AiMT Advances in Military Technology
Vol. 10, No. 1, June 2015

Mechanical and Computational Design for Control of a 6-PUS Parallel Robot-based Laser Cutting Machine

R. Zavala-Yoé*, R. Ramírez-Mendoza and J. Ruiz-García

Tecnológico de Monterrey, Escuela de Ingeniería y Ciencias, Calle del Puente 222, Ejidos de Huipulco, 14380 Mexico City, Mexico.

The manuscript was received on 1 December 2014 and was accepted after revision for publication on 29 June 2015.

Abstract:

A 6-PUS parallel robot is modelled, designed and controlled in terms of Newton – Euler equations in order to be implemented numerically. Direct and inverse kinematics as well as direct and inverse dynamics are analysed and solved. Direct kinematics is solved by introducing a novel numeric-geometric method here referred to as method of arcs. Next, direct and inverse dynamics problems are solved offering advantages traditional methods do not. Two types of controllers were implemented in order to get a desired performance. Two 3D robot designs are also shown. Numerical computations and simulations were developed in MATLAB. The whole design and control converges to a laser cutting machine application which is given at the end of the document.

Keywords:

Parallel robot, Modelling, Control, Laser applications.

1. Introduction.

Laser processes are widely used in civil and non-civil industries. Marking and cutting with lasers are typical applications of this. Since parallel robots have high payloads and are mechanically robust [9, 10], it is proposed here to design one of these robots for this effect. Current parallel robots come from the Stewart-Gough platform which was designed in the seventies and has been subject of numerous studies. However, little has been investigated about the different variants of its kinematic chain, in this case the 6-PUS robot. Of the few who have studied this robot, most have done it in a very general way [1, 5, 7, 10, 11]. One advantage that this model (6-PUS) has over the traditional 6-UPS is that the actuators remain fixed on the base. In some cases, this simplifies the

* *Corresponding author: Tecnológico de Monterrey, Campus Ciudad de México, Puente 222, Ejidos de Huipulco. México 14380. Phone: +52 555 54832020-1448, fax: +52 555 5483202020, E-mail: rzavalay@itesm.mx.*

construction of the robot and, on the other hand, causes part of the payload to be supported by the reaction forces of the floor. Thus, the actuators require less load capacity. In order to solve the direct kinematics problem, a novel numerical method is presented. This method is based on a simplified geometry of the robot in such a way that each pair of limbs forms a triangle. The locus of the top vertex of this triangle is an arc. The algorithm (method of arcs) is presented, as well as some results of its implementation. Inverse kinematics problem is included too. On the other hand, inverse dynamics is deduced proposing an improvement in a design given in [11]. Next, direct dynamics is solved. Finally, the controlled robot performance is illustrated by examples of a robot-based laser cutting machine.

2. Inverse Kinematics

As it is well known, a parallel robot consists of a fixed base, a set of limbs and an end effector referred to as the platform. Recall also that parallel robot's nomenclature is based in the types of joints which constitute the mechanism. Thus, 6-PUS means that our manipulator has six limbs, each of them consisting of a prismatic (P) plus a universal (U) plus spherical (S) joints. In robot kinematics, there are two main issues to deal with: direct or forward kinematics and inverse kinematics. Finding the position and orientation of the platform given the position/length of the actuators, leads to solve the direct kinematics problem.

The inverse kinematics problem consists of obtaining the position/length of the actuators (also known as the configuration of the robot) given a desired position and orientation of the platform. For parallel manipulators, such as the 6-PUS robot, the difficulty to obtain the inverse and direct kinematics is inverted with respect to serial robots [10], [9]. Given a certain position and orientation $\boldsymbol{p}$ of the end effector, it is relatively simple to calculate the length of each articulation with a simple sum of vectors. Fig. 1 and equations 1 and 2 give an example of this. In Fig. 1, vector $\boldsymbol{p}$ represents the desired position of the end effector. Vector $\boldsymbol{a}_i$ is a known parameter of the robot. The magnitude of vector $\boldsymbol{b}_i$ is also a parameter of the robot, and its orientation is given by the desired orientation of the platform; $\boldsymbol{d}_i$ is the vector going from A_i to D_i. By referencing all this vectors to the same coordinate system A_i, we can find the magnitude of each vector $\boldsymbol{r}_i$ as $\boldsymbol{r_i} = {}^{Ai}\boldsymbol{p} + {}^{Ai}\boldsymbol{b}_i - {}^{Ai}\boldsymbol{a}_i - \boldsymbol{d}_i$. where $\boldsymbol{p} = [p_x \; p_y \; p_z \; \psi \; \theta \; \phi]^T$ and the magnitude of $\boldsymbol{r}_i$ represents the position/length of the actuator [9, 10]. Recall that ψ, θ, and ϕ are the Euler angles [10]. Working out the latter equation yields the closed form of the inverse kinematics for this manipulator:

$$d_{ij} = \left(p_j + b_{ij} - a_{ij}\right) \pm \sqrt{\left(p_j + b_{ij} - a_{ij}\right)^2 - \left(|\boldsymbol{p}|^2 + |\boldsymbol{b}_i|^2 + |\boldsymbol{a}_i|^2 - r^2 + 2\boldsymbol{p}^T\left(\boldsymbol{b}_i - \boldsymbol{a}_i\right) - 2\boldsymbol{b}_i{}^T\boldsymbol{a}_i\right)} \quad (1)$$

$$i = 1,\ldots 6; \quad j = \{x, y, z\}.$$

However, given the free nature of the universal and spherical joints, if the position/length of the actuators is known, it is not easy to calculate the platform's position. Many researchers have studied the direct kinematics of the Stewart-Gough platform. Since the analytic solution requires solving a 16th degree equation, most of the articles that talk about this subject mention iterative numeric methods to solve the problem. It has been shown that because of the freedom possessed by the joints, given the length of the actuators it is possible to find 40 positions of the platform that satisfy equation (1) [3, 4, 10, 11]. Nevertheless, most of the studies refer to the 6-UPS platform, and very few [9, 10] analyse its variant 6-PUS. Compare with [14].

3. Singularity Conditions

While kinematic analysis gives us a relationship between the position of the platform and the length of the actuators, sometimes it is important to know the relationship between the velocity of the actuated limbs and the angular velocity of the platform, especially while studying the dynamics of the manipulator. The Jacobian matrix gives us that relationship, and that is why it is important to analyse it. Consider Fig. 1.

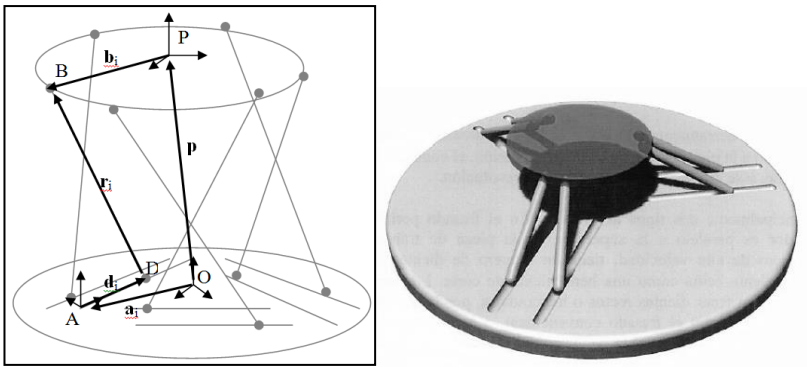

Fig. 1 a) Geometry of a general 6-PUS manipulator. b) First CAD design.

The output velocity vector is given by $\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{v}_p & \boldsymbol{\omega}_b \end{bmatrix}^T$, where $\boldsymbol{v}_p$ is the point P velocity and $\boldsymbol{\omega}_b$ is the angular velocity of vector $\boldsymbol{b}_i$. The closed loop kinematic equation is given by

$$\overline{OP} + \overline{PB} = \overline{OA} + \overline{AD} + \overline{DB}\,. \tag{2}$$

Taking time derivative of the latter equation yields

$$\boldsymbol{v}_p + \boldsymbol{\omega}_b \times \boldsymbol{b}_i = 0 + \dot{\boldsymbol{d}}_i + \boldsymbol{\omega}_r \times \boldsymbol{r}_i\,. \tag{3}$$

Defining a unit vector $\hat{\boldsymbol{d}}_i$ in the same direction as $\boldsymbol{d}_i$, a unit vector $\boldsymbol{s}_i$ in the same direction as $\boldsymbol{r}_i$ and taking dot product in the latter equation with that $\boldsymbol{s}_i$ yields:

$$\boldsymbol{s}_i \cdot \boldsymbol{v}_p + (\boldsymbol{b}_i \times \boldsymbol{s}_i) \cdot \boldsymbol{\omega}_b = \dot{\boldsymbol{d}}_i \left(\boldsymbol{s}_i \cdot \hat{\boldsymbol{d}}_i \right). \tag{4}$$

Rewriting this equation for $I = 1, \ldots, 6$ in terms of matrices produces

$$\begin{bmatrix} \boldsymbol{s}_1^T & (\boldsymbol{b}_1 \times \boldsymbol{s}_1)^T \\ \boldsymbol{s}_2^T & (\boldsymbol{b}_2 \times \boldsymbol{s}_2)^T \\ \vdots & \vdots \\ \boldsymbol{s}_6^T & (\boldsymbol{b}_6 \times \boldsymbol{s}_6)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_p \\ \boldsymbol{\omega}_b \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_1 \cdot \hat{\boldsymbol{d}}_1 & 0 & 0 & \cdots & 0 \\ 0 & \boldsymbol{s}_2 \cdot \hat{\boldsymbol{d}}_2 & 0 & \cdots & 0 \\ 0 & 0 & \boldsymbol{s}_3 \cdot \hat{\boldsymbol{d}}_3 & \cdots & 0 \\ 0 & 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \boldsymbol{s}_6 \cdot \hat{\boldsymbol{d}}_6 \end{bmatrix} \begin{bmatrix} \dot{d}_1 \\ \dot{d}_2 \\ \dot{d}_3 \\ \dot{d}_4 \\ \dot{d}_5 \\ \dot{d}_6 \end{bmatrix}. \tag{5}$$

The latter equation implies that a parallel robot will present two kinds of singular configurations which will arise when the Jacobian matrices, shown in equation (6), become singular. Direct kinematics singularities come from $\det(\boldsymbol{J}_x) = 0$ and inverse kinematics singularities come from $\det(\boldsymbol{J}_q) = 0$.

$$\boldsymbol{J}_x = \begin{bmatrix} \boldsymbol{s}_1{}^T & (\boldsymbol{b}_1 \times \boldsymbol{s}_1)^T \\ \boldsymbol{s}_2{}^T & (\boldsymbol{b}_2 \times \boldsymbol{s}_2)^T \\ \vdots & \vdots \\ \boldsymbol{s}_6{}^T & (\boldsymbol{b}_6 \times \boldsymbol{s}_6)^T \end{bmatrix}, \qquad \boldsymbol{J}_q = \begin{bmatrix} \boldsymbol{s}_1 \cdot \hat{\boldsymbol{d}}_1 & 0 & 0 & \cdots & 0 \\ 0 & \boldsymbol{s}_2 \cdot \hat{\boldsymbol{d}}_2 & 0 & \cdots & 0 \\ 0 & 0 & \boldsymbol{s}_3 \cdot \hat{\boldsymbol{d}}_3 & \cdots & 0 \\ 0 & 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \boldsymbol{s}_6 \cdot \hat{\boldsymbol{d}}_6 \end{bmatrix}. \quad (6)$$

An algorithm to compute numerically the constraints described above is given next.

Algorithm 1. Direct and Inverse Kinematics Singularities.
Step 1. Fix *xmin, deltax, xmax; ymin,deltay,ymax; zmin,deltaz,zmax.*
Step 2. For *x=xmin:deltax:xmax*.
Step 3. For *y=ymin:deltay:ymax.*
Step 4. For *z=zmin:deltaz:zmax.*
Step 5. Compute vectors $\boldsymbol{b}$ according to the actual position $\boldsymbol{b} = f(x,y,z)$.
Step 6. Compute inverse kinematics $\boldsymbol{d} = f(x,y,z,\boldsymbol{b})$ with equation (1).
Step 7. Determine vectors $\boldsymbol{s}$ for that position, $\boldsymbol{s} = f(\boldsymbol{d},x,y,z)$.
Step 8. Obtain $\boldsymbol{J}_x = f(\boldsymbol{b},\boldsymbol{s})$ with equation (6).
Step 9. Calculate and store $|\det(\boldsymbol{J}_x)|$ for that point. Store these points in p_{Jx}.
Step 10. Obtain $\boldsymbol{J}_q = f(\boldsymbol{d},\boldsymbol{s})$ with equation (6).
Step 11. Calculate and store $|\det(\boldsymbol{J}_q)|$ for that point. Store these points in p_{Jq}.
Step 12. End *z*, End *y*, End *x*.
Step 13. Plot the points $p_{Jx}(x,y,z)$ obtained in step 9 in gradient colours.
Step 14. Ídem for $p_{Jq}(x,y,z)$ obtained in step 11.

The graphs and interpretations are given in section 8.

4. Forward Kinematics: Numerical Algorithm.

As mentioned above, when we find the position and orientation of the platform given the position/length of the actuators, we are solving the direct or forward kinematics problem. The first thing that must be done is to establish the different coordinate systems that will be used: *O* is at the centre of the base, *P* at the centre of the platform, A_i at the beginning of each actuator, with the *x* axis pointing in the direction of the movement. D_i and B_i are the ends of the *i*th leg, being D_i the points that moves along the rail and B_i the point fixed on the platform. The method proposed in this paper assumes that the platform is a triangle in which there are only three connection points for the legs, one in each vertex, so that each point has two legs connected to it (see Fig. 2). This architecture is known as "6-3". The general idea of the proposed method is that using the corresponding transformation matrices, three arcs corresponding to the possible locations of the three vertices of the triangle are generated.

So, the distances between each point of each arc and each point of the other arcs are calculated. After calculating all distances, we must find pairs of points whose distance is equal to the side of the triangle. See algorithm 1 which ends up with a numerical solution (section 8) of the direct kinematics for this proposed design [13].

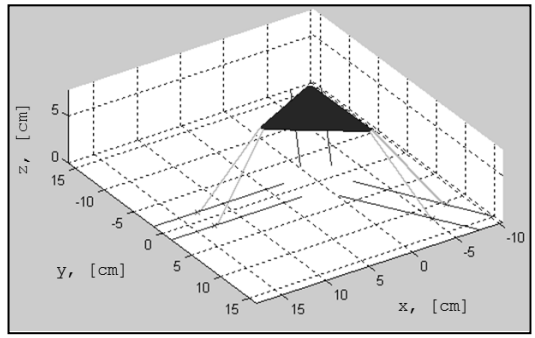

Fig. 2 Modified design with triangular base and end effector.

Algorithm 2. Direct kinematics (method of arcs)
Step 1 Calculate the six transformation matrices ${}^{o}\boldsymbol{H}_{\mathrm{Ai}}$ [10],that transforms a coordinate from system $\boldsymbol{A}_i$ to system $\boldsymbol{O}$.
Step 2 Given the desired lengths of the actuators, find the points corresponding to the centre of the triangle formed by each pair of legs.
Step 3 For each of the three triangles, find its height and the angle rho between the axis of the two actuators.
Step 4 Find and plot all the points of that arc.
Step 5 Find, save and plot the distance between each point of arc 1 and each point of arc 2.
Step 6 Repeat last step using arcs 1 and 3, repeat last step using arcs 2 and 3.
Step 7 Find and save each pair of points (1, 2) whose distance is equal to the length of the platform's.
Step 8 Repeat last step using arcs 1 and 3, repeat last step using arcs 2 and 3.
Step 9 Find the pairs of points (1, 3) whose point 1 exists on the list generated during step 8
Step 10 Find the pairs of points (2, 3) that exist on the list generated during step 11.
Step 11 Save and plot the three points and the solution (platform).

5. Inverse and Direct Dynamics

For the dynamic model, there are two different problems to solve. The direct dynamics derives the acceleration produced in the end effector, due to a given set of actuator forces. The inverse dynamics derives the forces needed in the actuators for a desired acceleration trajectory in the end effector. Little has been studied on the dynamics of parallel robots, and even less on that of the 6-PUS, [8-10, 12, 16]. In general, for parallel robots, the inverse dynamics is not very complex, but the direct dynamics is difficult to obtain due to possible existence of unknown reaction forces in the passive joints.

5.1 Inverse Dynamics

Consider Fig. 3. Let G be the platform's centre of mass. The platform has a mass of m and inertia matrix $\boldsymbol{I}$. The acceleration of point G is denoted by a_G. Let C be the point of application of external force $\boldsymbol{F}$ and moment $\boldsymbol{M}$. The force $\boldsymbol{f_i}$ which acts on point B_i can be decomposed on two components. The force $\boldsymbol{f_{si}}$ that goes along the main axis of the limb,

along unit vector $\boldsymbol{s}_i$ and the force perpendicular to $\boldsymbol{s}_i$, due to inertia, which will be denoted $\boldsymbol{f}_{Ni}$.

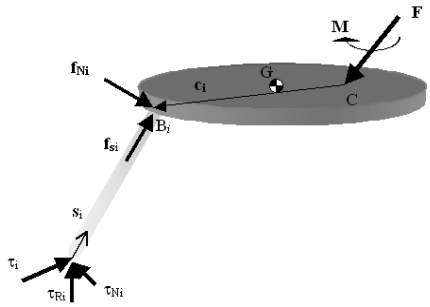

Fig. 3 Forces exerted on the end-effector.

We then have

$$\boldsymbol{f}_i = \boldsymbol{f}_{si} + \boldsymbol{f}_{Ni}. \tag{7}$$

In [9] a general dynamic model for the 6-PUS robot is developed, based on the dynamics of the 6-UPS model and considering that

$$\boldsymbol{f}_{si} = \tau_i \left(\boldsymbol{s}_i \cdot \hat{\boldsymbol{d}}_i \right) \boldsymbol{s}_i \tag{8}$$

where $\hat{\boldsymbol{d}}_i$ and $\boldsymbol{s}_i$ are the unit vectors pointing along the axes of the actuators and the limbs, respectively. However, this model does not consider the reaction forces exerted by the base and the actuators' restrictions. The consequence of this omission is that every time the limbs are in a vertical position, the effect of the actuator forces on the platform becomes zero (note the dot product in the equation), and since these are the only forces considered in the model, the platform would *fall down*. To prove this, the model used in [9] was implemented in an algorithm which applies a constant force, and a simulation was run (see section of Numerical Simulations and [13], [10]). Now, in order to deduce the inverse dynamics of the robot we proceed as follows. First, we consider only the force exerted by the limbs on the platform according to (8). Let $\boldsymbol{F}_N$ (3×1) be the resultant force of all $\boldsymbol{f}_{Ni}$, i = 1, …, 6 and $\boldsymbol{M}_N$ its resultant moment around point *C*. Let vector $\boldsymbol{c}_i$ be the vector that goes from point *C* to point *B*. Let $\boldsymbol{f}_{si} = f_{si}\boldsymbol{s}_i$ be the force that each limb exerts on the platform. The equations of equilibrium for the system are:

$$\boldsymbol{F} = \sum_{i=1}^{6} f_{si}\boldsymbol{s}_i + \boldsymbol{F}_N, \tag{9}$$

$$\boldsymbol{M} = \sum_{i=1}^{6} f_{si}\left(\boldsymbol{c}_i \times \boldsymbol{s}_i\right) + \boldsymbol{M}_N. \tag{10}$$

Let $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}_N$ the screw vectors for the external force and the resultant of the limbs, respectively.

$$\boldsymbol{\sigma} = \begin{bmatrix} \boldsymbol{F}_{(3\times1)} \\ \boldsymbol{M}_{(3\times1)} \end{bmatrix}_{(6\times1)}, \quad \boldsymbol{\sigma}_N = \begin{bmatrix} \boldsymbol{F}_{N\,(3\times1)} \\ \boldsymbol{M}_{N\,(3\times1)} \end{bmatrix}_{(6\times1)}. \tag{11}$$

Rewriting (9) and (10) in matrix form we have:

$$\begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{M} \end{bmatrix} = \sum \begin{bmatrix} \boldsymbol{s}_i \\ \boldsymbol{c}_i \times \boldsymbol{s}_i \end{bmatrix} f_{si} + \begin{bmatrix} \boldsymbol{F}_N \\ \boldsymbol{M}_N \end{bmatrix} . \tag{12}$$

We see that the first term of the right side of (12) is directly related with the Jacobian matrix $\boldsymbol{J}_x$ so that if we define a vector $\boldsymbol{f_s} = [f_{s1}, f_{s2}, \ldots, f_{s6}]^{\mathrm{T}}$ we can rewrite equation (12) as:

$$\boldsymbol{\sigma} = \boldsymbol{J}_x{}^T \boldsymbol{f}_s + \boldsymbol{\sigma}_N \tag{13}$$

where $\boldsymbol{f_s}$ is the 6×1 vector formed by the magnitude of each one of the forces that the limbs exert on the platform. Moment $\boldsymbol{M}_G$ applied on the platform with respect to point G is

$$\boldsymbol{M}_G = \boldsymbol{M} + \boldsymbol{e} \times \boldsymbol{F} \tag{14}$$

where $\boldsymbol{e}$ is the vector that goes from G to C; that is the moment arm of $\boldsymbol{F}$. Let $\boldsymbol{a}_G$ be the linear acceleration vector of point G. The Newton – Euler equations for the platform can be written as follows:

$$\boldsymbol{F} + m\boldsymbol{g} = m\boldsymbol{a}_G \ , \tag{15}$$

$$\boldsymbol{M}_G = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} \tag{16}$$

where $\boldsymbol{g}$ is the gravity acceleration vector and ω is the angular velocity of the platform. Developing algebraically the latter equations and defining $\boldsymbol{e}^*$ we obtain an expression for $\boldsymbol{\sigma}$, for any vector $\boldsymbol{x}$, as follows:

$$\boldsymbol{e} \times \boldsymbol{x} = (\boldsymbol{e}^*)\boldsymbol{x}, \text{ that is } \boldsymbol{e}^* = \begin{bmatrix} 0 & -e_3 & -e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix}, \tag{17}$$

$$\boldsymbol{\sigma} = \boldsymbol{G}\dot{\boldsymbol{W}} + \boldsymbol{H} \ , \tag{18}$$

$$\boldsymbol{G} = \begin{bmatrix} m & m\boldsymbol{e} \\ -m(\boldsymbol{e}^*) & \boldsymbol{I} - m(\boldsymbol{e}^*)^2 \end{bmatrix}, \ \boldsymbol{H} = \begin{bmatrix} m(\boldsymbol{v} - \boldsymbol{g}) \\ \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} + m(\boldsymbol{e}^*)(\boldsymbol{v} - \boldsymbol{g}) \end{bmatrix} . \tag{19}$$

These matrices have a physical meaning: $\boldsymbol{G}$ corresponds to the inertia matrix and $\boldsymbol{H}$ is the vector of forces due to gravity. By balancing (18) and (13) we have:

$$\boldsymbol{G}\dot{\boldsymbol{W}} + \boldsymbol{H} = \boldsymbol{J}_x{}^T \boldsymbol{f}_s + \boldsymbol{\sigma}_N \ . \tag{20}$$

With this we have derived the relation between the derivative of the twist over C (which corresponds to the acceleration) and the forces exerted on the platform by the limbs. Working out this equation we can find new expression for $\boldsymbol{\sigma_N}$, which in matrix form is expressed as:

$$\boldsymbol{\sigma}_N = \boldsymbol{T}\dot{\boldsymbol{W}} + \boldsymbol{U} \ , \tag{21}$$

$$\boldsymbol{T} = \begin{bmatrix} \sum_{i=1}^{6} \frac{I_i}{r_i{}^2} (\boldsymbol{s}_i^*)^2 \boldsymbol{K}_i \\ \sum_{i=1}^{6} \frac{I_i}{r_i{}^2} (\boldsymbol{c}_i^*)(\boldsymbol{s}_i^*)^2 \boldsymbol{K}_i \end{bmatrix}, \ \boldsymbol{U} = \begin{bmatrix} \sum_{i=1}^{6} \frac{I_i}{r_i{}^2} (\boldsymbol{s}_i^*)^2 \boldsymbol{L}_i \\ \sum_{i=1}^{6} \frac{I_i}{r_i{}^2} (\boldsymbol{c}_i^*)(\boldsymbol{s}_i^*)^2 \boldsymbol{L}_i \end{bmatrix} . \tag{22}$$

Substituting (22) in (21) yields and solving for $\boldsymbol{f_s}$, which corresponds to the forces that the limbs need to exert on the platform in order to generate certain acceleration $\dot{\boldsymbol{W}}$:

$$\boldsymbol{f}_s = \boldsymbol{J}_x^{\,T}(\boldsymbol{G}-\boldsymbol{T})\dot{\boldsymbol{W}} + \boldsymbol{J}_{\boldsymbol{x}}^{\,-\boldsymbol{T}}(\boldsymbol{H}-\boldsymbol{U}). \tag{23}$$

Each $\boldsymbol{f_{si}}$ points in the direction of the corresponding unit vector $\boldsymbol{s_i}$. Neglecting friction on the universal and prismatic joints, Fig. 3 shows that this force $f_{si}\boldsymbol{s}_i$ is formed by three components:

$$f_i \boldsymbol{s}_i = \tau_i \hat{\boldsymbol{d}}_i + \tau_{Ri}\hat{\boldsymbol{k}} + \tau_{Ni}\hat{\boldsymbol{n}}_i \ . \tag{24}$$

where:

τ_i = magnitude of the force exerted by the linear actuator.

τ_{Ri} = magnitude of the vertical reaction force exerted by the base.

τ_{Ni} = magnitude of the vertical reaction force exerted by the actuator' s restriction.

$\hat{\boldsymbol{d}}_{\mathrm{i}}$ = unit vector in the direction of linear actuator i.

$\hat{\boldsymbol{k}}$ = vertical unit vector, pointing upwards.

$\hat{\boldsymbol{n}}_{\mathrm{i}} = \hat{\boldsymbol{k}} \times \hat{\boldsymbol{d}}_{\mathrm{i}}$, vector on the xy plane, perpendicular to $\hat{\boldsymbol{d}}_{\mathrm{i}}$.

Note that $\hat{\boldsymbol{d}}_i, \hat{\boldsymbol{n}}_i, \hat{\boldsymbol{k}}$ are orthogonal and correspond to the *x*, *y*, and *z* directions of the coordinate system placed on *A*. For that reason, the three right-side members of equation (24) correspond to the *x*, *y* and *z* components of force $f_i\boldsymbol{s}_i$ expressed on coordinate system A_i. The rotation matrix ${}^{Ai}\boldsymbol{R}_O$ can be derived from the construction parameters of the robot.

$${}^{A_i}\boldsymbol{f}_{si} = {}^{A_i}\boldsymbol{R}_O \, f_i \boldsymbol{s}_i = \begin{bmatrix} \tau_i \\ \tau_{Ni} \\ \tau_{Ri} \end{bmatrix}. \tag{25}$$

Equations (23) and (25) then complete the inverse dynamics of the robot, with which we can solve the forces needed on the actuators (and the reaction forces of the base) for a desired acceleration trajectory.

5.2 Direct Dynamics

The direct dynamics is more complicated. Even though it is simple to solve (23) and get an expression of the acceleration in terms of the forces exerted by the limbs (equation 26), as seen before such forces are made up of three components, of which only one is known.

$$\dot{\boldsymbol{W}} = (\boldsymbol{G}-\boldsymbol{T})^{-\boldsymbol{1}}\boldsymbol{J}_x^{\,T}\boldsymbol{f}_s - (\boldsymbol{G}-\boldsymbol{T})^{-\boldsymbol{1}}(\boldsymbol{H}-\boldsymbol{U}). \tag{26}$$

Just as the inverse kinematics is relatively simple for parallel robots, but direct kinematics is complex because we do not know the values of the passive joints, we can see that the direct dynamics in this case is made difficult due to the lack of information of the reaction forces. We know, however, that these forces exerted on point D produce no work since there is no displacement in those directions. The Virtual Work Principle tells us that we can write an equation using the differential displacements of each of the

points in the mechanism, due to the forces applied on each point, and the total sum of the work produced by each force in that case is zero. In other words, assuming that the forces on the platform oppose and cancel the forces on the actuators, we then have that for differential (translational and rotational) displacements $\delta \boldsymbol{x} = [\delta \boldsymbol{P} \quad \delta \boldsymbol{\Theta}]^T$, the total work produced is [10, 15]:

$$\boldsymbol{\tau} \cdot \delta \boldsymbol{d} - \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{M} \end{bmatrix} \cdot \begin{bmatrix} \delta \boldsymbol{P} \\ \delta \boldsymbol{\Theta} \end{bmatrix} = 0 \, . \tag{27}$$

We can use Jacobian matrices to relate the differential displacements on the platform to differential displacements on the actuators.

$$\boldsymbol{J}_x \delta \boldsymbol{x} = \boldsymbol{J}_q \delta \boldsymbol{d} \, , \tag{28}$$

$$\delta \boldsymbol{x} = \boldsymbol{J} \delta \boldsymbol{d} \, . \tag{29}$$

Substituting (28) in (26) and after some algebraic development we obtain in (30) the direct dynamics model of the robot

$$\dot{\boldsymbol{W}} = \boldsymbol{G}^{-1} \boldsymbol{J}^{-T} \boldsymbol{\tau} - \boldsymbol{G}^{-1} \boldsymbol{H} \, . \tag{30}$$

6. Kinematic Control

As mentioned in section 2, the inverse kinematics of this 6-PUS manipulator is given by equation (1). So, given a desired position ***p***, the result will be an actuator vector $\boldsymbol{d} = [d_1 \ldots d_6]^{\mathrm{T}}$ which says how long each actuator must expand or contract to achieve the desired position ***d***. It is easy to see that in order to generate a desired trajectory, a set of positions ***p*** is required. As a consequence, a kinematic position control (open loop) can be done by programming equation (1). The corresponding experiments were done and are given in section 8.

7. Dynamic Control

Equation (30) had to be constructed on line during numerical simulations in order to implement a closed loop system with a PD controller (Fig. 4). The block with a little robot image represents such equation (30). As a consequence of the symmetric architecture of the robot, we chose matrices $K_p = k_p \boldsymbol{I}$ and $K_D = k_d \boldsymbol{I}$, $k_p, k_d \in \Re^+$, with ***I*** the identity matrix of order six, see Fig. 3. Section 8 provides results.

Algorithm 3. PD controller.
Step 1 Fix position, velocity and acceleration initial conditions.
Step 2 Solve inverse kinematics with algorithm 1 and obtain vectors ***b*** and ***s***.
Step 3. Fix *t_inicial*, Δt, *t_final*; While $t \leq t\_final$ do step 5 to step 9.
Step 4. Choose the desired ***d*** as ***d_des***, obtain Jacobian matrices and error vectors ***d_err***, $\dot{\boldsymbol{d}}\_\boldsymbol{err}$.
Step 5. Compute matrices ***T, U*** and ***G, H*** solving equations (19) and (22), respectively.
Step 6. Compute $\boldsymbol{\tau} = \boldsymbol{K}_p \boldsymbol{d}_{err} - \boldsymbol{K}_D \dot{\boldsymbol{d}}_{err}$.
Step 7. Determine accelerations in terms of force with equation (30).
Step 8. Update velocities: $\boldsymbol{W}(t_{i+1}) = \boldsymbol{W}(t_i) + \dot{\boldsymbol{W}}(t_i) \Delta t$

Step 9: Update positions: $\boldsymbol{x}(t_{i+1}) = \boldsymbol{x}(t_i) + \boldsymbol{W}(t_{i+1})\Delta t$

Step 10. Solve inverse kinematics (eq.1) and obtain vectors $\boldsymbol{b}$ and $\boldsymbol{s}$ for the new position.

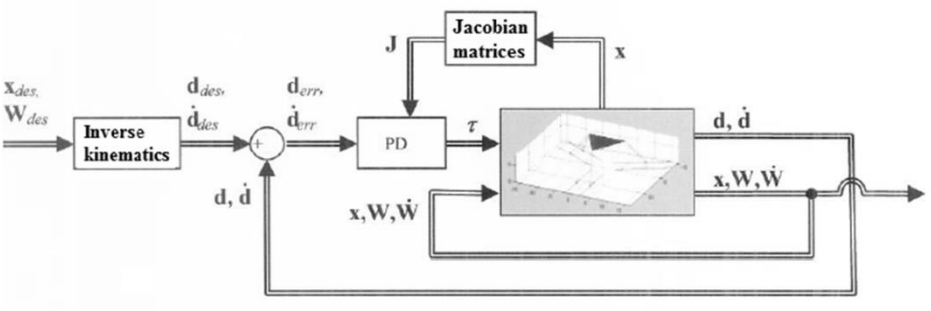

Fig. 4 Complete closed loop control system for the 6-PUS parallel robot.

8. Numerical Simulations

In this section, we provide the simulations corresponding to the inverse and direct kinematics cases, direct and inverse dynamics problems as well as kinematic and dynamic control.

8.1 Inverse Kinematics.

This case is illustrated with two examples. The first one considers a fixed desired position given by $\boldsymbol{p} = [p_x\ p_y\ p_z\ \psi\ \theta\ \phi]^T = [-1\ 3\ 5\ 0.2\ 0.3\ -0.8]^T$. The resulting actuator vector $\boldsymbol{d}$ yielded to be $\boldsymbol{d}$ = [5.49 4.96 1.87 2.15 5.09 5.57]. The second example resulted to be more interesting. A desired trajectory is stored in a long vector $\boldsymbol{p}$ in order to write down a letter "M". See Fig. 5.

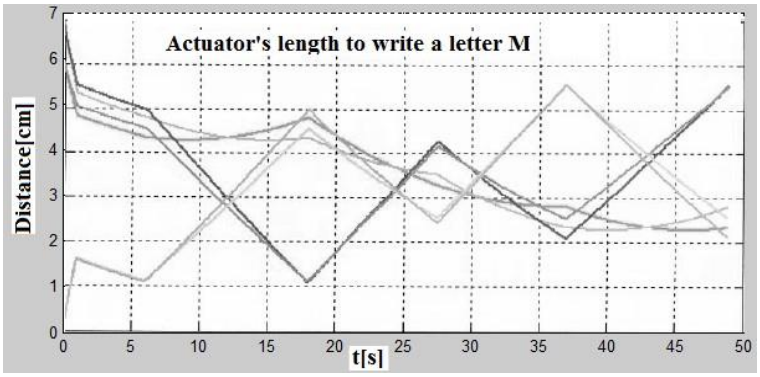

Fig. 5 Writing down letter "M" needs these trajectories of the six actuators.

8.2 Inverse Kinematics Singularities.

Matrix $\boldsymbol{J}_q$ has full rank within a region close to the origin of the workspace. So, for any given point in the workspace, as longer the distance from the origin, as closer $\boldsymbol{J}_q$ to loose rank. The following figure illustrates the values of $|\det(\boldsymbol{J}_q)|$ in the workspace.

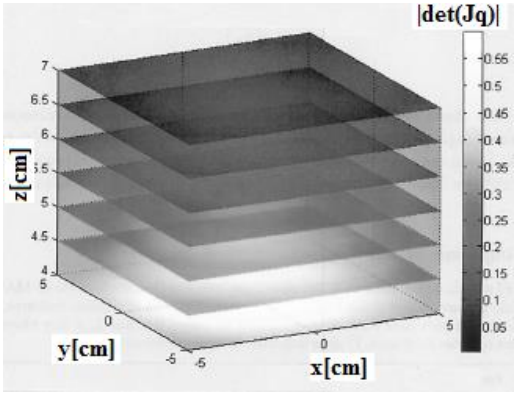

Fig. 6 Values of $|det(\boldsymbol{J}_q)|$.

It is possible to plot a surface associated to the latter figure where $\det(\boldsymbol{J}_q) = 0$. This surface represents the workspace of the manipulator. See Fig. 7.

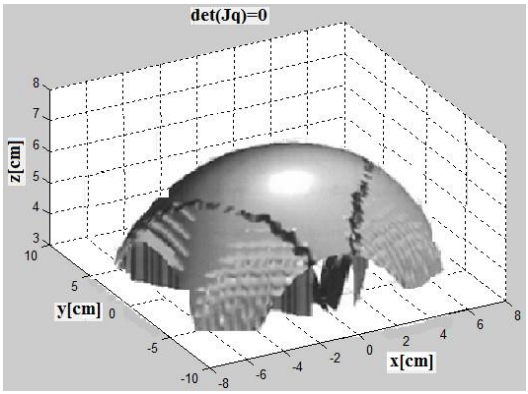

Fig. 7 Surface where $\det(\boldsymbol{J}_q) = 0$.

8.3 Direct Kinematics

The results for the direct kinematics are presented here. Fig. 8a shows the arcs generated by each pair of legs. The distance between each pair of points of arcs 1 and 2 is also shown. The horizontal plane corresponds to the length of the side of the platform and the crosses represent the pair of points whose distance is equal to the same one as the side of the platform. The white dot is the final solution found by algorithm 1 (Section 3). Fig. 8b shows the base of the manipulator along with the rails of the actuators. Here three resulting positions of the platform are shown for a given actuators vector $\boldsymbol{d}$ [13].

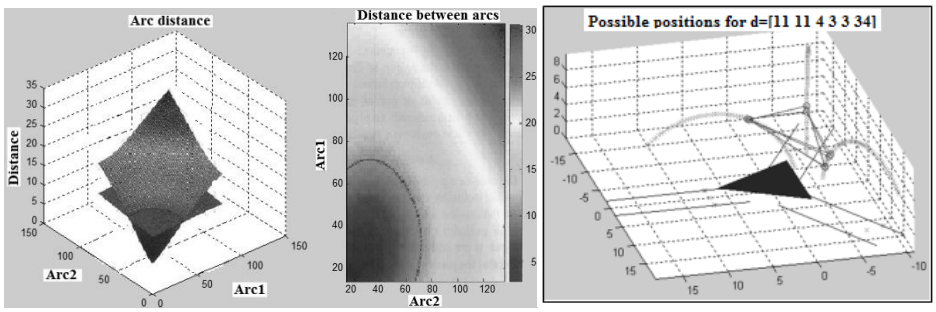

Fig. 8 a) Distance between points belonging to arcs 1 and 2.
b) Resultant positions of the platform.

In [2] there is a very general methodology to obtain the direct kinematics of any parallel robot using an iterative algorithm. That algorithm was implemented for the robot analysed in this paper; however, the geometric method proposed in this paper –inspired by a solution presented by [1] gave better results than those obtained by the iterative method based on the work presented in [2]. It is important to note that the method works because the configuration 6-3 simplifies each pair of legs, as one leg with only one degree of freedom. This is a very useful technique when working with parallel manipulators. We can also see that in this type of manipulators it is very common for the direct kinematics to have many solutions, so in order to plan a trajectory it is very important to know the previous position.

8.4 Direct Kinematics Singularities.

Proceeding in this case, as it was done for the inverse kinematic singularities, a figure is obtained which shows $|\det(\boldsymbol{J}_x)|$.

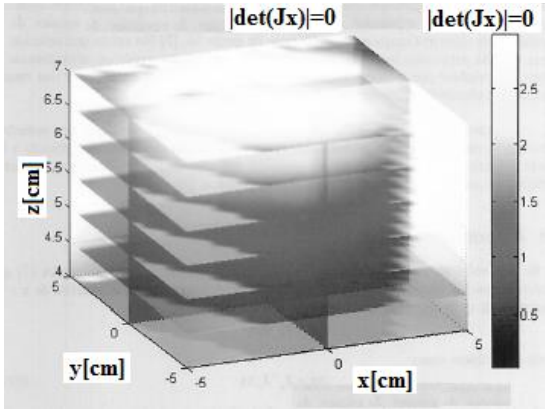

Fig. 9 Numerical values of $|det(\boldsymbol{J}_x)|$ in the workspace

Darker regions appear as closer to the centre of the workspace we are. This means that $\boldsymbol{J}_x$ loses rank when the platform is close to the base. This happens when limbs and motion axes are collinear. Observing Fig. 9, it is deduced that x, y, and z should be limited to $x,y \in [-5, 5]$, $z \in [4, 7]$ cm for all Euler angles equal to zero. However, in order to keep

the platform within some fixed values like those, it is important to determine what position corresponds to the actuators parameters. This issue is solved by the direct kinematics.

8.5 Inverse Dynamics

As it was explained in Section 5.1, the reaction forces exerted by the robot's base and actuator's restrictions were not taken into account in [9]. Hence, when the robot is standing up (completely vertical position), it falls down. To illustrate this fact, the model considered in [9] was implemented in an algorithm which applies a constant input force. Simulations revealed the resultant behaviour just described. Acceleration and position trajectories described that every time the vectors s are in a vertical position, the actuator forces have no effect on the platform, and thus it falls down. As it does, $\boldsymbol{s}$ vectors start losing verticality, since the dot product $(\boldsymbol{s} \cdot \boldsymbol{d})$ rises, and the actuator forces have an effect on the platform and it rises again. In order to improve the model, the reaction forces of the actuator restrictions were taken into account. See plots in [16].

8.6 Direct or Forward Dynamics

Since forward dynamics is needed to solve the control problem (see Dynamic Control Section), the simulations for open loop forward dynamics were omitted but see [16].

8.7 Kinematic Control Simulations and Models proposed

It was explained in Section 6 (Kinematic Control) that an open loop position control can be numerically implemented by programming equation (1), i.e. the inverse kinematics vector equation. With all the background developed so far, it was possible to create several animated 3D designs in Visual Nastran. Two of them are shown in Fig. 10. After refining the whole modelling and control design, a triangle shape was chosen for the base and end effector. See also Section 4.

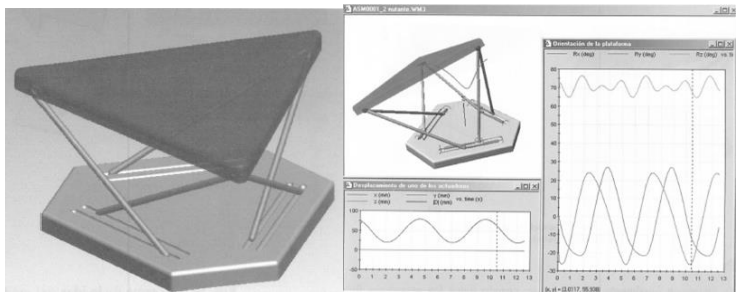

Fig. 10 Early stage 3D render model designs.

8.8 Dynamic Control.

The control loop was designed to have a good regulation performance. Laser cuttings are done via series of step inputs. A simple PD controller was proposed to control the robot's platform in the joint space for each actuator. Since the robot presents a symmetric architecture, all the controllers were chosen identical. The results of the PD controller performance are described next for a step input of amplitude 7cm. The platform position in "z" (platform height) behaved as is indicated in Figs 11a and 11b.

The resulting steady state error in the actuators was 0.5cm, about 7% of the final value. The controller gains were k_p = 300 and k_d = 30.

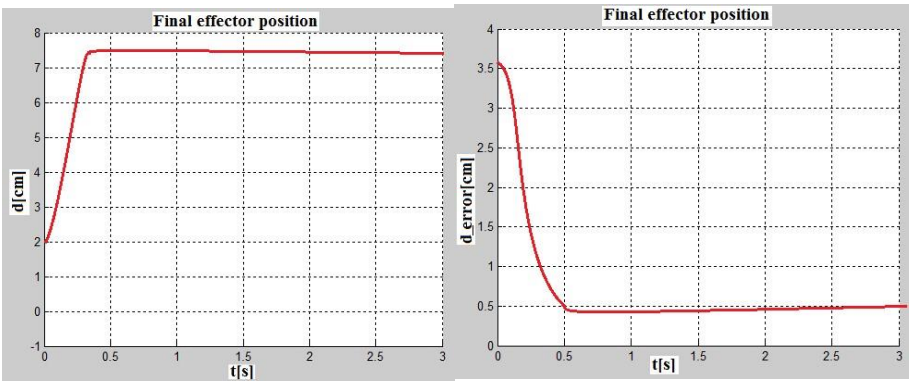

Fig. 11 a) PD controller step response of the platform . b) Steady state error.

In order to improve the latter performance, a PD plus gravity compensation (PD+G) was also implemented. So, step 6 in algorithm 3 was replaced by the following command: Step 6: Compute

$$\boldsymbol{\tau} = \boldsymbol{K}_p \boldsymbol{d}_{err} - \boldsymbol{K}_D \dot{\boldsymbol{d}}_{err} + \boldsymbol{J}^T \begin{bmatrix} 0 & 0 & \text{mg} & 0 & 0 & 0 \end{bmatrix}^T$$

In this case the steady state error was equal to zero after 0.75 seconds of raising time.

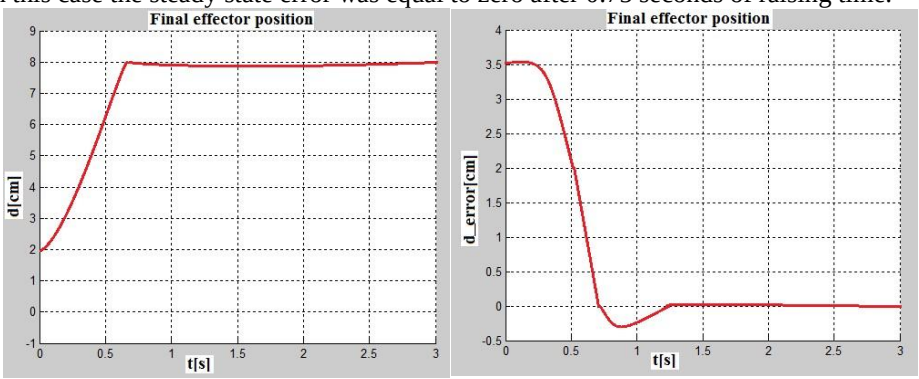

Fig. 12 a)PD+G controller step response b) Steady state response.

8.9 Laser Cutting Machine application design

As it is well known and as it was mentioned in the Abstract and Introduction, parallel robots have a higher payload than their serial counterparts. Mounting a laser cutting device under the robot platform would not modify all the mathematical modelling developed until now. The inertia terms do not change a lot as a result of the inherent light weight of the laser device, as well as the above mentioned payload. For instance, consider to cut a piece following an "M" shape as illustrated in Fig. 13. If such a task is developed with kinematics (see Fig. 5) or by dynamical control (see Fig. 12) the goal will be accomplished. Since the robot architecture is symmetric, the actuators and the PD+G controllers are identical among them. As a consequence, the cutting process will succeed as deduced from the initial design until the closed loop control.

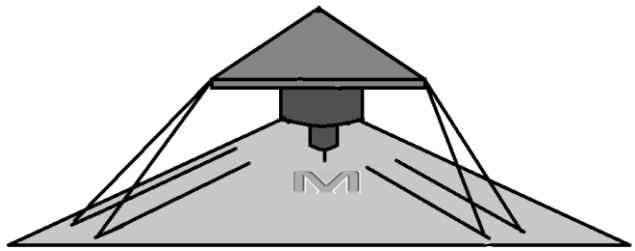

Fig.13 Parallel robot-based laser cutting machine numerical prototype.

9. Conclusions

In this paper, a novel design of a 6-PUS parallel robot was given. Besides, a new method to solve the forward kinematics was presented. In addition, inverse and forward dynamics problems were solved in order to implement a control system for the inverse dynamics model. Singularities were found numerically and a PD and a PD+G controllers were designed for this robot. Both controllers presented a good performance, although PD+G's was definitely better. The controllers' performance was good enough for the present purposes and as a result of their relative simplicity it was not necessary to deal with numerical problems as stiffness, convergence, algebraic loops, etc., as with nonlinear/intelligent controllers. High payload and PDs help to deal with varying inertias on the platform or below it as a laser cutter device.

References

[1] ÁNGELES, J. *Fund. Rob.Mech.Systems.* New York: Springer, USA, 2003, 254 p.

[2] BRUYNINCKX, H. *Parallel Robots. The Robotics WEBook*. [on line], August 2005.<https://static.aminer.org/pdf/PDF/000/351/256/dualities_between_serial_and_parallel_manipulators.pdf>.

[3] CRUZ, P. and FERREIRA R, Kinematic Modeling of Stewart-Gough Platforms. In *ICINCO 2005 Rob. Autom.* Barcelona: Spain, Springer, 2006, p. 93-99.

[4] GAO, X; et al. Generalized Stewart-Gough Platforms and Their Direct Kinematics. *IEEE Trans. on Robotics*, 2005, vol. 21, no. 2, p. 141-151.

[5] HOPKINS, B. and WILLIAMS R. Kinematics, design and control of the 6-PSU platform. *The Industrial Robot*. Bedford, 2002, vol. 29, no. 5, p. 443-445.

[6] JAFARI, F. and MCINROY, J. Orthogonal Gough-Stewart Platforms for Micromanipulation. *IEEE Trans. on Rob. Autom.*, 2003, vol.19, no.4,p.595-603.

[7] KHALIL, W. and GUEGAN, S. Inverse and Direct Dynamic Modeling of Gough-Stewart Robots. *IEEE Trans. on Rob.*, 2004, vol. 20, no. 4, p. 754-762.

[8] LING-FU, K. and SHI-HUI, Z. Research on a Novel Parallel Engraving Machine and its Key Technologies. *Int. J. of A. Rob. Syst*, 2004, vol. 1, no., 4, p. 273–286.

[9] MERLET, J. *Parallel Robots*. Dordrecht: Springer, 2006, 412 p.

[10] TSAI, L. *Robot Analysis*. New York: Wiley, 1999. 520 p.

[11] ARACIL, R. Robots Paralelos. *Rev. Iberoam*., 2006, vol.3, no. 1, p. 16-28.

[12] YIU, Y. K. and H. CHENG. On the Dynamics of Parallel Manipulators. In *Proc. of the 2001 IEEE Int. Conf. Rob. & Aut.* Seoul: Korea, IEEE, 2001, p. 3766-3771.

[13] RUIZ-GARCÍA, J. and ZAVALA-YOÉ, R. and CHAPARRO-ALTAMIRANO, D. and RAMIREZ-MENDOZA, R. Direct Kinematics of a 6-PUS Parallel Robot Using a Numeric-Geometric Method, In *Proc. of the IEEE Int. Conf. Mech., Elec. Auto. Eng.* Morelos: México, IEEE, 2013, p. 46-50.

[14] CHAPARRO, D. and ZAVALA-YOE, R. and RAMIREZ-MENDOZA, R. Kinematic and Workspace Analysis of a Parallel Robot Used in Security Applications, In *Proc. of the IEEE Int. Conf. Mech. Elec. Auto. Eng.*, Morelos: México, IEEE, 2013, p. 3-8.

[15] HIBBELER, R. C. *Eng. Mech.* New Jersey: Prentice Hall, 1998, 315 p.

[16] RUIZ-GARCÍA, J., ZAVALA-YOÉ, R. CHAPARRO-ALTAMIRANO, D., RAMIREZ-MENDOZA, R. Direct and Inverse Dynamics of a 6-PUS Parallel Robot, In *Proc. IEEE Int. Conf .Mech. Elec. Auto. Eng.* Morelos: México, IEEE, 2013, p. 21-26.