DEMOCRAT: A DEsign MethOdology for the Conception of Robot with parallel ArchiTecture

J-P. Merlet
INRIA Sophia-Antipolis
BP 93, 06902 Sophia-Antipolis Cedex, France
E-mail: Jean-Pierre.Merlet@sophia.inria.fr

Abstract

This paper presents a design methodology for parallel robots having to satisfy a set of performance constraints. After a reduction of the number of the design parameters some performance requirements are used to compute a closed region in the parameters space (in which a point define an unique robot geometry) which contain all the robot geometries satisfying the requirements. The requirements which have not been used in the first step are described then in a high level language procedure which take as input a robot geometry and determine if the robot fulfill the users requirements. The closed region determined at the first step is then discretized and the robot geometry which correspond to each node of the discretisation is submitted to the procedure. The node leading to the robot(s) fulfilling at best the constraints is the design solution.

1 Introduction

Parallel robot have been extensively studied in the recent years and are now starting to appear as commercial products. One of the challenging problem in this field is to determine the "optimal" robot geometry with respect to the user's requirements.

In this paper this optimal design problem will be restricted to the Gough platform architecture i.e. a parallel robot consisting of a base plate and a mobile platform connected in parallel by 6 identical legs. An universal joint connect the leg to the base while a ball and socket joint connect the leg to the platform. In each of the leg a linear actuator enable to change the leg length. The position/orientation of the platform is controlled by changing the leg lengths, which are therefore the articular variables of the robot.

The optimal design of this type of parallel robot has drawn a lot of interest of researcher [1, 2, 4, 5, 6, 7, 8, 9, 12, 13, 14]. Basically we may distinguish two different approaches:

- the "trial-and-error" method
- the "cost function" approach

In the first method the designer changes interactively the design parameters and uses a mechanical simulator to determine the performances of the robot. This approach is tedious as the design parameters are numerous and needs a lot of time as many performance criterion are difficult to estimate.

The "cost function" approach follows the above steps:

1. reduction of the number of design parameters by appropriate assumptions
2. construction of a real valued function whose value should be minimal for the "optimal" robot
3. utilization of a numerical procedure for computing the parameters that minimize the cost-function by changing the values of the design parameters

Step 1 is clearly justified as they are usually an important number of design parameters: for example a Gough-platform has at least 36 parameters (the coordinates of all the passive joint centers). Step 2 is in fact difficult to implement:

- being given the user's requirements the expression of the cost function is difficult to find
- many performance requirements of parallel robots are antagonistic. Therefore the result will be deeply affected by the weight imposed on the requirements in the cost function
- the criterion used in the cost function must be chosen with care. For example Gosselin has shown that the Gough-type platform with the maximal workspace volume is a robot which is singular in most of its configurations
- the cost function may have numerous local minima and consequently the minimization procedure may have difficulty to locate the global minima
- the calculations of many features of parallel robots which may appear in the cost function

are computer intensive. For example the computation of maximal articular forces that the robot will sustained for a given load as the robot moves in a given workspace must be performed with a discretisation method in the 6-dimensional workspace and will therefore need an important amount of computer time

The purpose of `DEMOCRAT` is to provide an alternative solution to the optimal design problem. The successive steps of this methodology will be presented in the following sections.

2 Reduction of the number of parameters

In the sequel the centers of the passive joints on the base will be denoted A_i, their equivalent on the moving platform B_i. A reference frame $O, (x, y, z)$ and a moving frame $C, (x_r, y_r, z_r)$ are defined. The end-effector position will be defined by the location of C in the reference frame and its orientation by a rotation matrix R. The performances of a parallel robots are dependent upon various design parameters. For any features the 36 coordinates of the joint centers A_i, B_i will be always important. To reduce this number we will make the following assumptions (figure 1):

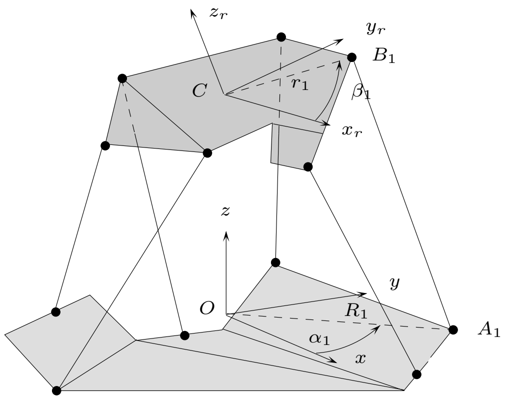

Figure 1: The design parameters

- we know the lines going through O on which lie the joint centers A_i (in other words the angles α_i are known).
- we know the lines going through C on which lie the joint centers B_i (in other words the angles β_i are known).
- the relative heights of A_i, B_i are known

Consequently the joint centers location are determined by the distances R_1^i from A_i to O and the distances r_1^i from B_i to C. Therefore we have twelve design parameters and a robot geometry is defined by a point in the 12 dimensional parameters space.

2.1 The design planes

Some features of parallel robots, considered for one leg i are only dependent upon the location of the pair (A_i, B_i) and not upon the location of the other joint centers. The leg lengths is defined as $||\mathbf{AB}||$:

$$||\mathbf{AB}|| = ||\mathbf{AO} + \mathbf{OC} + R\mathbf{CB_r}|| \quad (1)$$

where $\mathbf{CB_r}$ are the coordinates of the B points in the mobile frame. The length of leg i is therefore a function of the pair (R_1^i, r_1^i) and is not a function of the pairs $(R_1^j, r_1^j), j \neq i$. Similarly the actuator velocities $\dot{\rho}$ for a given generalized velocity $\dot{\mathbf{X}}$ of the platform are obtained by:

$$\dot{\rho} = J^{-1}\dot{\mathbf{X}} \quad (2)$$

where J^{-1} is the inverse jacobian matrix of the robot. For a given leg the corresponding row of this matrix is a function of only the pair (R_1^i, r_1^i).

For the criterion involving this type of feature we may therefore split the parameters space in 6 *design planes* in which the x coordinates of a point represent a value for the design parameters R_1^i and the y coordinate a value for r_1^i. A robot geometry is defined by a set of 6 points in the 6 design planes.

Note that the design planes may be reduced to an unique design plane if the base and platform are planar and the joint centers A_i, B_i all lie on a circle. In that case we have only two design parameters which are the radius of the base and the radius of the moving platform.

3 The search regions

At the current stage we have defined the parameters space for the robot geometry. The next stage will be to determine all the closed regions of the parameters space (which will be called the *search regions*) such that some possible users requirements will be fulfilled only by robot geometries defined by points inside the search regions.

In the current version of `DEMOCRAT` two types of users requirements may be used to determine the search regions. Both use the fact that for some criterion the parameters space may be split in 6 design planes. These criterion are then workspace requirements and the maximal articular velocities requirements.

3.1 The `design` algorithm

The algorithm `design` [11] enable to deal with the workspace requirements. As input it takes the minimal and maximal values of the leg lengths ρ_{min}, ρ_{max}, optional mechanical limits on the passive joints and a set of segments, called the *segment trajectories*, describing the trajectories for the point C (the orientation of the end-effector being fixed for each segment) that the robot must be able to perform with the leg lengths in the range $[\rho_{min}, \rho_{max}]$ and with passive joint motions within the mechanical limits.

As output `design` will compute a closed region in each of the design planes so that a robot geometry defined by 6 points in these regions will be such that the workspace of the robot will include all the segment trajectories.

Let us summarize the theory underlying `design`:

- for each segment trajectory the robots such that the constraint $\rho \leq \rho_{max}$ is satisfied for any position on the segment must have their design plane points inside two ellipses $\mathcal{E}_{M1}, \mathcal{E}_{M2}$
- for each segment trajectory the robots such that the constraint $\rho \geq \rho_{min}$ is satisfied for any position on the segment must have their design plane points outside an union of ellipses $\mathcal{E}_m$
- therefore the robots satisfying the constraints $\rho_{min} \leq \rho \leq \rho_{max}$ have their design plane points inside the region $\mathcal{R} = (\mathcal{E}_{M1} \cap \mathcal{E}_{M2}) - \mathcal{E}_m$
- the region $\mathcal{R}^j$ are computed for all the trajectories and the final region is obtained as $\cap\mathcal{R}^j$.

Figure 2 shows an output of the `design` algorithm in the case where we have only one segment trajectory and only one design plane. The x axis represent possible value for R_1 while the y axis represent possible value for r_1. The region $\mathcal{R}$ is drawn in thick lines.

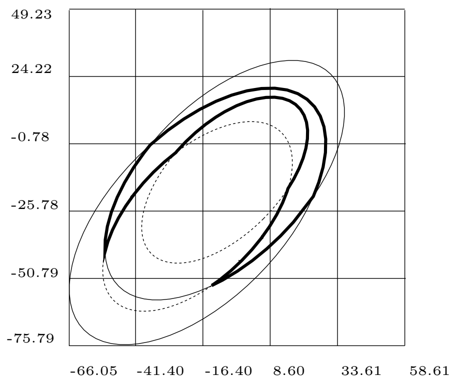

Figure 2: An output of the `design` algorithm. All the robots such that the specified segment trajectory lie within their workspace have their design plane point inside the region drawn in thick line.

3.2 The `vitesse_design` algorithm

The algorithm `vitesse_design` [10] takes as inputs a bound $\dot{\rho}_l$ on the absolute values of the velocity of the linear actuators, a set of segment trajectories and a velocity input for the point C called the *velocity objective.* The outputs are 6 regions $\mathcal{C}$ in the design planes describing all the robots such that the velocity objective may be performed for any position of C on the trajectories with $|\dot{\rho}| \leq \dot{\rho}_l$.

Let us summarize the basic principles of the algorithm. Note first that any position of C on a segment trajectory M_1M_2 may be characterized by a scalar λ in the range [0,1]with $\mathbf{OC} = \mathbf{OM_1} + \lambda\mathbf{M_1M_2}$. The algorithm has the following features:

- on a trajectory the maximum of $\dot{\rho}^2$ is obtained either for $\lambda = 0, 1$ or for a value λ_n solution of a first order equation
- the equation $\dot{\rho}^2 = \dot{\rho}_l^2$ either for $\lambda = 0, 1, \lambda_n$ is a conic. These conics split the design plane into regions where $\dot{\rho}^2 \leq \dot{\rho}_l^2$ and $\dot{\rho}^2 \geq \dot{\rho}_l^2$. Let $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_n$ be the regions where $\dot{\rho}^2 \leq \dot{\rho}_l^2$ for $\lambda = 0, 1, \lambda_n$
- the equations $\lambda_n = 0$ and $\lambda_n = 1$ define conics. These conics split the design plane into region and let us denote $\mathcal{R}_{\backslash\prime}, \mathcal{R}_{\backslash\infty}$ the regions where respectively $\lambda_n \geq 0$ and $\lambda_n \leq 1$

The output of the algorithm is therefore

$$\mathcal{C} = (\mathcal{R}_0 \cap \mathcal{R}_1) \cap (\mathcal{R}_n \cap \mathcal{R}_{\backslash\prime} \cap \mathcal{R}_{\backslash\infty})$$

Figure 3 shows an output of the `vitesse_design` algorithm in the case where we have one design plane and one segment trajectory. Evidently by computing

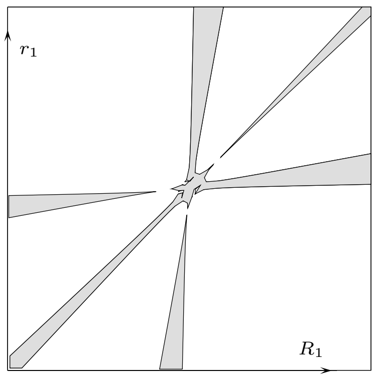

Figure 3: An output of the `vitesse_design` algorithm. The gray region is the output of the algorithm.

the intersection of the regions obtained as results from `design` and `vitesse_design` we will obtain the region where both constraints will be satisfied. Note that both algorithms are rather fast: the computation time ranges from 100 ms to a few minutes according to the number of segments trajectories.

4 The search

Using the result of the previous sections we have determined the search region. We may now consider the other constraints for determining the optimal robot. A grid is created on the search region: each of its node represents an unique location for the pair (A_i, B_i).

For each node `DEMOCRAT` will create the corresponding robot and test if it satisfies the user's requirements.

4.1 Specifying the user's requirement

The user's requirements are specified in a high-level C-like language. This language has variables, arrays, loops, test conditions etc..and additional instructions related to features of parallel robots. For example the instruction:

```
%V0=minimal stiffness in cube
            center 0 0 30 , 10 10 10
```

will enable to compute the minimal values of the diagonal of the stiffness matrix of the robot as C moves in a cube centered in (0,0,30) and whose edges have a lengths 10. These stiffness will be stored in the array `V0`.

The user's requirement are defined as a procedure which is evaluated by `DEMOCRAT` for each node of the grid. It returns 0 if the robot does not fulfill the user's requirements, 1 if it fulfills the requirements (in which case it is added to the list of solutions) and 2 if it fulfills the requirements and is better than the previous solution.

Let us consider an example. The requirements are to determine the robot whose workspace is at least a cube centered in (0,0,30) and whose edges have a lengths 10, has no singularity inside the cube, has a maximal positioning error along the x axis better than 0.4 for a sensor errors of ± 0.1 for any position in the cube and whose minimal stiffness along the x axis has the greatest possible value. Figure 4 shows the description of these requirements.

4.2 Efficient evaluation of the robot performances

The determination of some performances of a parallel robot may be computer intensive as they may need to be determined for any position of the end-effector in a given volume. Most of the performances may be evaluated for a given posture of the platform but then a discretisation should be performed in the 6 dimensional workspace to evaluate the performances over the whole workspace. To deal with this problem we have developed new algorithms enabling to compute efficiently some performances. These algorithms are able to compute the features for any *translation workspace* i.e. for any position of C inside a given a given volume (note that it does not mean that we assume that position and orientation are decoupled: the workspace we are considering is constituted of all the possible positions of C for a given orientation of the platform). This volume may be a box or any volume defined by a set of cross-sections in the 3D space or may be also an hypercube in the articular space (for example the hypercube defined by $\rho_{min} \leq \rho \leq \rho_{max}$).

Note that for a general workspace which include orientation requirements we will still need to discretize the orientation components. But the discretisation will now be only in a 3-dimensional space instead of the initial 6-dimensional one, therefore reducing drastically the computation time. The following efficient algorithms are available in `DEMOCRAT`:

- `ro_extreme`: compute the minimal and maximal leg lengths necessary to describe the workspace
- `vitesse`: compute the minimal and maximal articular velocities needed to perform a cartesian/angular velocity in the workspace. It can also compute the range of motion of the passive joints of the robot.
- `raideur`: compute the minimal and maximal stiffness of the robot within the workspace
- `singularite`: determine if there is a singularity

```
/* at the first call we initialize the best stiffness to -1 */
if (first_call==1) bloc
    {best_stiffness} = -1
end_bloc
 /* we verify if there is a singularity in the workspace, if yes
just abort */

%0= singularity in cube center 0 0 30 , 10 10 10
if ( %0 >0 ) bloc
        abort
        end_bloc

/* sensor accuracy, articular stiffness and orientation */
sensor_accuracy= 0.1 , 0.1 , 0.1 , 0.1 , 0.1 , 0.1
articular_stiffness= 100 , 100 , 100 , 100 , 100 , 100
psi=0 teta=0 phi=0

/* to compute the accuracy we use a discretisation
method, so we have to define the steps size */
step x 1 step y 1 step z 1
/* now find the maximal x-errors for the workspace */
%V0=maximal accuracy in cube center 0 0 30 , 10 10 10
%1=%V0[1][1]
/* if the x-error is lower than 0.4 we may consider the stiffness,
otherwise we just abort */
if (%1 <0.4 ) bloc
    /* minimal stiffness for any position in the workspace */
    %V0=minimal stiffness in cube center 0 0 30 , 10 10 10
    {current_stiffness}=%V0[1][1]
        if ({current_stiffness} > {best_stiffness}) bloc
            {best_stiffness}={current_stiffness}

            save_R1_r1 /* save robot in result file */
            quit
        end_bloc
end_bloc
abort
```

Figure 4: An example of user's requirement description

inside the workspace

All these algorithms are based on exact methods and does not rely on a discretisation method. They are usually fast (the computation time ranges from a few milliseconds to a few seconds).

5 Implementation of DEMOCRAT

The current implementation of DEMOCRAT is written in Tcl/Tk and C++. The designer first determine the search region either by using the `design` or `vitesse_design` algorithms or by specifying a search box (or a mixture of them). Then the designer may execute the search phase after having defined the requirements in a file. DEMOCRAT will create the grid in the search region and start evaluating the robots defined by the nodes of the grid. This process is fully automated, DEMOCRAT displaying at regular time interval the total computation time and the main features of the current optimal robot (if any). The designer may stop the process at any time and backtrack if necessary. When the computation is finished the design parameters of the optimal robot(s) are stored in a file.

6 Advantages and drawbacks of DEMOCRAT

Clearly the most time consuming part of DEMOCRAT is the search phase as some features may need an important computation time to be evaluated. But this drawback will also be present with the cost function approach. With the reduction of the search region insure that this evaluation will be performed a minimal number of time.

Another drawback of DEMOCRAT is the assumption made on the position of the A_i, B_i. But as the computation time of `design` and `vitesse_design` is low it is possible to modify iteratively the values of the angles α, β until a satisfactory solution has been obtained. For example in one of our application we have modified incrementally the values of these angles until the search domain with the largest area has been obtained.

The advantages of DEMOCRAT is its versatility which is present at two levels:

- at the requirement level the language can be easily extended to deal with almost all types of requirements.
- at the implementation levels: as soon as new algorithms are discovered as well as for the determination of the search region or the search phase they can be easily included in DEMOCRAT

7 Application examples

The methodology proposed in the previous sections was used to design various fine positioning manipulators for the European Synchrotron Radiation Facility (ESRF) located in Grenoble. The purpose of these manipulators is to support various devices dealing with X-rays.

7.1 Example: the HFM2 manipulator

The nominal load for this manipulator is about 850 kg. The desired robot workspace, its accuracy and stiffness requirements (last line) are defined in table 1.

x	y	z	θ_x	θ_y	θ_z
±30	-	±20	±5	±5	0-10
±0.01	-	±0.1	±0.1	±0.1	±0.05
++	- -	-	-	-	+++

Table 1: Workspace, accuracy and stiffness requirements (units: mm, mrad)

The stroke of the linear actuator was fixed to 80 mm so that existing actuators can be reused.

It was assumed that all the joint centers were lying on circles (i.e. R_1 and r_1 are identical for all joints). Basically the joint centers are disposed symmetrically along three lines with an angle of 120 degree between them but to avoid interference between the actuators an angle γ of 20 degree was used for adjacent joint centers (figure 1), both on the base and on the moving platform. A set of 19 segment trajectories were specified for defining the desired workspace.

Our first problem was to determine the value of the minimal leg length ρ_{min}. To define this value we have first computed the area of the search region in the design plane as a function of ρ_{min}(figure 5). Using this graph it was possible to determine that ρ_{min} should lie between 590 and 835. Various trials has enabled to compute that a value of 750 was the most suited for our purpose. The resulting allowed zone is presented in figure 6.

Next we have to determine the geometry leading to the desired accuracy with the maximal possible error for the length sensor together with a resulting satisfactory stiffness. We have decided to consider the robot whose sensor accuracy should be not less than 2 μm and to select the robot whose stiffness for the rotation around the z axis is the best.

The allowed zone was sampled (each point of the zone represent an unique robot) and the sensor accuracy and stiffness was computed for each point. It was found that the robot with the maximum stiffness along

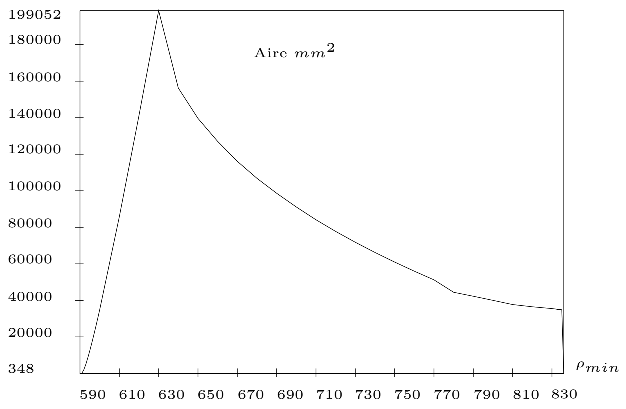

Figure 5: Variation of the area of the allowed zone as a function of ρ_{min}.

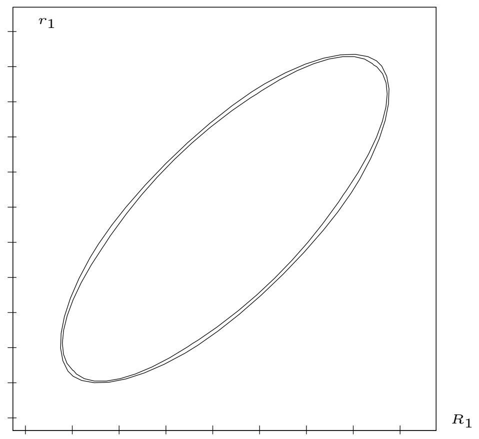

Figure 6: *Allowed zone for* ρ_{min}*=750.*

the x axis and for the rotation around the z axis has a sensor accuracy of 4μm, leading to the worst case accuracy defined in table 2. It may be seen that these

Δ_x	Δ_y	Δ_z	Δ_{θ_x}	Δ_{θ_y}	Δ_{θ_z}
0.01	0.0095	0.0049	0.0093	0.0105	0.0117

Table 2: Maximal positioning error for a sensor error of 4 μm (mm,mrad)

errors lie well within the accuracy requirement. It has also been noted that the maximal sensor error leading to the desired accuracy is extremely variable according to the geometry: a ratio of 120:1 between the best and worst case was observed. A 3D view of the workspace is presented in figure 7. The maximum articular force

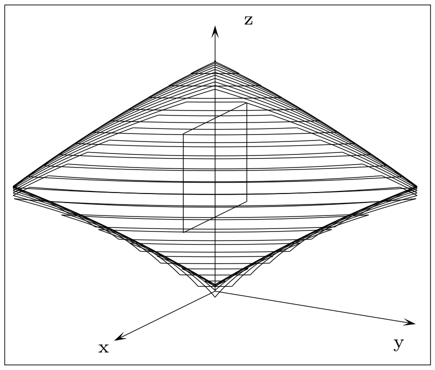

Figure 7: 3D view of the workspace for $\psi = \theta = \phi = 0$

was estimated to be at most 2000 N and it was determined that the ball-and-socked joint should enable a rotation of 6.27 degree.

In another design example the overall mass of the load and the bench was varying from 500 kg to 1000 kg and the accuracy was in the range of 1 to 10 μm. The result of the design process [3] is presented in figure 8.

The repeatability of this robot under a load of 230 kg was determined using X-ray interferometry: it was estimated to be better than 0.1 μm and therefore in compliance with the accuracy requirements. Ten other prototypes have now been built.

8 Conclusion

A methodology for the design of parallel manipulator has been proposed. Instead of relying on a cost function approach we first determine the minimal search domain in the parameters space which define

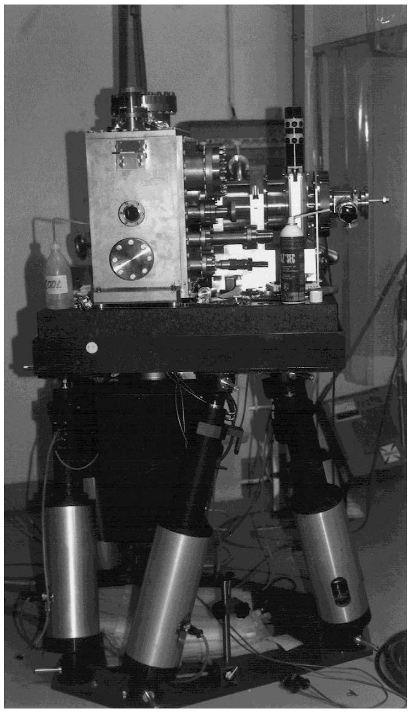

Figure 8: The ESRF-INRIA fine positioning device

all the robots satisfying some of the designer requirements. Then in a second step a discretisation of the search domain is used, each node defining an unique robot geometry. We then test if the robots corresponding to the nodes fulfill the requirements (described by using a high-level language), this enabling to determine the "optimal" robot(s). In order to increase the efficiency of this methodology it is necessary to develop algorithms enabling to compute efficiently the main features of a parallel robot. Some of them have been presented in this paper but still open problems remain like, for example:

- computing the maximal positioning errors of the robot, being given the sensor errors, for any workspace of the robot
- computing the maximal articular forces for a given load for any workspace of the robot

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] Claudinon B. and Lievre J. Test facility for rendez-vous and docking. In *36th Congress of the IAF*, pages 1–6, Stockholm, October, 7-12, 1985.

[3] Comin F. Six degree-of-freedom scanning supports and manipulators based on parallel robots. *Rev. Sci. Instrum.*, 66(2):1665–1667, February 1995.

[4] Gosselin C. and Angeles J. The optimum kinematic design of a spherical three-degree-of-freedom parallel manipulator. *J. of Mechanisms, Transmissions and Automation in Design*, 111(2):202–207, 1989.

[5] Gosselin C. and Hamel J.-F. The Agile Eye: A high performance three-degree-of-freedom camera-orienting device. In *IEEE Int. Conf. on Robotics and Automation*, pages 781–787, San Diego, May, 8-13, 1994.

[6] Han C-S., Hudgens J.C., Tesar D., and Traver A.E. Modeling, synthesis, analysis and design of high resolution micromanipulator to enhance robot accuracy. In *IEEE Int. Workshop on Intelligent Robot and Systems (IROS)*, pages 1153–1162, Osaka, November, 3-5, 1991.

[7] Khatib O. and Bowling A. Optimization of the inertial and acceleration characterics of manipulators. In *IEEE Int. Conf. on Robotics and Automation*, pages 2883–2889, Minneapolis, April, 24-26, 1996.

[8] Ma O. and Angeles J. Optimum architecture design of platform manipulator. In *ICAR*, pages 1131–1135, Pise, June, 19-22, 1991.

[9] Masory O. and Wang J. Workspace evaluation of Stewart platforms. In *22nd Biennial Mechanisms Conf.*, pages 337–346, Scottsdale, September, 13-16, 1992.

[10] Merlet J-P. Articular velocities of parallel manipulators, Part II: Finding all the robots with fixed extremal articular velocity for performing a fixed cartesian velocity over a whole workspace. In *IEEE Int. Conf. on Robotics and Automation*, pages 3262–3267, Albuquerque, April, 21-28, 1997.

[11] Merlet J-P. Designing a parallel robot for a specific workspace. Research Report 2527, INRIA, April 1995.

[12] Pittens K.H. and Podhorodeski R.P. A family of Stewart platforms with optimal dexterity. *J. of Robotic Systems*, 10(4):463–479, June 1993.

[13] Stoughton R. and Arai T. A modified Stewart platform manipulator with improved dexterity. *IEEE Trans. on Robotics and Automation*, 9(2):166–173, April 1993.

[14] Zanganeh K.E. and Angeles J. On the isotropic design of general six-degree-of-freedom parallel manipulators. In J-P. Merlet B. Ravani, editor, *Computational Kinematics*, pages 213–220. Kluwer, 1995.