Determination of the workspace of a 3PRPR parallel mechanism for human-robot collaboration

Alexandre Lecours[1], Clément M. Gosselin[2]

[1] *Département de Génie Mécanique, Université Laval, alexandre.lecours.1@ulaval.ca*

[2] *Département de Génie Mécanique, Université Laval, clement.gosselin@gmc.ulaval.ca*

Abstract

This paper presents an algorithm, based on a geometric approach, to obtain the workspace of a parallel manipulator. For several steps in the process, a short review of different methods is presented, along with the presentation of the preferred algorithm. This work was performed in the context of the design of a robot for human-robot collaboration, which is also presented as an example.

Keywords: Workspace, parallel robot, human-robot collaboration.

Détermination de l'espace de travail d'un mécanisme parallèle 3PRPR pour la collaboration humain-robot

Résumé

Cet article présente un algorithme, basé sur une approche géométrique, pour obtenir l'espace de travail d'un robot parallèle. Pour plusieurs étapes du processus, une courte revue des différentes méthodes possibles est présentée et la solution la plus efficace est identifiée. Ce travail a été développé dans le contexte de la conception d'un robot de collaboration humain-robot, qui sera aussi présenté à titre d'exemple.

Mots-clé: Espace de travail, robot parallèle, collaboration humain-robot.

1 INTRODUCTION

The determination of the workspace of parallel mechanisms has been an important research topic for several years (see for instance [1, 2, 3, 8, 13] and several others). Indeed, it is well known that the closed-loop kinematic chains used in parallel mechanisms restrict the range of motion of the platform and therefore considerably reduce the workspace of parallel mechanisms. Therefore, the determination of their workspace is of great importance for their design.

The algorithms developed in the literature for the determination of the workspace of parallel mechanisms can be essentially classified in three different categories, namely $i)$ geometric methods (see for instance [2, 3]), $ii)$ numerical methods (see for instance [8, 13]) and $iii)$ discretization methods (see for instance [1]).

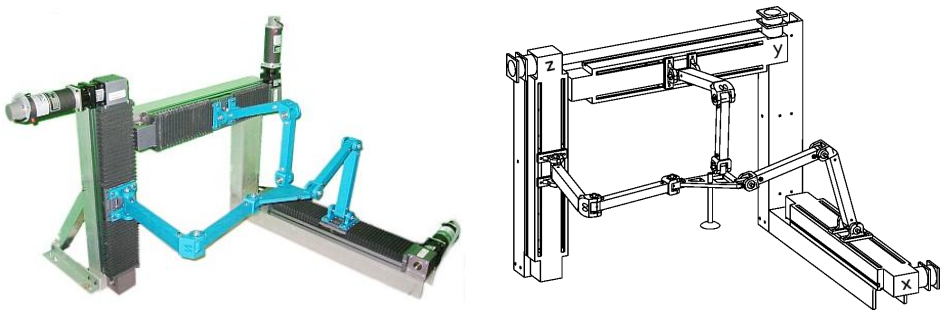
This paper presents an algorithm for the determination of the workspace of a parallel robot which is being designed for human-robot collaboration (HRC) in the handling of light payloads. The design of the robot is based on the Tripteron, a Cartesian 3PRRR parallel robot. The main advantage of using a parallel robot is the large payload to robot weight ratio and as it will be seen, the Tripteron also has other advantages. In this application, it is required that the workspace to footprint ratio be as large as possible. Therefore, the determination of the workspace is a critical issue. An algorithm to obtain the workspace of a parallel manipulator is presented here. As mentioned above, this subject has already been discussed by many authors, but this paper presents a synthesis and some simplifications as well as some ideas for new methods. The geometric design of the Tripteron is first recalled, a modified design for the HRC application is then presented, followed by the workspace determination algorithm and an example.

2 TRIPTERON DESIGN

The Tripteron, shown in figure 1(a), consists of three legs of the PRRR type orthogonally attached to a common platform. For each leg, the direction of the P (prismatic) joint and the axes of the R (revolute) joints are all parallel [6]. There are several advantages in using the Tripteron for human-robot collaboration. First, there is a large payload to robot weight ratio and the actuators are attached to the structure. Also, the movement is fully decoupled (the Jacobian is the identity matrix in all configurations) so it is more intuitive, easier to control, its dexterity and kinematic performances remain constant all over the workspace (no singularities) [6] and gravity acts on a single axis which makes it easier to statically balance the mechanism. Due to the design, the end-effector always maintains a constant orientation. The workspace could be complex [9] but under certain conditions regarding the arm lengths, the workspace could simply be a rectangular prism [6], which makes the interaction with the robot even more intuitive.

3 HRC ROBOT DESIGN

The geometric design of the Tripteron robot is not optimal for HRC applications and hence some variations were studied. The chosen geometry differs from the Tripteron on two main points: (1) the bottom rail has been moved upward to be attached with the other two so that it would be easier to fix to a structure and less cumbersome, (2) the rails have been placed such that the x and z rails intersect the y rail at its center point as shown in figure 2. In order to obtain a geometrically simple workspace, the maximal length of the legs must be greater than the total



(a) Tripteron prototype. (b) Tripteron CAD model.

Figure 1: Tripteron.

travel of the actuator as shown in figure 6. As shown in figure 3, by placing the vertical (Z) rail in the middle of the rail, the length of the arm required for this rail is smaller thereby leading to smaller deflections and stresses and thus smaller inertia and mass of the links are required. This is not the case with all the arms, but one should note that the vertical rail's arm is critical because gravity acts along this direction. It should also be noted that for a large workspace, the RR links following the P joint are cumbersome and need to be very resistant (increased inertia and mass). For a better performance, they have been replaced by a RP-equivalent mechanism so one arm can be represented as a PRPR chain. This modification does not affect the robot's kinematic properties [6]. The chosen RP-equivalent mechanism, represented in figure 4, consists of an accordion-like mechanism. The number of sections influences the deflection, stresses and workspace (angular limits). For each arm, two mechanisms are linked together in order to increase the stiffness. The final mechanism is represented in figure 2 where the lines joining the rails to the effector are RP-equivalent mechanisms represented in figure 4. The mechanisms are here shown separately for a better clarity of the figures.

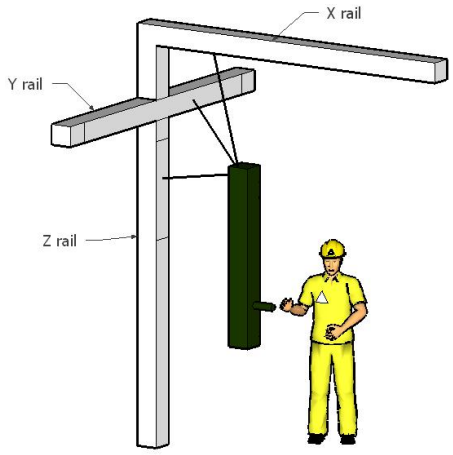


Figure 2: HRC Robot design.

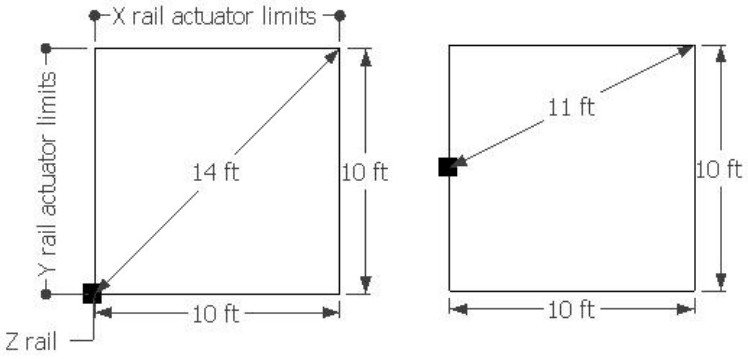


Figure 3: Link lengths.

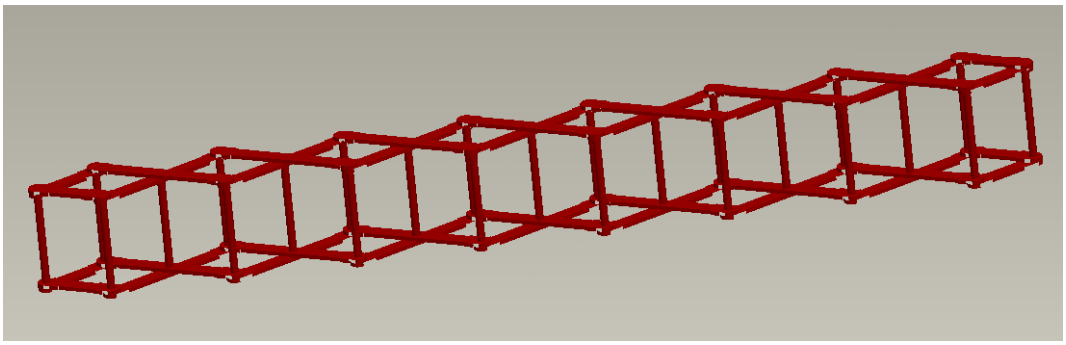


Figure 4: RP-equivalent mechanism.

4 WORKSPACE

4.1 General concept

The required length of the arms can be obtained by a simple geometric analysis using figure 3. However, these arm lengths may not lead to the best solution because the workspace is not the only parameter in the design. If the links are too long, it increases the stresses, the deflection due to bending, the inertia and the weight and it may not lead to the best solution. Thus, for a given link length, the workspace must be computed to see if it is acceptable, referring to its simplicity. To obtain the workspace, three main approaches are possible. The first one consists in discretizing the space, i.e., evaluate if a given point can be reached by solving the inverse kinematic problem or by finding some reachable points using the direct kinematic problem. If the procedure is repeated for a set of points, the total workspace is obtained. The major disadvantages of such techniques is that they are not computationally efficient and that the given representation is more difficult to understand since it consists of a cloud of points. Alternatively, the robot may be drawn using a commercial CAD system. This method would be used for a final representation, but it would be harder to optimize the workspace [2]. The other approach, that is implemented, is based on the geometric approach proposed in [3]. The main idea is that the workspace of the robot is the intersection of the workspace of each of its arms, also referred to as the vertex spaces. In this case, the workspace is the intersection of three perpendicular hollow cylinders. The external radius of a cylinder is the length of the fully extended arm while the internal radius is the minimum distance that can be reached by the arm. In the case of a RR arm, the external radius would be the sum of the link lengths and the internal radius the difference. For our RP arm, the radii are the minimum and maximum limits of the prismatic joints. Finally, the length of the cylinder is the travel of the actuator.

4.2 Inverse algorithm

The purpose of the inverse algorithm is to compute the arm lengths required to have a workspace bounded by a rectangular prism. Even if these lengths may not be used in the final design, their determination provides an approximation of the required link lengths. The maximal and minimal lengths have to be considered because they both affect the workspace. As mentioned and referring

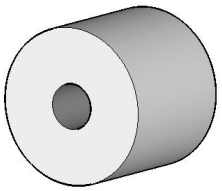


Figure 5: Hollow cylinder vertex space.

to figure 6, the maximal length must be greater than the outer limits of the actuator and the minimal length must be smaller than the distance to the nearest actuator limit (otherwise there will be a hole in the workspace). Thus, giving the rails positions and the end effector offsets, one can compute the required minimal and maximal arm lengths. If the length conditions are not met, the workspace will not be a rectangular prism and it must be computed with the direct algorithm that follows. In figure 6, the rectangle dimensions are the two other actuators travel as shown in figure 3 for the Z arm.

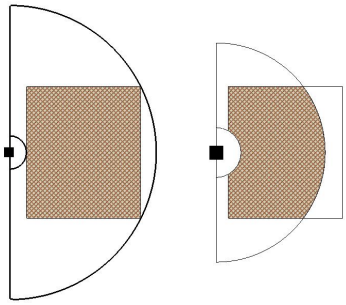


Figure 6: Workspace: Sufficient and insufficient arm length.

4.3 Direct algorithm

The rectangular prism workspace, obtained by the actuator limits, may not lead to the best solution. Given the rail positions, the arm lengths, the end effector offset and the angular limits on the arms movement, one can compute the workspace in order to optimize the solution. As mentioned earlier, the geometric approach consists in finding the intersections of the workspace of each arm. It is possible to analytically determine the intersections, but it is difficult to obtain a good representation. An easier way is to take a planar section of the workspace by performing a discretization, for example in the vertical direction. Thus, instead of searching the intersection of 3D objects to find the workspace directly, the program will determine the intersections of 2D objects, in order to find 2D workspace for several planar sections. By combining these 2D workspaces, the final 3D workspace will be obtained. In this case the vertical (Z) rail arm workspace is a vertical hollow cylinder and hence the projection onto the XY plane is a hollow disk. Because the other arm workspaces are perpendicular to the projection plane, their projections are rectangles. Note that in the latter case, if we consider the minimal length, we have hollow cylinders and then the projections can become

two rectangles separated by an empty space. Thus, in order to find the robot workspace, we have to find, for each vertical increment, the intersection of a hollow disk with rectangles.

The steps in obtaining the workspace are as follows:

1. Define the base parameters (rail positions, actuator limits, effector offset, angular limits and arm minimum and maximum lengths).

   for each step in Z

2. Define the arm workspace section.

3. Compute the intersections between all of the segments.

4. Verify if the segments are part of all arm workspaces.

5. Compute the area.

   end step in Z

6. Compute the volume.

7. Draw a representation.

4.3.1 Arm workspace section

There are two types of objects appearing in workspace sections: rectangles and circles. Thus, there are two types of segments: lines and circles (or arcs of circles by considering joint limits). In order to compute the 2D workspace, we need to define the segments. For lines, start point, end point, type of element (line or circle) and arm number are needed. For circles, radius and angular limits are also needed. Note that instead of circle, an ellipse could be considered because a circle is a type of ellipse and thus the solution would be more general. We consider circles (or ellipses) as entities distinct from lines (a circle can be represented by a series of lines) in order to ensure the accuracy and computational efficiency of the algorithm. Finally, as in figure 7, a number is attributed to each segment in order to keep track of the procedure.

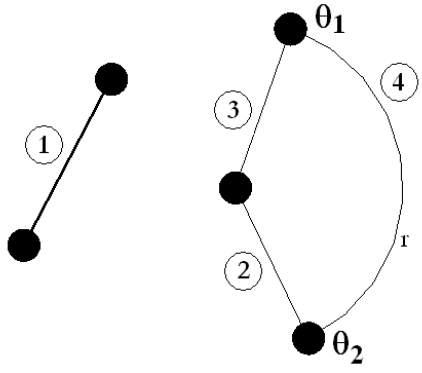


Figure 7: Arm workspace section procedure.

4.3.2 Intersections

The objective of the next two sections is to determine which segments are part of the workspace. First, all the intersections between all of the segments are computed to obtain a list of points. First of all, the end points of the base elements are intersection points. By taking them directly, there is no need to compute the intersections between the elements of a same body. Then, the intersections between the other elements are computed. Basically, there are intersections between two lines, a line and a circle and two circles. For each intersection, two points are created (that is one for each intersecting segment) with the point location and arm and segment number as information. The point location is redundant but the arm and segment number are not. The list of intersection points are then grouped by segments and ordered for each segment. For a segment, the double points are ignored and all of the points are ordered. There is a special note for complete circles because they have the same start and end point so the end point must not be ignored. For each segment, the list of points is transformed into a series of lines attached to each other. For example, if there are three intersection points (A,B,C) on a segment, segments A-B, and B-C are created. Figure 8 represents a summary of the intersection procedure.

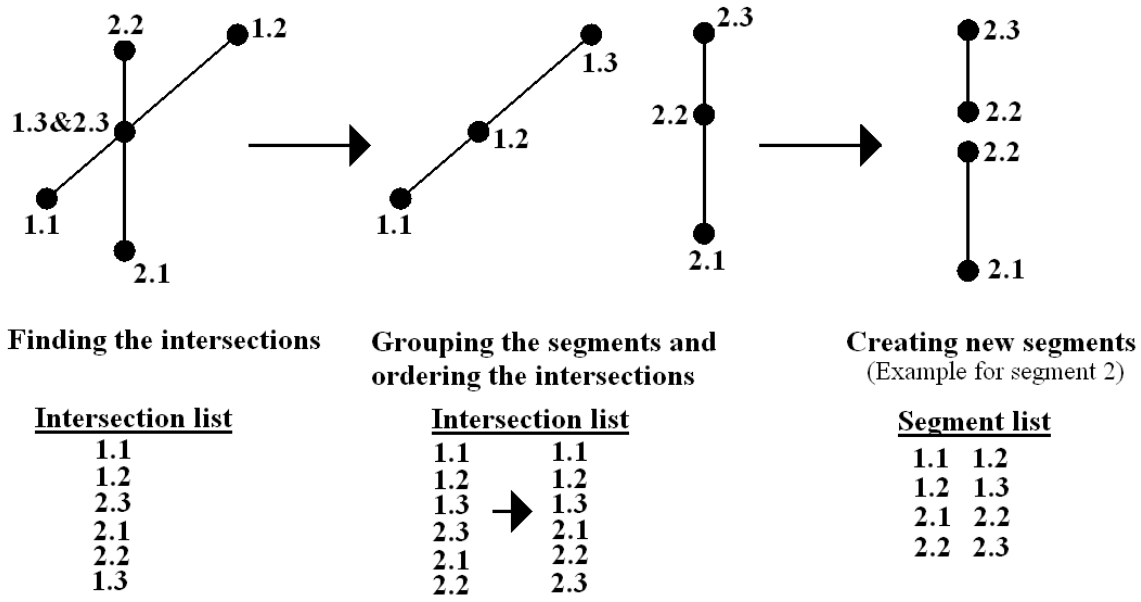
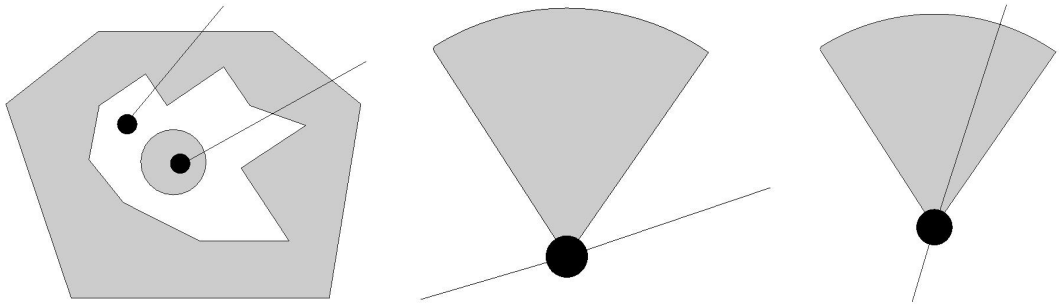


Figure 8: Intersections procedure.

4.3.3 Verifying if the segments are part of the workspace boundary

The segments obtained in the preceding section from the base segments must be tested in order to determine if they are part of the workspace boundary. This is accomplished by verifying if the segment is part of all the vertex spaces. It is clear that a segment is part of its own arm vertex space so it is only needed to verify if it is in the other two arm workspaces. Because of the segmentation performed in the preceding section, it is clear that if a point on a segment is in a workspace, the entire segment is necessarily in this workspace. Thus, to test if a segment is in the workspace,

it suffices to test if a point on that segment is inside this workspace. Here, the middle point is considered for simplicity. Note that the start and the end point must not be taken for this test because if the segment is outside the workspace and the test point is at the limit of it, the test would return a positive answer although it should return a negative one. The best general way to know if a point is inside or outside a workspace is to draw a line for this point toward "infinity" [10]. If the number of segments that are crossed is even, the point is not in the workspace while if it is odd, it is inside. An example is provided in figure 9(a). A particular case can arise if this test line crosses an intersection as in figure 9(b)(c).



(a) Test example. (b) Particular case 1. (c) Particular case 2.

Figure 9: Workspace verification.

In case (b), two intersections are counted and in (c) three, for the same situation which is that the point is outside the workspace. We could count the point only one time but in this case, count would be one in (b) and two in (c). In [10], it is proposed to cancel the result and to start again with a new test line or to completely ignore this case as it should not happen. It is clear that the best method is to change the line direction in order for the algorithm to work in all situations. For very complex situations, it could happen that any test line would encounter this problem, for example, if the surrounding limits are made of several small lines. It is for this reason and for other numerical issue that it would not be a good idea to discretize a circle into a group of small lines. Finally, if the segment is inside all of the workspace, its parameters are saved in a vector. Other ways to test a segment were explored but they require more information and only work for simple cases.

4.3.4 Computing the area and volume

In order to compute the volume of the workspace, the surface area must be computed for each section obtained for a given vertical step (Z increment). Three main methods could be used in order to do this: (1) an analytical integral (2) a numerical integral (3) use the Gauss Divergence Theorem. Two first methods require a geometric interpretation of the segments in order to know which one is below which one, which may be difficult. The suggested method is to use the Gauss Divergence Theorem applied to planar regions. Thus, there is no need to place the segments in order but it does require a minimal knowledge of the workspace section to know the direction of the normal vector to the curve. For a given segment, one has

$$S_i = \frac{1}{2} \int_{\partial\Omega_i} \mathbf{s}^T \mathbf{n} d\partial\Omega_i \tag{1}$$

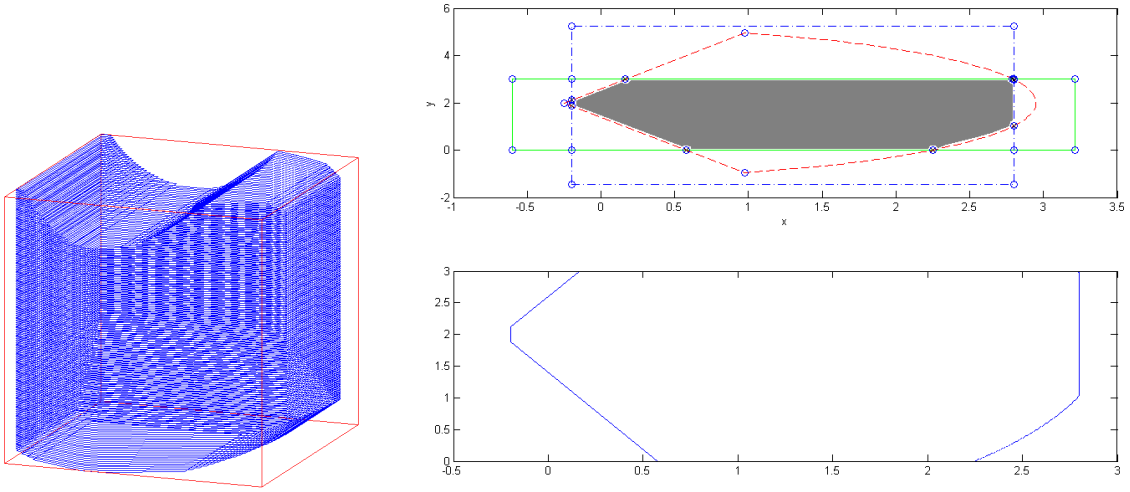
where $\partial\Omega_i$ is the boundary of the planar region associated with segment i, **s** is the position vector of a point of $\partial\Omega_i$ and **n** is the outward unit normal vector to the curve $\partial\Omega_i$. The area of the workspace is then obtained by adding the S_i computed for each segment [4], [5].

$$S = \sum_{i=1}^{n} S_i \tag{2}$$

Finally, to compute the volume, one must only add all areas obtained for the sections and multiply by the discretization step in Z.

4.3.5 Drawing a representation

Finally, for each vertical step, all lines that are in the 2D workspace are drawn and figure 10(a) is obtained. Options can be added to the visualization such as the limits to verify the results, offsets, rails representation and a 2D workspace intersection and representation, for a given Z step, as shown in figure 10(b). To obtain a better representation, the process could be repeated with another 2D discretization. Thus, by combining the two representations, a grid would be obtained, which is easier to visualize, but harder to program as the algorithm needs to be implemented two times.



(a) 3D Workspace.

(b) 2D workspace. *Solid lines represent the Y arm workspace, dashed lines the Z arm workspace, dash-dot lines the X arm workspace and the dark region the total workspace, also represented in the bottom part of the figure.*

Figure 10: Workspace representation.

4.3.6 Workspace optimization

Given the characteristics of the robot, it is now possible to obtain the workspace. However, it would be very useful to include the workspace algorithm in a loop (varying the robot parameters) in order to optimize the solution regarding the workspace, deflections, stresses and the robot dynamic properties. The optimal solution would be chosen using a given criterion.

5 CONCLUSIONS

First, the Tripteron and the modified design for a human-robot collaboration robot were presented. Then, some methods to obtain the workspace were reviewed and a geometric approach was presented and implemented. The workspace algorithm, the determination of a 2D section of the workspace and a method to determine if a segment is in the workspace were explained. Finally, the computation of the volume and the representation of the total workspace were discussed. Future work on this subject will consist in including this algorithm in an optimization loop in order to automatically find the best solution regarding the workspace, deflections, stresses and the robot dynamic properties.

6 ACKNOWLEDGMENTS

This work was supported by The Natural Sciences and Engineering Research Council of Canada (NSERC) as well as by the Canada Research Chair Program and General Motors (GM) of Canada.

REFERENCES

[1] Bonev, I.A. and Ryu, J., 2001, A new approach to orientation workspace analysis of 6-DOF parallel manipulators, Mechanism and Machine Theory, Vol. 36, No. 1, pp. 15–28.

[2] Bonev, I., 2002, Geometric analysis of parallel mechanisms, Ph.D. Thesis, Département de Génie Mécanique, Université Laval, Québec.

[3] Gosselin, C.M., 1990, Determination of the Workspace of 6-DOF Parallel Manipulators, ASME Journal of Mechanical Design, Vol.112, No.3, pp.331-336.

[4] Gosselin, C.M. and Guillot, M., 1991, The Synthesis of Manipulators with Prescribed Workspace, Journal of Mechanical Design, Vol.113, pp.451-455.

[5] Gosselin, C.M. and Jean, M., 1996, Determination of the workspace of planar parallel manipulators with joint limits, Journal of Robotics and Autonomous Systems, Vol.17, No.3, pp.129-138.

[6] Gosselin, C.M., Kong, X., Foucault, S. and Bonev, I., 2004, A fully-decoupled 3-DOF Translational Parallel Mechanism, Comptes-Rendus de la 4th Chemnitz Parallel Kinematics Seminar / 2004 Parallel Kinematic Machines International Conference, Chemnitz, Germany, 20–21 Avril, pp. 595-610.

[7] Gosselin, C.M., Masouleh, M.T., Duchaine, V., Richard, P., Foucault, S. and Kong, X., 2007, Parallel Mechanisms of the Multipteron Family: Kinematic Architectures and Benchmarking, 2007 IEEE International Conference on Robotics and Automation, 10-14 April 2007, pp.555-560.

[8] Haug, E.J., Luh, C.-M., Adkins, F.-A. and Wang, J.-Y., 1996, Numerical algorithms for mapping boundaries of manipulator workspaces, Journal of Mechanisms, Vol. 118, p. 228.

[9] Kong, X. and Gosselin, C., 2002, Kinematics and Singularity Analysis of a Novel Type of 3-CRR 3-DOF Translational Parallel Manipulator, The International Journal of Robotics Research, Vol. 21, No. 9, 791-798.

[10] Merlet, J.-P., 1992, Manipulateures parallèles, 5e partie: Détermination de l'espace de travail à orientation constante, INRIA, No.1645.

[11] Merlet, J.-P., 1997, Designing a Parallel Manipulator for a Specific Workspace, The International Journal of Robotics Research, Vol. 16, No. 4, pp. 545–556.

[12] Merlet, J.-P., 1999, Determination of 6D Workspaces of Gough-Type Parallel Manipulator and Comparison between Different Geometries, The International Journal of Robotics Research, Vol. 18, No. 9, pp. 902–916.

[13] Snyman, J.A., du Plessis, L.J. and Duffy, J., 2000, An Optimization Approach to the Determination of the Boundaries of Manipulator Workspaces, ASME Journal of Mechanical Design, Volume 122, No 4, pp. 447–456.