Robotica (1997) volume 15, pp 367–373. Printed in the United Kingdom © 1997 Cambridge University Press

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

J-P. Merlet

INRIA Sophia-Antipolis, BP 93, 06902 Sophia-Antipolis Cedex (France)

SUMMARY

This paper presents a design methodology for parallel robots having to satisfy a set of performance constraints. Some of these constraints are used to compute a closed region in the parameters space (in which a point define an unique robot geometry) which define all the robot geometries fulfilling these constraints. Then a grid is created over this region and for each node of this grid the requirements, expressed in a high level language, are evaluated. The node leading to the robot best fulfilling the constraints is the design solution.

KEYWORDS: DEMOCRAT; Parallel architecture; Design.

1. INTRODUCTION

Parallel robot have been extensively studied this recent years and are now starting to appear as commercial products. One of the challenging problem in this field is to determine the robot design parameters such that the robot will be able to perform a given task in the best manner or in other words to determine the ''optimal'' robot with respect to the user's requirements.

The optimal design of parallel robot has drawn a lot of interest of researchers in the past.[1–10] Basically, the approach is the same in all of these works:

a. Reduction of the number of design parameters by appropriate assumptions
b. Utilization of a numerical optimization procedure for computing the parameters that minimize a cost-function. If F_i are the values of the features present in the user's requirement the cost function $\mathcal{C}$ is expressed as: $\mathcal{C} = \sum_i w_i F_i$ where the w_i are weights.

Step a 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 b is, in fact, difficult to implement:

- being given the user's requirements the expression of the cost function is difficult to find
- many features of parallel robots are antagonistic. Therefore the result will be deeply affected by the weight imposed on the features in the cost function
- the elements of 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 its nominal position. Consequently, both factors should be present in the cost function.
- 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 a considerable amount of computer time

The purpose of DEMOCRAT is to provide a solution to this problem for the Gough-type parallel robots. In an initial step the number of design parameters will be reduced. The design parameters are used to define a *parameters space* in which a point defines an unique robot geometry. Then DEMOCRAT will work along two steps:

a. Some of the design constraints are used to determine a closed region in the design parameters space which include all the possible robot geometries satisfying, at least partially, these constraints. This step will be called the *cutting* phase.
b. The above region is discritized (each node represents an unique robot geometry) and for each node the user's requirements are evaluated. These requirements are expressed through a procedure written in a high-level language. For each node this procedure is evaluated and return 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 (in which case it is stored as the only solution). This phase is called the *refining* phase.

2. Reduction of the 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.

https://doi.org/10.1017/S0263574797000453 Published online by Cambridge University Press

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):

- 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 the joint centers 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.

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. For example, the length of leg i, as the end-effector is moving along a given trajectory, is completely known as a function of the trajectory of the end-effector and the location of the A_i, B_i points. Thus for each leg we define a special plane, called the *design plane,* 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. Therefore under our assumptions a point in this plane represents an unique location of the point A_i, B_i and the parameters space is defined as the set of the 6 design planes. A robot geometry is defined by a set of 6 points in the 6 design planes: this set will be called the *representative set* of the robot. Each point in the set will be called the *representative point* of the robot.

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; the radius of circle on which lie the A_i points and the radius of the circle on which lie the B_i points.

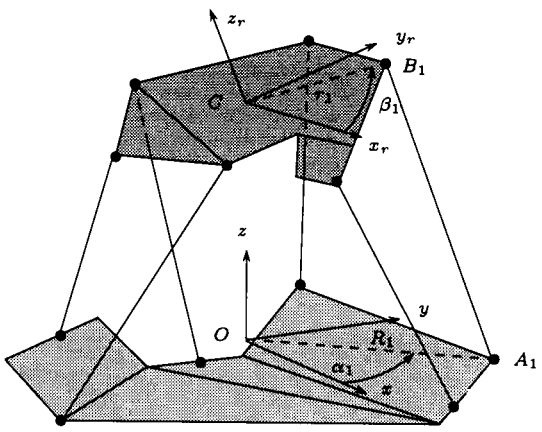

Fig. 1. The design parameters.

3. DEMOCRAT

After having reduced the number of design parameters we may now start explaining how DEMOCRAT is working. In the first step we will determine closed regions in the design planes such that all the robots satisfying some constraints must have their representative points inside the closed region.

3.1 The cutting phase

In DEMOCRAT two main features are used to compute the closed region in the design planes; then workspace requirements and the maximal articular velocities requirements.

3.1.1 **The design algorithm.** The algorithm design[11] enables one to deal with the workspace requirements. As input, it takes the minimal and maximal values of the leg lengths ρ_{min}, ρ_{max} and a set of segments, called the *segment trajectories,* describing a trajectory 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 constraint that for any position of the robot along the segment trajectories the leg lengths lie within $[\rho_{min}, \rho_{max}]$.

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 (considered with respect to the leg lengths constraints) will include all the segment trajectories.

Note that **design** accept optional inputs:

- mechanical limits on the passive joints at A_i: the representative points inside the closed regions in the design planes will be such that the leg length constraints are still satisfied together with the mechanical limits on the joints.
- legs interference may also be checked if there is only one design plane (the joint centers A_i lie on a circle with radius R_1 and the B_i lie on a circle with radius r_1). The legs are assumed to be reduced to the mathematical segment A_iB_i.

Let us summarize the theory underlying **design**:

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

Figure 2 shows an output of the **design** algorithm in the

https://doi.org/10.1017/S0263574797000453 Published online by Cambridge University Press

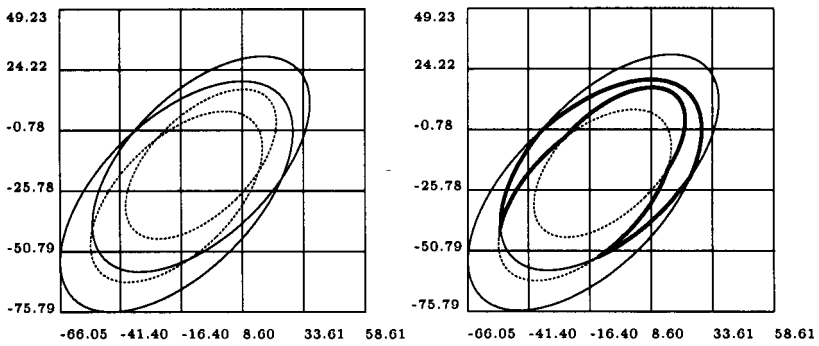

Fig. 2. An output of the **design** algorithm. All the robots such that the specified segment trajectory lie within their workspace have their representative point inside the region drawn in thick line on the right drawing.

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. On the left side of this figure the maximal ellipsis are drawn in thin lines while some ellipses of $\mathscr{E}_m$ are drawn in dashed lines. On the right side the region $\mathscr{R}$ is drawn in thick lines.

3.1.2 **The vitesse_design algorithm.** The algorithm **vitesse_design** 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 $\mathscr{C}$ in the design planes such that for any robot having its representative points inside the regions the velocity objective may be performed for any position of C on the specified segment trajectories with an articular velocity whose absolute values is always lower than $\dot{\rho}_l$.

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 $[0, 1]$. Indeed we may write:

$$\mathbf{OC} = \mathbf{OM_1} + \lambda \mathbf{M_1M_2}$$

The algorithm has the following features:

- on a segment trajectory the maximum of the square of the articular velocity is obtained either for $\lambda = 0, 1$ or for a value λ_n solution of a first order equation
- in the design planes 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 $\mathscr{R}_0$, $\mathscr{R}_1$, $\mathscr{R}_n$ be the regions where $\dot{\rho}^2 \leq \dot{\rho}_l^2$ respectively for $\lambda = 0, 1, \lambda_n$
- in the design plane the equations $\lambda_n = 0$ and $\lambda_n = 1$ define conics. These conics split the design plane into regions and let us denote $\mathscr{R}_{\backslash'}$, $\mathscr{R}_{\backslash\infty}$ the regions where respectively $\lambda_n \geq 0$ and $\lambda_n \leq 1$
- the output of the algorithm is therefore $\mathscr{C} = (\mathscr{R}_0 \cap \mathscr{R}_1) \cap (\mathscr{R}_n \cap \mathscr{R}_{\backslash'} \cap \mathscr{R}_{\backslash\infty})$

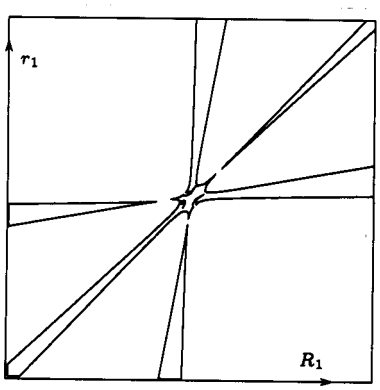

Fig. 3. An output of the **vitesse_design** algorithm: here we have only one design plane and we have one segment trajectory. The region with the border drawn in thick line is the output of the algorithm.

Figure 3 shows an output of the **vitesse_design** algorithm in the case where we have one design plane and one segment trajectory. The region delimited by the border drawn in thick lines is the output of the algorithm.

Evidently by computing 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.

3.2 The refining phase

Using the result of the previous sections we have drastically reduced the size of the search domain in the design planes. We may now consider other constraints for refining our result. A grid is created for the regions determined in the cutting phase. Each node of this grid represents an unique location for the pair (A_i, B_i).

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

3.2.1 **Efficient evaluation of the robot features.** As we have seen in the introduction some of these requirements are computer intensive as some features need to be determined for any position of the end-effector in a given volume. Therefore we have developed new algorithms enabling to compute efficiently some features. These algorithms are able to compute the features for any *translation workspace* i.e. for any position of C inside a given volume. This volume may be a box or any volume

https://doi.org/10.1017/S0263574797000453 Published online by Cambridge University Press

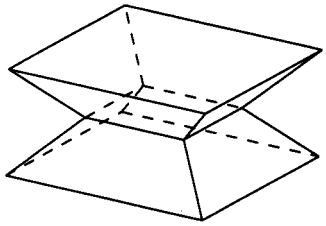

Fig. 4. An example of workspace volume which can be treated by the algorithms in DEMOCRAT.

defined by a set of cross-sections in the 3D space (Figure 4) 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 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).

3.2.2 **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:

%VO

= minimal stiffness in cube center 0 0 30, 0 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 VO.

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 5 shows the description of these requirements.

This program is evaluated for each node of the grid. At the very first call to this program the variable {best_stiffness}, which will contain the optimum value of the minimal stiffness is initialized to -1 (line 1–3). Then it will be tested if there is a singularity within the workspace cube (line 4–7). If there is a singularity the abort instruction is executed. This instruction tell DEMOCRAT that this robot does not fulfill the requirements. DEMOCRAT will therefore move to the next node and submit the new robot to the procedure. In line 8–10 we initialize some data. In line 12 we will compute the maximal positioning error along the x axis using a discretisation method whose step sizes are defined in line 11. In line 14 we test if this positioning error is better than 0.4: if not we execute the abort instruction at line 24. Otherwise we compute the minimal x stiffness of the robot (line 15) using the raideur procedure. If this stiffness is lower than the current value of {best_stiffness} (test at line 17) the abort instruction at line 24 is executed, otherwise we put the stiffness value in the variable {best_stiffness} (line 18) and execute the **save_R1_r1** instruction (line 20). This instruction tell DEMOCRAT that the current robot is optimal and its design parameters are to be saved in the result file.

4. IMPLEMENTATION OF DEMOCRAT

The current implementation of DEMOCRAT is written in Tcl/Tk. The designer may first execute the cutting phase: for example it may compute the closed region of the design planes corresponding to workspace requirements and visualize the resulting region. Then the region corresponding to velocity constraints may be computed. This region may be intersected with the workspace region to obtain the final search region. Then the designer may move to the refining 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.

5. ADVANTAGES AND DRAWBACKS OF DEMOCRAT

Clearly the most time consuming part of DEMOCRAT is the refining 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 cutting phase we 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

https://doi.org/10.1017/S0263574797000453 Published online by Cambridge University Press

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

4 %0= singularity in cube center 0 0 30 , 10 10 10
5 if ( %0 >0 ) bloc
6        abort
7        end_bloc

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

/* to compute the accuracy we use a discretisation
method, so we have to define the steps size */
11 step x 1 step y 1 step z 1
/* now find the maximal x-errors for the workspace */
12 %V0=maximal accuracy in cube center 0 0 30 , 10 10 10
13 %1=%V0[1][1]
/* if the x-error is lower than 0.4 we may consider the stiffness,
otherwise we just abort */
14 if (%1 <0.4 ) bloc
    /* minimal stiffness for any position in the workspace */
15    %V0=minimal stiffness in cube center 0 0 30 , 10 10 10
16    {current_stiffness}=%V0[1][1]
17        if ({current_stiffness} > {best_stiffness}) bloc
18            {best_stiffness}={current_stiffness}

20            save_R1_r1 /* save robot in result file */
21            quit
22        end_bloc
23 end_bloc
24 abort
```

Fig. 5. An example of user's requirement description.

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 advanages 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 cutting phase or the refining phase they can be easily included in DEMOCRAT

6. 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.

6.1 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 I.

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 the 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

Table I. Workspace, accuracy and stiffness requirements

x	y	z	θ_x	θ_y	θ_z
±30 mm	—	±20 mm	±5 mrad	±5 mrad	0–10 mrad
±0.01 mm	—	±0.1 mm	±0.1 mrad	±0.1 mrad	±0.05 mrad
+ +	— —	—	—	—	+ + +

https://doi.org/10.1017/S0263574797000453 Published online by Cambridge University Press

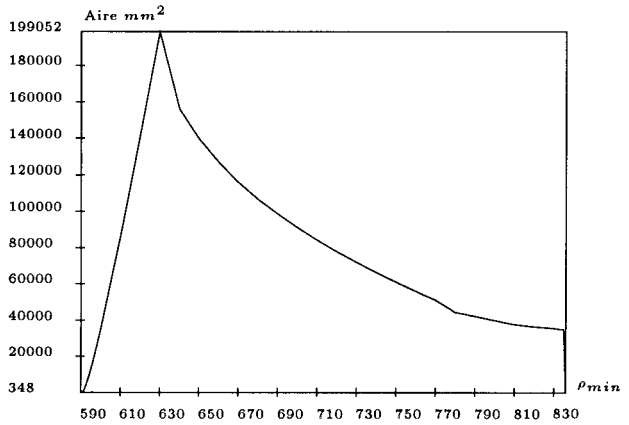

Fig. 6. Variation of the area of the allowed zone as a function of ρ_{min}.

computed the area of the search region in the design plane as a function of ρ_{min} (Figure 6).

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.

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 the x axis and for the rotation around the z axis had a sensor accuracy of 4 μm, leading to the worst case accuracy defined in Table II.

It may be seen that these 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. The maximum articular force was estimated to be at most 2000 N and it was determined that the ball-and-socket joint should enable a rotation of 6.27 degree.

6.2 Example

In this example the overall mass of the load and the bench vary from 500 kg to 1000 kg and has to be manipulated with an accuracy of the order of 1 to 10 μm. The result of the design process[12] is presented in Figure 7.

Table II. Maximal positioning error for a sensor error of 4 μm (mm, mrad)

Δ_x	Δ_y	Δ_z	Δ_{θ_x}	Δ_{θ_y}	Δ_{θ_z}
0.010000	0.009549	0.004870	0.009272	0.010488	0.011673

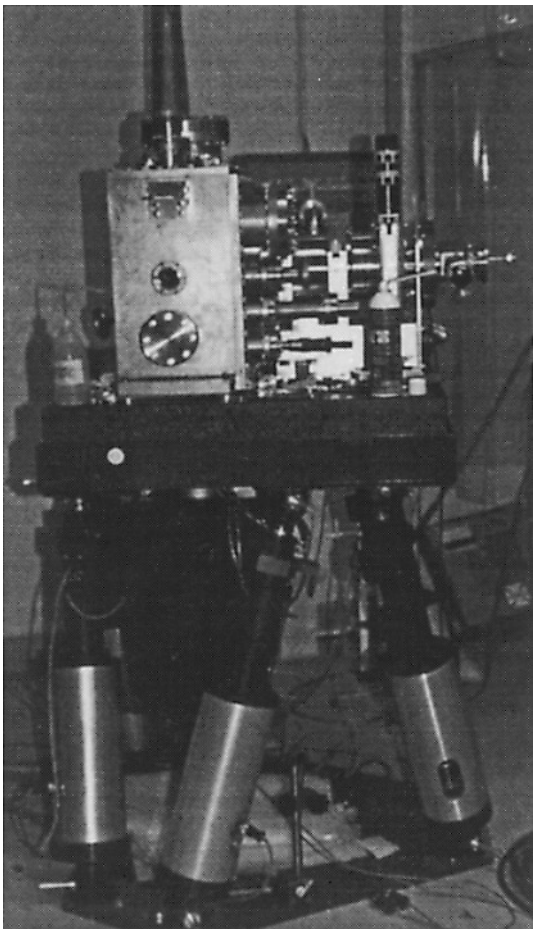

Fig. 7. The ESRF-INRIA fine positioning device.

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.

7. 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 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.

https://doi.org/10.1017/S0263574797000453 Published online by Cambridge University Press

References

1. B. Claudinon and J. Lievre, ''Test facility for rendez-vous and docking'' *36th Congress of the IAF* Stockholm (October, 7–12, 1985) pp. 1–6.
2. D. Douady, ''Contribution à la modélisation des robots parallèles: conception d'un nouveau robot à 3 liaisons et six degrés de liberté'' *PhD thesis* (Université Paris VI, Paris, December 9, 1991).
3. C. Gosselin and J. Angeles, ''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).
4. C. Gosselin and J.-F. Hamel, ''The Agile Eye: A high performance three-degree-of-freedom camera-orientating device'' *IEEE Int. Conf. on Robotics and Automation,* San Diego (May, 8–13, 1994) pp. 781–787.
5. C-S. Han, D. Tesar and A. Traver, ''The optimum design of a 6 dof fully parallel micromanipulator for enhanced robot accuracy'' *ASME Design Automation Conf.* Montréal (September, 17–20, 1989) pp. 357–363.
6. C-S. Han, J.C. Hudgens, D. Tesar and A.E. Traver, ''Modeling, synthesis, analysis and design of high resolution micromanipulator to enhance robot accuracy'' *IEEE Int. Workshop on Intelligent Robot and Systems* (*IROS*), Osaka (November, 3–5, 1991) pp. 1153–1162.
7. O. Ma and J. Angeles, ''Optimum architecture design of platform manipulator'' *ICAR,* Psia (June, 19–22, 1991) pp. 1131–1135.
8. O. Masory and J. Wang, ''Workspace evaluation of Stewart platforms'' *22nd Biennial Mechanisms Conf.* Scottsdale (September, 13–16, 1992) pp. 337–346.
9. F. Sternheim, ''Tridimensionnal computer simulation of a parallel robot. Results for the Delta 4 machine'' *18th Int. Symp. on Industrial Robot,* Lausanne (April, 26–28, 1988) pp. 333–340.
10. R. Stoughton and T. Arai, ''A modified Stewart platform manipulator with improved dexterity'' *IEEE Trans. on Robotics and Automation* **9**(2) 166–173 (April, 1993).
11. J-P. Merlet, ''Designing a parallel robot for a specific workspace'' *Research Report 2527* (INRIA, April, 1995).
12. F. Comin, ''Six degree-of-freedom scanning supports and manipulators based on parallel robots'' *Rev. Sci. Instrum.* **66**(2) 1665–1667 (February, 1995).

https://doi.org/10.1017/S0263574797000453 Published online by Cambridge University Press