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.

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

Abstract

This paper presents an algorithm for finding all the possible parallel robot geometries such that the end-effector may perform a given translation velocity for any location of the end-effector in a given workspace, the constraint being that the actuator velocities should be always lower than a given bound. All these possible geometries are defined by a region in a particular design parameters space. This algorithm should be useful for the optimal design of parallel robots.

1 Introduction

The design of parallel manipulators involves various objectives which are either to be optimized or to be reached: positioning workspace, positioning accuracy, maximal articular forces over a given workspace etc.. One of these objectives may be that the end-effector should be able to perform a given cartesian velocity for any of its position in a given workspace, the constraint being that the articular velocities should not exceed a given value. Numerous papers have been devoted to the relations between articular velocities and cartesian/angular velocity of the end-effector [1],[3],[4], [7],[8] but none, to the best of the author knowledge, have addressed the problem of determining all the robot geometries fulfilling a constraint on the articular velocities. In this paper we will consider the Gough type parallel manipulator [2] illustrated in figure 1.

In the sequel C will denote the center of the moving platform, O the origin of the reference frame, A_i, B_i respectively the centers of the base joints and platform joints of leg i. We will assume that the orientation of the moving platform is constant (and therefore the vector $\mathbf{CB_i}$ is constant). A vector whose coordinates is expressed in the moving frame will be denoted by an index $_r$.

We will assume that the velocity $\dot\rho_i$ of the linear

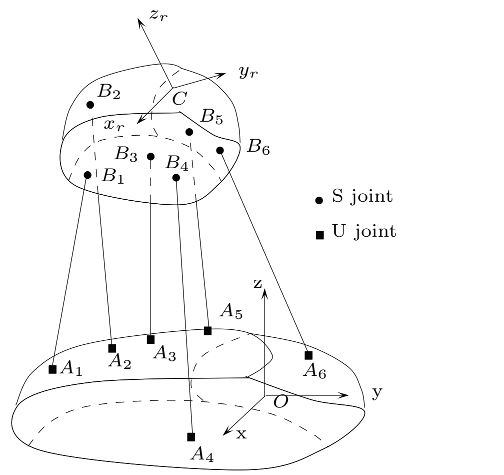

Figure 1: The classical Gough type parallel robot

actuator of leg i should satisfy the constraint:

$$-\dot\rho_{i_l} \le \dot\rho_i \le \dot\rho_{i_l} \tag{1}$$

The articular velocities $\dot\rho$ are related to the cartesian and angular velocities $\mathcal{V}$ of the end-effector by:

$$\dot\rho = J^{-1}(X)\mathcal{V} \tag{2}$$

where X is the pose of the robot and J^{-1} is its inverse jacobian matrix. For a given cartesian/angular velocity $\mathcal{V}$ the articular velocities are therefore pose dependent. The velocity of leg i may be written as:

$$\dot\rho_i = J_i^{-1}\mathcal{V}$$

where J_i^{-1} is the ith row of the inverse jacobian matrix. It is well known that this row can be written as:

$$\frac{\mathbf{A_iB_i}}{||\mathbf{A_iB_i}||} \quad , \quad \mathbf{CB_i} \times \frac{\mathbf{A_iB_i}}{||\mathbf{A_iB_i}||}$$

Note that for a given leg the articular velocity does not depend upon the velocity of the other legs: consequently we will drop the leg subscript in the sequel. As the angular velocity is equal to 0 we get:

$$\dot{\rho} = \frac{\mathbf{AB}.\mathbf{V_c}}{||\mathbf{AB}||} \tag{3}$$

where $\mathbf{V_c}$ is the cartesian velocity vector of the end-effector. Note that as $||\mathbf{AB}||$ is the length ρ of the leg, the articular velocity is equal to the norm of the cartesian velocity multiplied by the cosine of the angle between the cartesian velocity vector and the unit vector of the leg: hence the articular velocity will never exceed the norm of the cartesian velocity vector.

2 The design planes

Equation (3) shows that the articular velocity is dependent upon 6 design parameters: the 3 coordinates of A and the 3 coordinates of B. We will reduce the number of design parameters to 2 with the following assumptions:

- point A lie on a known line going through O the origin of the reference frame
- point B lie on a known line going through C.

Therefore the only remaining design parameters are the distances R_1 from O to A and r_1 the distance from B to C (figure 2). For each leg we define the

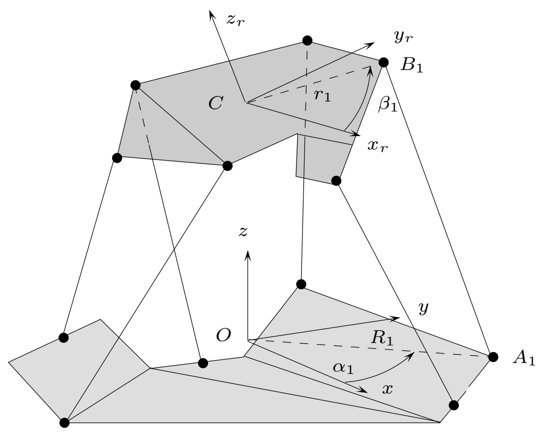

Figure 2: The design parameters

design plane as a plane with a frame such that the x, y coordinates of a point represent the values of the design parameters R_1, r_1. Consequently any point in this plane defines an unique location for A, B and we have 6 design planes, one for each leg. We have:

$$\mathbf{AO} = R_1 \left(\begin{array}{l} \cos(\alpha) \\ \sin(\alpha) \\ 0 \end{array} \right) = R_1 \mathbf{u}$$

$$\mathbf{CB_r} = r_1 \left(\begin{array}{l} \cos(\beta) \\ \sin(\beta) \\ 0 \end{array} \right) = r_1 \mathbf{v} \tag{4}$$

Our purpose is to determine all the robot geometries (i.e. the locations of the A, B points) such that for any position of C in a given workspace the articular velocities are within the range $[-\dot{\rho}_l, \dot{\rho}_l]$. This problem is equivalent to finding the region (denoted the *correct region*) of the design plane which define all the possible locations of the A, B points.

Note that the design plane we are considering is similar to the one we used in our design methodology `DEMOCRAT 1`. The idea behind `DEMOCRAT 1` is to determine first the correct region in the design plane for constraints on various features of parallel robots. For example we have presented in [6] an algorithm for finding the closed region of the design planes defining all the robot geometries such that their workspace include a specified workspace.

In the second step of `DEMOCRAT1` the intersection of all the correct regions (denoted the *design region*) will define all the robot geometries which satisfy all the constraints. In a third step a grid is created in the design region and for each node of this grid (which correspond to an unique robot geometry) we may compute the values of others features of the robot in order to determine the "optimal" robot geometry.

3 Maximal velocity for a segment

As stated previously we want to compute the correct region for a limit given on the maximal articular velocities when the end-effector lie inside a given workspace. We will assume here that this workspace is defined by a set of segments (called the *segment trajectory*) which describe the possible location of C and we will assume that for each segment trajectory given by the designer the orientation is kept constant all over the segment (but the designer may define the same segment with various orientations of the end-effector).

This way of describing a desired workspace is in fact not so restrictive. Let us assume that in fact the desired workspace is defined by a faceted object in 3D, each face being planar. As the orientation is constant, point B will lie inside a similar volume $\mathcal{R}$ obtained by translating the previous region by the vector $\mathbf{CB}$. Consider the line D going through A with vector $\mathbf{V_c}$ and a line D_1 going through A and a point M in $\mathcal{R}$. If α is the angle between these two lines the articular velocity for this location of the end-effector is $||\mathbf{V_c}|| \cos\alpha$. Fixing a maximal articular velocity $\dot{\rho}_l$ lower than $||\mathbf{V_c}||$ is therefore equivalent to determine the location of A, B such that the angle between the two lines is always greater than a given angle. The limit cases will always be obtained for an edge of $\mathcal{R}$ and consequently only the edges are important.

We will now expose the principle of the algorithm for determining the correct region in the design plane.

3.1 Principle of the algorithm

Let assume that point C is moving along a segment defined by two points M_1, M_2. Any position of C on the segment may be written as:

$$\mathbf{OC} = \mathbf{OM_1} + \lambda \mathbf{M_1M_2} \tag{5}$$

where λ is a scalar in the range [0,1]. The vector $\mathbf{AB}$ can be decomposed as $\mathbf{AB} = \mathbf{AO} + \mathbf{OC} + \mathbf{CB}$. Using equations (4,5) we get:

$$\mathbf{AB} = R_1\mathbf{u} + \mathbf{OM_1} + \lambda\mathbf{M_1M_2} + r_1\ R\mathbf{v} \tag{6}$$

where R is the rotation matrix defining the orientation of the end-effector. We may now write the constraint equation:

$$\frac{\mathbf{AB}.\mathbf{V_c}}{||\mathbf{AB}||} = \pm\dot{\rho}_l$$

As $||\mathbf{AB}||$ is the leg length ρ this equation may be transformed as:

$$\rho^2(\dot{\rho}_l)^2 - (\mathbf{AB}.\mathbf{V_c})^2 = 0 \tag{7}$$

Using equation (6) we get:

$$R_1^2(\dot{\rho}_l^2 - (\mathbf{u}.\mathbf{V_c})^2) + r_1^2(\dot{\rho}_l^2 - (\mathbf{v}.\mathbf{V_c})^2) + \\ 2R_1r_1(\dot{\rho}_l^2\mathbf{u}.\mathbf{v} - (\mathbf{u}.\mathbf{V_c})(\mathbf{v}.\mathbf{V_c})) + D = 0 \tag{8}$$

where D is only a function of λ. In the design plane this equation define a set of conics parameterized by λ. Let us denote $a = \mathbf{u}.\mathbf{V_c}$, $b = \mathbf{v}.\mathbf{V_c}$, $c = \mathbf{u}.\mathbf{v}$. The nature of the conics is given by the following relations:

- if $\Delta = (2cab - a^2 - b^2)/(c^2 - 1)$ is positive: the conics will be an ellipsis for $\dot{\rho}_l$ from 0 to $\sqrt{\Delta}$ excluded. For $\dot{\rho}_l = \sqrt{\Delta}$ the conics is a parabola and for $\dot{\rho}_l > \sqrt{\Delta}$ the conics is an hyperbole
- otherwise the conics is always an hyperbole or does not exist if $\dot{\rho}_l > ||\mathbf{V_c}||$.

For a given λ this conics will separate the design plane in regions and one of this region will be such that for any point inside the region the articular velocity is in the range $[-\dot{\rho}_l, \dot{\rho}_l]$. In fact it is not necessary to worry about the λ parameter: indeed the derivation of the articular velocity with respect to λ leads to an expression whose numerator is $a\lambda + b$. Therefore the articular velocity will be extremal either for $\lambda = 0, 1$ or for $\lambda = -b/a = \lambda_n$ if this value is in the range [0,1]. Consequently it is sufficient to check the inequalities on the articular velocity for these 3 values of λ.

Let C_0, C_1 be the conics obtained from (8) for $\lambda = 0, 1$. Each of these conics separate the plane in regions, one of which contain all the possible values of R_1, r_1 such that for the considered λ the absolute value of the articular velocity is lower than $\dot{\rho}_l$. These regions will be denoted Z_0 for $\lambda = 0$ and Z_1 for $\lambda = 1$.

Now by substituting $\lambda = \lambda_n$ in equation (8) we get a fourth order equation in R_1, r_1 which can be factorized in two second order equations. These equations define also two conics C_{n_1}, C_{n_2} which divide the plane into regions, and for each conic one of the region contains all the values of R_1, r_1 such that the absolute value of the articular velocity is greater than $\dot{\rho}_l$ for $\lambda = \lambda_n$. These two regions will be denoted Z_{n_1}, Z_{n_2}. Note that these regions define forbidden position in the design plane only if λ_n is in the range [0,1]. We have:

$$\lambda_n = \frac{aR_1^2 + br_1^2 + cR_1r_1 + dR_1 + er_1 + f}{uR_1 + vr_1 + w} = \frac{n}{d}$$

If $u = v = w = 0$, then λ_n does not exist and so does Z_{n_1}, Z_{n2}. Otherwise $n = 0$ define a conic C_{n0} which split the plane in two regions $Z_{n>0}, Z_{n<0}$ in which $n > 0$ and $n < 0$. Similarly $d = 0$ split the plane in two half-planes $Z_{d>0}, Z_{d<0}$ for which $d > 0$ and $d < 0$. Figures 3,4 shows examples of the conics involved in the algorithm, for one segment and one trajectory.

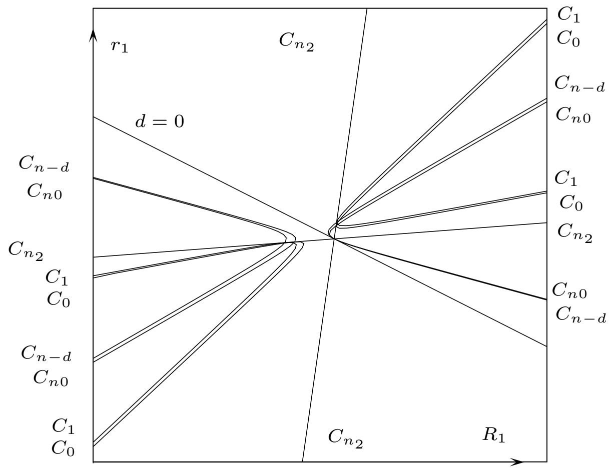

Figure 3: Examples of conics involved in the algorithm. Here $C_0, C_1, C_{n0}, C_{n-d}$ are hyperbola, C_{n_1} does not exist and C_{n_2} is a special case of hyperbola (two intersecting lines).

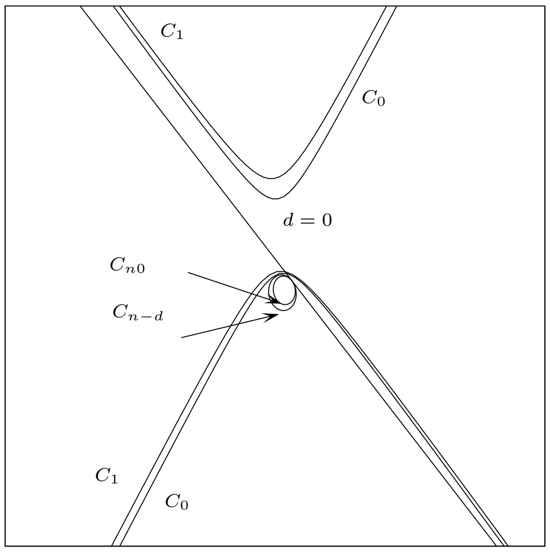

Figure 4: C_0, C_1 are hyperbola, C_{n0}, C_{n-d} are ellipsis, C_{n_1} and C_{n_2} does not exist.

The region $Z_{\lambda_n>0}$ for which $\lambda_n \geq 0$ is therefore obtained as:

$$Z_{\lambda_n>0} = (Z_{d>0} \cap Z_{n>0}) \cup (Z_{d<0} \cap Z_{n<0})$$

Similarly the conic C_{n1} obtained for $n - d = 0$ split the plane into region and we define the region $Z_{\lambda_n<1}$ for which $\lambda_n \leq 1$. This region is therefore obtained as:

$$Z_{\lambda_n<1} = (Z_{d>0} \cap Z_{n-d<0}) \cup (Z_{d<0} \cap Z_{n-d>0})$$

Finally the region Z_n in which λ_n is in the range [0,1] is obtained as the intersection of $Z_{\lambda_n>0}$ and $Z_{\lambda_n<1}$.

A valid point in the design plane should belong to Z_0 and Z_1 (this ensure that the absolute value of the articular velocity is lower than the bound for $\lambda = 0, 1$) but should not belong to both Z_{n_1} and Z_n and both to Z_{n_2} and Z_n (otherwise the articular velocity is greater than the bound for $\lambda = \lambda_n$). Consequently the valid region Z in the design plane is obtained as:

$$Z = (Z_0 \cap Z_1) - ((Z_{n_1} \cap Z_n) \cup (Z_{n_2} \cap Z_n))$$

3.2 Practical implementation

In our practical implementation we assume that the design plane is bounded by a rectangle which is either defined by the user or is the bounding box of the correct region determined for the workspace constraint.

Every part of the conics which is involved in the calculation is approximated by a polygonal line. Consequently the operations on the regions are reduced to boolean operations on polygons (intersection and subtraction). The accuracy of this approximation can be modified by changing the number of points of the polygonal lines which are used to approximate the conics.

The data of the example presented at the top of figure 3 have been used to compute the correct region, represented in figure 5. The computation time for the

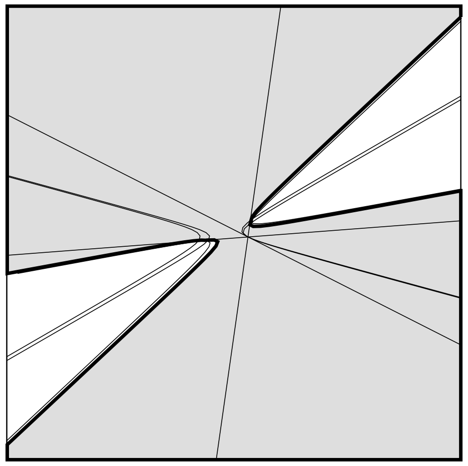

Figure 5: The correct region is the gray region

example presented in figure 5 was 2.6 seconds on a SUN-4 workstation (a more clever implementation of the algorithm may reduce drastically this time).

Clearly this algorithm enable to check a set of segment trajectory: the correct region is computed for each segment in the set, then the intersection of all the correct regions leads to the final correct region. Figure 6 present an example of such case. The computation time for the example presented in figure 6 was 9.21 seconds on a SUN-4 workstation. Note that the correct region are defined in each of the 6 design planes. But if the joint centers on both the base and the platform lie on a circle then we have an unique design plane. In that case the correct region is computed for each leg and for each segment trajectory, then the intersection of all the correct regions leads to the desired result. Figure 7 shows the conics involved in the algorithm for all the links in the example of figure 3 and figure 8 shows the final correct region.

The computation time for this example is 59.89s on a SUN 4 workstation.

We may also combine this procedure with the result obtained with our algorithm which deal with the workspace constraint in the same design plane [6]. The correct region for the velocity constraints is computed, then the correct region is intersected with the correct region due to the workspace constraint. Figure 9

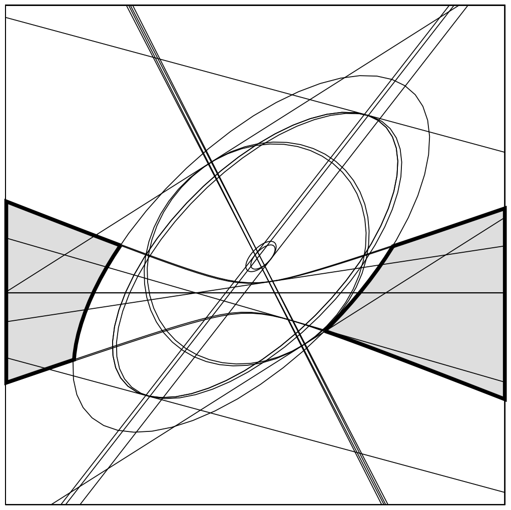

Figure 6: The correct region is the gray area while the conics used in the algorithm are drawn in thin lines

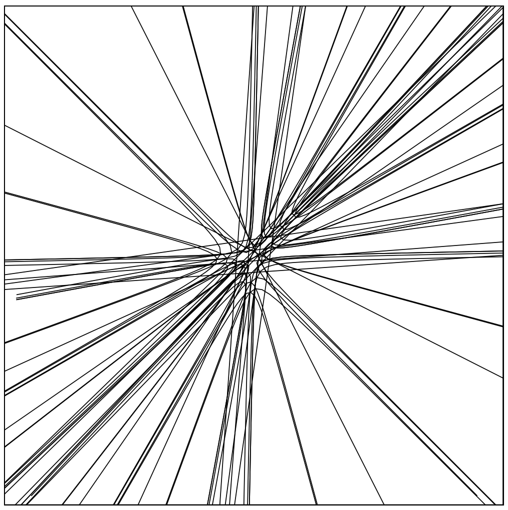

Figure 7: The conics involved in the algorithm when the A, B points lie on circles.

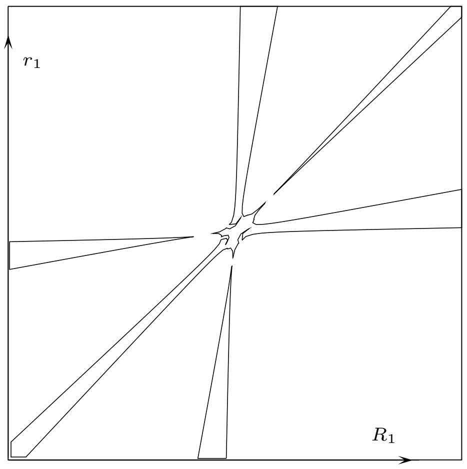

Figure 8: The correct region

shows an example of such computation. The resulting

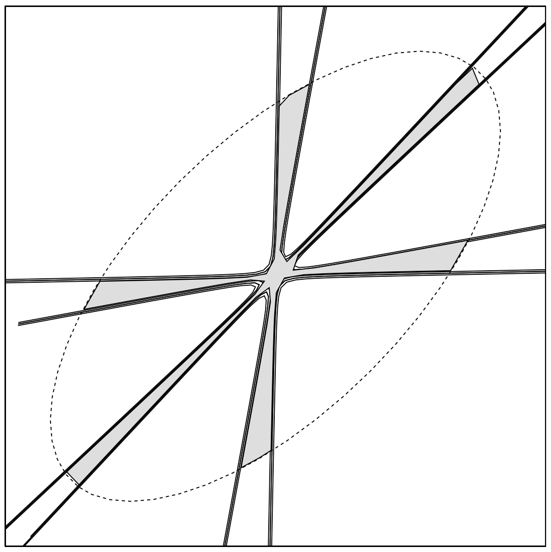

Figure 9: The zone in dashed line represent the correct region due to workspace constraints. The conics involved in the algorithm are shown in thin line and the final correct region is drawn in gray.

region defines all the robot geometries which have a workspace including a desired workspace together with fulfilling the constraint on the articular velocities.

Note that it is possible to verify the correctness of the result for any point inside the correct region by using an algorithm we have developed to compute the extremal values of the articular velocity for a given cartesian/angular velocity whatever is the position of

C inside a given volume. It is indeed possible to show that we can compute the extremum of the articular velocities with an arbitrary accuracy if the workspace is a box. Any other workspace type is split in as many boxes as necessary until the desired accuracy on the extremum of the articular velocities is reached.

4 Another utility of the algorithm

Let assume that the cartesian velocity is defined as a unit vector $\mathbf{V}$. The algorithm will therefore compute the region of the parameter space such that the absolute value of the quantity $\mathbf{AB}.\mathbf{V}/||\mathbf{AB}||$ does not exceed a given value. But this quantity is the cosine of the angle between the link direction and the vector $\mathbf{V}$. Consequently we are able to determine all the robots such that the angle of the passive joints with any fixed direction does not exceed a given value, which is useful for the design of the passive joints.

5 Extension to other types of parallel robots

The algorithm has been presented for the Gough-type parallel robot but may be extended for other types of parallel robots. Indeed it is well known that most of parallel robots have an inverse jacobian matrix of the same form as the one of the Gough-type robot. Therefore the principle of the algorithm will be similar.

Consider for example the parallel robots with fixed leg lengths but whose A_i points moves on a line with unit vector $\mathbf{u_i}$. The velocity $\dot{\gamma_i}$ of point A_i is related to the cartesian and velocities and can be established as [5]:

$$\dot{\gamma_i} = \frac{\mathbf{A_iB_i}.\mathbf{V}}{\mathbf{u_i}.\mathbf{A_iB_i}} \tag{9}$$

If C moves on a segment the derivative of the articular velocity with respect to λ is constant. Hence the minimal and maximal articular velocities will be obtained either for $\lambda = 0$ or $\lambda = 1$.

6 Conclusion

We have presented an algorithm for computing all the possible parallel robot geometries such that the absolute value of the articular velocities does not exceed a given value when the end-effector of the manipulator performs various cartesian velocities, whatever is the position of the end-effector in a given volume.

This algorithm will be integrated in our design methodology `DEMOCRAT` for the conception of parallel manipulator in the near future.

References

[1] Gosselin C. *Kinematic analysis optimization and programming of parallel robotic manipulators.* Ph.D. Thesis, McGill University, Montréal, June, 15, 1988.

[2] 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.

[3] Ling S-H. and Huang M.Z. Kinestatic analysis of general parallel manipulators. In *ASME Mechanisms Design Conf.*, Minneapolis, September, 14-16, 1994.

[4] Martinez J.M.R. and Duffy J. A simple method for the velocity and acceleration analysis of in-parallel platforms. In *9th World Congress on the Theory of Machines and Mechanisms*, pages 842–846, Milan, August 30- September 2, 1995.

[5] Merlet J-P. *Les Robots parallèles.* Hermès, Paris, 1990.

[6] Merlet J-P. Workspace-oriented methodology for designing a parallel manipulator. In *IEEE Int. Conf. on Robotics and Automation*, pages 3726–3731, Minneapolis, April, 24-26, 1996.

[7] Sorli M. and others . Mechanics of Turin parallel robot. In *9th World Congress on the Theory of Machines and Mechanisms*, pages 1880–1885, Milan, August 30- September 2, 1995.

[8] Zanganeh K.E. and Angeles J. Instantaneous kinematics and design of a novel redundant parallel manipulator. In *IEEE Int. Conf. on Robotics and Automation*, pages 3043–3048, San Diego, May, 8-13, 1994.