return 0.

III. THE LOCAL PLANNER

The first objective of the local planner is to determine if there is valid trajectory with one way point. This eventual way point must be be located in the workspace, hence we have bounds for its coordinates. A *way point box* is a box of the operational space that may contain the way point $\mathbf{W_1}$. The robot workspace will be decomposed into way point boxes and for each box we will use algorithm $\mathcal{A}_P$ on the trajectory components $S\mathbf{W_1}, \mathbf{W_1}G$ to test if the trajectory with the center of the box as $\mathbf{W_1}$ is valid. If not we will use algorithm $\mathcal{A}_\mathcal{P}$ to determine if the current box does not contain any valid trajectory, in which case we will discard it, otherwise the box will be bisected. We will not elaborate on the bisection strategy (i.e. which variable will be bisected) but it plays an important role in the efficiency of the algorithm.

If an optimality criteria has been defined and a valid trajectory has been found with $\mathcal{H} = l$, then we use $\mathcal{O}$ to determine if the box may include trajectories with $\mathcal{H} < l - \epsilon$. If this is not the case the box will be discarded.

A. Incremental addition of way points

It may occur that there is no valid trajectory with only one way point or that the purpose of the planner is to determine a trajectory whose value of $\mathcal{H}$ is optimal up to the accuracy ϵ. In that case we will incrementally add way points until either a trajectory (or a better value of $\mathcal{H}$) is found or we are able to determine that the current trajectory is optimal. The dimension of the way point boxes will be $k\,n$ where k is the number of way points.

B. The algorithm

We present here the algorithm for the case where an optimality criteria has been defined. The number of way points will be denoted N and the algorithm starts with $N = 1$. A flag T, initially set to 0, will be set to 1 if a valid trajectory has been found. The value of $\mathcal{H}$ for this trajectory will be denoted $\mathcal{H}_T$. The algorithm will process a list $\mathcal{L}$ of way point boxes $\mathcal{B} = \{\mathcal{W}_1, \ldots \mathcal{W}_N\}$. The $i-th$ element of the list will be $\mathcal{B}_i$ and r will denote the total number of way point boxes in the list. The mid point of a way point box is obtained by taking the mid point of all the ranges of the variables in the box: it correspond to a set of fixed poses and will be written as $Mid(\mathcal{B}_i)$. The value of the optimality criteria obtained for a trajectory with N way point will be denoted l_o^N (l_o^0 will be equal to $+\infty$). When starting the algorithm we have $T = 0$ and there is one way point box $\mathcal{B}_1$ in $\mathcal{L}$, whose ranges correspond to the ranges of the robot workspace $\mathcal{E}$. The Cartesian product of N box $\mathcal{E}$ will be denoted $\mathcal{E}^N$.

$r = i = 1$, $\mathcal{L} = \{\mathcal{E}^N\}$, $T = 0$
while true **do**
    if $T = 1$ **then** $l_o^N = l_o^{N-1}$
    while $i \leq r$ **do**
        if $T = 1$ **and** $\mathcal{O}(\mathcal{B}_i) = -1$ **then** $i = i + 1$, **next**
        if $\mathcal{A}_\mathcal{P}(\mathcal{B}_i) = -1$ **then** $i = i + 1$, **next**
        if $\mathcal{A}_P(Mid(\mathcal{B}_i)) = 1$ **then**
           $l_w = \mathcal{H}(Mid(\mathcal{B}_i))$
           if $T = 1$ **and** $l_w < l_o^N$ **then** $l_o^N = l_w$
           if $T = 0$ **then** $T = 1$, $l_o^N = l_w$
        bisect $\mathcal{B}_i$, $r = r + 2$, $i = i + 1$, **next**
    end-do
    if $T = 1$ **and** $l_o^N < l_o^{N-1} - \epsilon$ **then** $N = N + 1$, **next**
    if $T = 1$ **and** $l_o^N = l_o^{N-1}$ **then break**
    if $T = 0$ **then** $N = N + 1$, **next**
end-do

Efficiency of the algorithm will be improved by adding *filtering* methods, i.e. heuristics that reduce the size of a way point box without using bisection. There are standard interval analysis filtering methods [7] but also adapted methods as will be shown in the examples.

Note that exiting the algorithm as soon as adding only one way point does not allow to improve significantly l_o (i.e. leads to a decrease larger than ϵ) does not allow to guarantee that a close approximation of the optimal trajectory has been found. In practice however this will imply that the global planner has performed poorly.

A first drawback of this algorithm is that its complexity quickly increase with the number of way points. This will be confirmed by the example: determining a trajectory with one or two way points is fast but the computation time drastically increase for 3 way points. But we may reduce the computation time for 2 (or more) way points with an incremental use of the local planner, an approach that we will call the *intermediary step method*. Assume that a trajectory with one way point has been found and let

$S, \mathbf{W_1}, G$ be the current trajectory. We use the local planner with a threshold equal to, for example, $\epsilon/2$ to determine if there is a trajectory with one way point $\mathbf{W_1}'$ between $S, \mathbf{W_1}$ that reduces the value of $\mathcal{H}$ compared to its value in the current trajectory. If this is the case we get the trajectory with 2 way points $S, \mathbf{W_1}', \mathbf{W_1}, G$ that has already a better $\mathcal{H}$ than the one way point trajectory. The process may be repeated with $\mathbf{W_1}, G$ and we will select the two way points trajectory that leads to the best improvement of $\mathcal{H}$. We get therefore a better initial trajectory for the local planer when looking for a trajectory with 2 way points.

A second drawback of this algorithm is its lack a memory. Indeed when checking a trajectory with two or more way points we will examine possible location for $\mathbf{W_1}, \mathbf{W_2}$ that have already been determined as non valid when we have looked for a trajectory with one way point.

IV. EXAMPLES

Four our tests we have used a 6 d.o.f. parallel robot that is presented in the next section.

A. Parallel robot

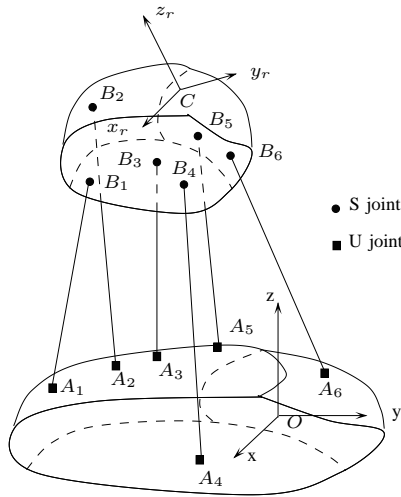As example of complex 6-dof closed-loop robot we will consider a Gough platform (figure 2). The fixed frame

Fig. 2.   A Gough-Stewart platform

$(O, \mathbf{x}, \mathbf{y}, \mathbf{z})$ will be called the *base frame* while a mobile frame $(C, \mathbf{x_r}, \mathbf{y_r}, \mathbf{z_r})$ attached to the platform will be called the *mobile frame*. The pose of the platform will be parametrized by the location of C in the base frame and 3 angles will be used to define the orientation of the mobile frame with respect to the base frame.

Motion of the platform are obtained by changing the length of the 6 legs, that are attached on the base at A_i and on the platform at B_i. The coordinates of the vector $\mathbf{OA_i}$ in the reference frame and the coordinates of $\mathbf{CB_{i_r}}$ in the mobile frame are known. The legs are restricted to have a minimal and maximal length, thereby restricting the workspace of this robot. Being given the location of the center C of the platform in the reference and the rotation

matrix $\mathbf{R}$ between the mobile and reference frame, the length ρ of a leg is determined as

$$\rho^2 = ||\mathbf{AO} + \mathbf{OC} + \mathbf{RCB}_r||^2 \qquad (1)$$

and we should have

$$\rho_{min}^2 \leq \rho^2 \leq \rho_{max}^2$$

The shape of the workspace due to this limitation is quite complicated [12] and planning a trajectory to fully lie in this workspace is a complex task. Other kinematic constraints may be considered as well: limitation of the passive joints motion at A, B [16], singularity avoidance [1], [5], [6], [8], [15] or leg interference [3], [13]. We will consider here only the limitation on the leg lengths and limits on the motion of the passive joints.

The optimality criteria that we will use is the length of the trajectory of the center C of the platform. This allows to use a dedicated filtering method that is available as soon as a trajectory of length l with one way point has been determined. Indeed any potential way point that will lead to a smaller trajectory length (i.e. smaller than $l - \epsilon$) will lie within an ellipsoid which is defined as the set of points M such that the sum of the distances from M to S, G is equal to $l - \epsilon$. A bounding box E of this ellipsoid may be found and its intersection I with the way point box B is calculated. If I is empty, then the box is discarded otherwise I is substituted to B.

B. Implementation and tests

To test the motion planning algorithm we have used the C++ interval arithmetic package `BIAS/Profil` and components of our interval analysis library `ALIAS`.

We assume first that the geometry of the robot is perfectly known (i.e. the location of the A_i, B_i are exact as indicated in table I). The minimal and maximal leg lengths

Leg	1	2	3	4	5	6
x_A	-9	9	12	3	-3	-12
y_A	9	9	-3	-13	-13	-3
x_B	-3	3	7	4	-4	-7
y_B	7	7	-1	-6	-6	-1
z_A, z_B	0	0	0	0	0	0

TABLE I

COORDINATES OF THE A, B POINTS FOR THE TEST ROBOT.

are 52.249605 and 55.749605.

The various poses are defined by the 3 coordinates of C followed by the three rotation angles. The start point S is $(0,0,52.1,0,0,0)$ and the goal G is $(11,5,52.1,0,0,0)$ (having pose with the same orientation and in an horizontal plane allows for an easy representation of the workspace). The distance between S, G is 12.083. First we consider only the limitation on the leg lengths as kinematic constraints. It may been that on figure 3 that the workspace has voids in its interior. We first try to determine a trajectory that keep constant the orientation of the platform. Trajectories with 1, 2 and 3 way points are presented in figure 3: their

lengths are respectively 19.5373, 17.1118, 16.7887 and these trajectories are established respectively in 1 second, 17 seconds and 11 mn 44 seconds on a DELL D400 laptop. As expected the computation time grows quickly with the number of way points. But the computation time does not significantly increase if we add as kinematic constraint that the determinant of the inverse Jacobian matrix has a value larger than a given threshold for any pose on the trajectory in order to ensure that the trajectory lies on a given kinematic branch.
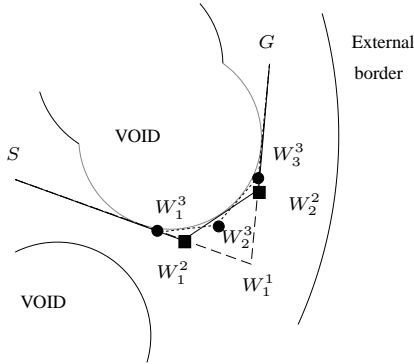
Fig. 3. Trajectories with 1, 2 (squared way points) and 3 (circled way points) way points.

Note that in this example the global planner should have provided more sampling points. The threshold ϵ for the length of the trajectory is fixed to 0.3.

If we now relax the constraint that the trajectory should lie in the plane $z = 52.1$ by allowing a z value in the range [50,55] while keeping the same orientation we found out in 0.26 second that the trajectory with one way point at (5.562,2.5,52.5351,0,0,0) has a length of 12.1144 (figure 4). As the minimal value of $\mathcal{H}$ is 12.083, the algorithm stops as adding new way point cannot make the length decrease by more than 0.3. Setting the threshold ϵ to 0.01 allows to determine a trajectory with length 12.0917 in a computation time of 1.2 seconds.
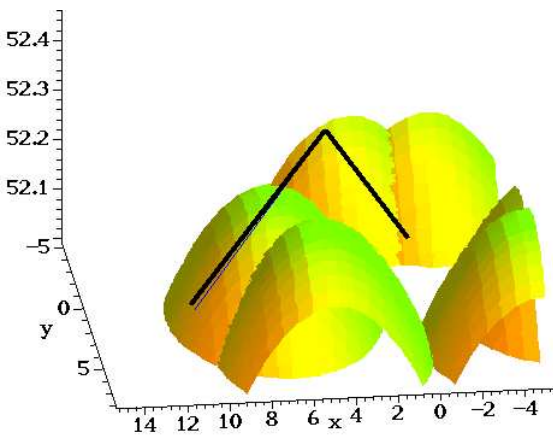

Fig. 4. A 3D trajectory. Note that the scales are different on the x, y, z axis for a better understanding of the trajectory.

We may also keep the trajectory in the same plane while allowing to change the platform orientation. For the same trajectory and allowing the orientation angles to lie in the range [-5,5] degrees, we find a one way point trajectory of length 12.3967 in 15 seconds and the planner immediately determine that adding a new way point will not shorten the trajectory length by more than 0.3.

We may also have to deal with uncertainties in the robot modeling. We assume a tolerance error of $\pm$ 0.01 on each coordinates of the A_i, B_i meaning for example that the x coordinate of A_1 may have any value in the range [-9.01,-8.99]. A trajectory with one way point and length 21.2389 is found in 8 seconds, with two way points and length 17.8501 in 5 minutes and 29 seconds. For three way points a trajectory of length 17.4191 is found in 4h27mn49s. In the later case by using the *intermediary step method* we may reduce this time to 3h41mn and the intermediary path allows to get an initial three way point trajectory of length 17.6235 in 16mn10s. These trajectories are presented in figure 5. If we allow z to lie in the range [50,55] a minimal
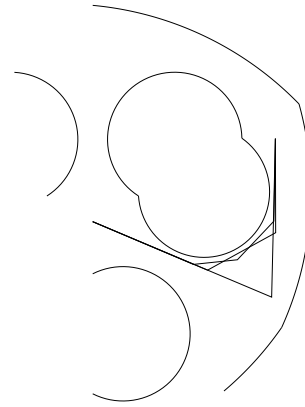

Fig. 5. Trajectories with one to three way points when the coordinates of A_i, B_i have a tolerance of $\pm$ 0.01

length trajectory is found in 2.65s.

Assume now that we have limits on the passive joint motion. To model this limits for the joints at A_i we assume that the angle between the $i-th$ leg and a fixed direction defined by the unit vector $\mathbf{n_i}$ should be lower than a threshold μ_i. This constraint may be written as

$$|\frac{\mathbf{A_iB_i.n_i}}{\rho_i}| \leq \cos(\mu_i) \qquad (2)$$

We consider the trajectory between S (0,0,52.2) and G (-8,5,52.2) and assume that the angle between the vertical direction and the legs should not exceed 17 degrees. Without considering these constraints we find a trajectory of length 16.3498 with one way point and of length 13.9694 with two way points (figure 6). But if the passive constraints are taken into account the algorithm found out in 3 seconds that there is no trajectory with only one way point. The previously calculated trajectory with two way points is also not satisfactory as part of it violate the joint limits. But a two way point trajectory that takes into account the joint limits may be found with a length of 14.1181 with
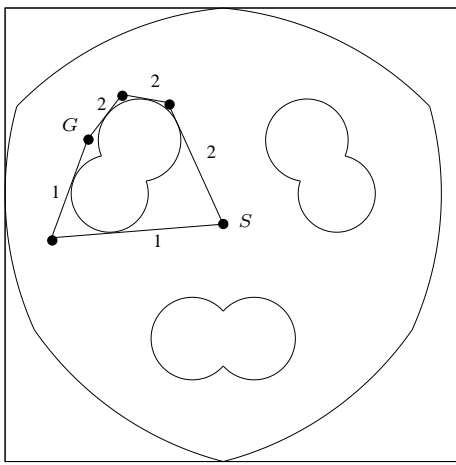
Fig. 6. Trajectories with one (1) and two (2) way points when the constraints on the passive joint limits are not considered.

a threshold on $\mathcal{H}$ of 0.1. Allowing an out of the plane trajectory leads to a one way point trajectory of length 9.5146 that is computed in less than one second (figure 7). A safe trajectory with one way point may also be found if we have an uncertainty of ± 0.01 on the location of A, B. If we keep the trajectory in the plane $z = 52.2$ but
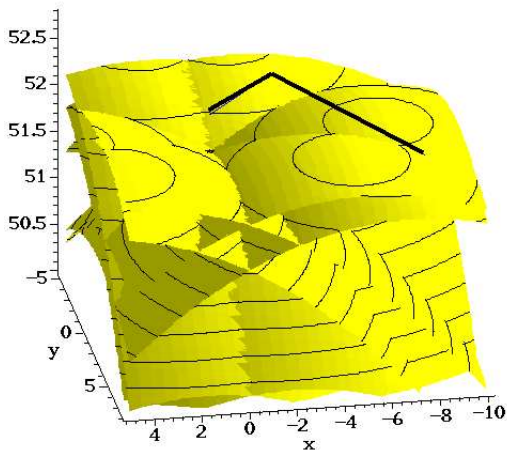

Fig. 7. A 3D trajectory that satisfies constraints on the passive joint motion

allow a change in the orientation angles in the range [-5,5] degrees we find a path of length 10.2584 with one way point in 4 seconds. A similar result is obtained if we have an uncertainty of ± 0.01 on the location of A, B.

V. CONCLUSION

Safe motion planning of closed-loop robot requires to verify complex kinematic constraints apart of the classical obstacle avoidance problem. Global motion planner have the advantage to provide a draft trajectory but a local planner must then be used to provide a certified trajectory (i.e. such that the kinematic constraints are satisfied all along the trajectory). The local planner proposed in this paper is based on interval analysis and allows both to

certify the trajectory and to manage possible uncertainties in the robot modeling.

REFERENCES

[1] Bhattacharya S., Hatwal H., and Ghosh A. Comparison of an exact and an approximate method of singularity avoidance in platform type parallel manipulators. *Mechanism and Machine Theory*, 33(7):965–974, October 1998.

[2] Canny J. *The complexity of robot motion planning*. MIT Press, Cambridge, 1988.

[3] Chablat D. and Wenger P. Moveability and collision analysis for fully-parallel manipulators. In *12th RoManSy*, pages 61–68, Paris, July, 6-9, 1998.

[4] Cortés J. and Siméon T. Probabilistic motion planning for parallel mechanisms. In *IEEE Int. Conf. on Robotics and Automation*, pages 4354–4359, Taipei, September, 14-19, 2003.

[5] 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.

[6] Dash A.K. and others . Workspace generation and planning singularity-free path for parallel manipulators. *Mechanism and Machine Theory*, 40(7):778–805, July 2005.

[7] Hansen E. *Global optimization using interval analysis*. Marcel Dekker, 2004.

[8] Jui C.K.K. and Sun Q. Path tracking of parallel manipulators in the presence of force singularity. *ASME J. of Dynamic Systems, Measurement and Control*, 127(4):550–563, December 2005.

[9] Latombe J.C. *Robot Motion planning*. Kluwer Academic Publishers, Boston, , 1991.

[10] LaValle S.M. *Planning algorithm*. Cambridge University Press, Cambridge, 2006.

[11] Merlet J-P. A parser for the interval evaluation of analytical functions and its applications to engineering problems. *J. Symbolic Computation*, 31(4):475–486, 2001.

[12] 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.

[13] Merlet J-P. and Daney D. Legs interference checking of parallel robots over a given workspace or trajectory. In *IEEE Int. Conf. on Robotics and Automation*, pages 757–762, Orlando, May, 16-18, 2006.

[14] Merlet J-P. and Donelan P. On the regularity of the inverse jacobian of parallel robot. In *ARK*, pages 41–48, Ljubljana, June, 26-29, 2006.

[15] Sen S., Dasgupta B., and Mallik A.K. Variational approach for singularity-path planning of parallel manipulators. *Mechanism and Machine Theory*, 38(11):1165–1183, November 2003.

[16] Su H-J., Dietmaier P., and J.M. McCarthy. Trajectory planning for constrained parallel manipulators. *ASME J. of Mechanical Design*, 125(4):709–716, December 2003.

[17] Trinkle J.C. and R.J. Milgram. Complete path planning for closed kinematic chains with spherical joints. *Int. J. of Robotics Research*, 21(9):773–789, September 2002.

[18] Xie D. and Anamato N.M. A kinematics-based probabilistic roadmap method for high dof closed chain systems. In *IEEE Int. Conf. on Robotics and Automation*, New Orleans, April, 28-30, 2004.

[19] Yakey J.H. and others . Randomized path planning for linkages with closed kinematic chains. *IEEE Trans. on Robotics and Automation*, 17(6):951–958, December 2001.