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

Kinematic and Dynamical Modelling for Control of a Parallel Robot-based Surveillance/Sentry Device

Ricardo Zavala-Yoé*, Ricardo A. Ramírez-Mendoza and Daniel Chaparro-Altamirano

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

The manuscript was received on 14 October 2014 and was accepted after revision for publication on 26 June 2015.

Abstract:

We contribute with a surveillance and defence system based on a 3SPS-1S parallel manipulator. The central constraining leg of the mechanism increases the stiffness of the system and forces the manipulator to have three pure rotation degrees of freedom. Determining inverse kinematics is trivial but solving forward kinematics is done by a numerical-geometrical approach obtaining a unique solution via artificial neural networks (ANN) and Newton-Raphson's method. An optimized workspace is calculated with a genetic algorithm (GA) and singularities are also computed. Inverse and forward dynamics of the manipulator are also solved for control purposes. Three different designs are presented: one is a classical PID controller and the other two are fuzzy PD controllers. One of them works in sliding mode.

Keywords:

Artificial intelligence, dynamics, fuzzy sliding mode controller, kinematics, parallel robots, surveillance systems.

1. Introduction

Parallel manipulators have received a lot of attention from researchers over the past couple of decades, due to the advantages they present over their serial counterparts, such as more accuracy, higher load capacity/robot mass ratio and more rigidity. Researchers have taken an interest in parallel robots with less than six degrees of freedom (DOF), because in some applications there is no need to be able to move and rotate the end effector in every direction, and using less than six DOF manipulators decreases the costs. Three DOF spherical manipulators, also known as parallel wrists, can be used as an

* *Corresponding author: Tecnológico de Monterrey, Calle del Puente 222, Ejidos de Huipulco, 14380 Mexico City, Mexico. Phone: +52 555 54832020, rzavalay@itesm.mx*

alternative to the wrists with three revolute joints for applications where there is need to orient something. Recall of that parallel robot's nomenclature is based on the types of joints which constitute the mechanism. Thus, 3SPS-1S means that our manipulator has three limbs, each of them consisting of a spherical (S) plus a prismatic (P) plus a spherical (S) joint. In addition, a central limb which moves by means of another spherical (S) joint is part of the structure (see Fig. 1). Over- constrained and not-over constrained parallel wrists have been studied. Over- constrained parallel manipulators in rotational (R) joints-based robots, such as Gosselin's 3-RRR manipulator [1], have the advantage of always performing spherical movements; however, when geometric errors occur, they undergo high internal loads and can sometimes jam [2]. Not-over constrained parallel wrists can be divided in two groups. The first group consists of mechanisms that only have spherical movements if some geometric conditions are met [3-5]. The second group has the base and the platform linked directly by a spherical joint [7] that forces the manipulator to rotate around it. The manipulators from the first group have singular configurations that must be avoided during the movement of the mechanism, otherwise they can lose their pure spherical motion and obtain some translational movements [2, 6]; furthermore, the spherical motion strongly depends on the correct manufacturing and mounting of each part; errors in these processes can lead to non-spherical movements of the platform. The spherical joint between the base and the platform of the manipulators from the second group does not allow the platform to translate; therefore they have bigger manufacturing error tolerances than the parallel wrists from the first group, which leads to cheaper manipulators.

The drawback of these manipulators is that they generally have smaller workspaces than those of the robots of the first group. Since the workspace is smaller, it is very important to design the manipulator so that it has a workspace big enough to suit the needs of the application at hand. Automated orientation mechanisms have been used in various security and defence applications, such as Samsung Techwin's SGR series surveillance robots or the SGR-A1 sentry gun [8]. These machines can be used as surveillance systems or as nonlethal defence mechanisms to protect houses, industries, border crossings, hospitals, etc. Surveillance systems using cameras with high optical zoom (> 30x) need very precise movements to keep the track of a target. Defence systems (sentry guns) must have very good accuracy in order to correctly hit their targets. The central leg of the 3SPS-1S parallel manipulator proposed by Cui and Zhang [7] eliminates unnecessary movements, forces the manipulator to have only rotational motion and improves the stiffness of the system. These characteristics, along with the fact that in general parallel manipulators have more accuracy and are able to carry more load than their serial counterparts, make the 3SPS-1S a very good candidate to be used in security and defence applications.

The paper is arranged like this: section 2 presents a brief description of the 3SPS-1S manipulator; in section 3 the inverse kinematics problem is solved; section 4 presents a geometric method to solve the direct kinematics of the manipulator and a ANN–based method to solve the forward kinematics locally; in section 5 the workspace of the manipulator is analysed and calculated, in addition a GA–based method to maximize the workspace maintaining the size of the robot as close as possible to a defined set of parameters is presented; in section 6 the Jacobian matrix is computed to determine singularities. Sections 7 and 8 solve inverse and direct dynamics, respectively, which are useful for control purposes (part 9). Finally, conclusions are formulated in section 10.

2. Manipulator Description

A particular case of the 3SPS-1S manipulator proposed by Cui and Zhang is shown in Fig. 1. It has three identical legs, made of two bodies, linked by an actuated prismatic joint. The legs are attached to the platform and the base by spherical joints. It is assumed that the platform and the base are circular, with radii r_p and r_b respectively, and that the spherical joints of the legs are located along these circumferences. There is also a central passive leg that connects the centre of the base to the centre of the platform using a spherical joint.

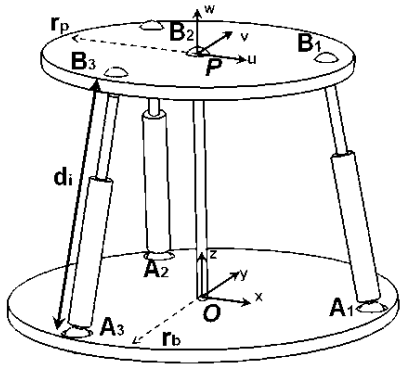

Fig. 1 Model of the 3SPS-1S robot

The general coordinate system *xyz* is located at the centre of the base (*O*), and the coordinate system of the platform *uvw* with origin *P* is located at the centre of the spherical joint of the central leg. Euler angles, roll, pitch and yaw, are denoted by ω, ϕ, and ψ, respectively. As shown in Fig. 1, the spherical joints are located on the base at points A_i, [7]. [2].

3. Inverse Kinematics

In order to move the platform to a desired position given by the angles ω, ϕ, and ψ, the length of the limbs d_i must be calculated. This problem is known as the inverse kinematics and it is very easy to solve for parallel robots. Cui and Zhang presented a solution to the general 3SPS-1S inverse kinematics problem [7], and Gan, Seneviratne and Dias presented a solution to a particular case similar to the manipulator shown in Fig. 1 [9]. Let $\boldsymbol{a}_i = [a_{xi} \;\; a_{yi} \;\; 0]^T$ be the vector from origin *O* to point A_i in the *xyz* system, ${}^B\boldsymbol{b}_i$ the vector from origin *P* to point B_i in the rotating system *uvw*, $\boldsymbol{p} = [0 \;\; 0 \;\; h]^T$ the vector between points *O* and *P*, and ${}^A\boldsymbol{R}_B$ an order 3 matrix that represents the rotation of system *uvw* with respect to the coordinate system *xyz*. The vector between points A_i and B_i can be written as

$$\overline{A_iB_i} = \boldsymbol{p} + {}^A\boldsymbol{R}_B \, {}^B\boldsymbol{b}_i - \boldsymbol{a}i \tag{1}$$

Since $d_i = |\overline{A_iB_i}|$, we can write the solution of the inverse kinematic problem as:

$$\begin{aligned} d_i &= \sqrt{(b_{xi} - a_{xi})^2 + (b_{yi} - a_{yi})^2 + (b_{zi} + h)^2} \\ {}^A\mathbf{R}_B \, {}^B\mathbf{b}_i &= \mathbf{b}_i = \begin{bmatrix} b_{xi} & b_{yi} & b_{zi} \end{bmatrix}^T, \; i = 1, 2, 3. \end{aligned} \tag{2}$$

4. Direct Kinematics

The direct kinematics problem is to obtain the orientation of the platform given the limbs length d_1, d_2, d_3. As opposed to the inverse kinematics, obtaining the forward kinematics of a parallel robot is a complex task, however, it is important because usually sensors are used to measure the position of the actuators, but not that of the platform; therefore, in order to know the current orientation of the platform given the information provided by the sensors, the forward kinematics must be calculated.

4.1. Geometric Method

Gan, Seneviratne and Dias have found an analytical way to solve this problem [9], where they get 8 solutions; however, it is also possible to solve the forward kinematics problem using a geometric method. Let S_p be a sphere with radius r_p and centre on point P, and S_i a sphere with radius d_i and centre on point A_i. The intersection of spheres S_p and S_i is circle C_i that represents all possible locations of point B_i given the limbs length. Let d_{B12} be the distance between points B_1 and B_2, d_{B13} the distance between points B_1 and B_3, d_{B23} the distance between points B_2 and B_3, c_{1i} a point in circle C_1, c_{2i} a point in circle C_2, and c_{3i} a point in circle C_3. If the distance between c_{1i} and c_{2i} is d_{B12}, the distance between c_{1i} and c_{3i} is d_{B13}, and the distance between c_{2i} and c_{3i} is d_{B23}, then those points are a solution to the forward kinematics. Doing this for every point in each circle (given a step size) it is possible to find all solutions to the forward kinematics problem. Fig. 2 shows two solutions of the forward kinematics problem obtained with the geometric method. The parameters used (in cm) were: $h = 25$, $r_b = 20$, $r_p = 15$, $d_1 = 29$, $d_2 = 26$, $d_3 = 22$.

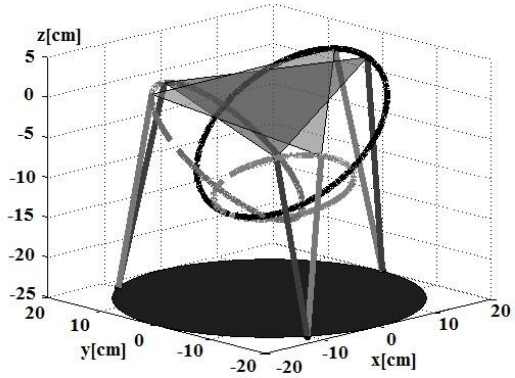

Fig. 2 Geometric method determined this forward kinematics

4.2. Local Forward Kinematics

The method presented by Gan, Seneviratne and Dias, and the geometric method usually find more than one solution to the forward kinematics. The problem is that given an initial orientation of the platform, it is not possible to know which of these solutions is the one that the mechanism will move to; furthermore, if we want to choose only one solution, it must be done manually. If we want to track the position of the platform at every moment, a method to quickly calculate the forward kinematics based on the current position of the platform is needed. To solve this problem, an artificial neural network (ANN) was proposed. The ANN consists of six inputs, three outputs and two hidden layers, one with four neurons and the other with eight neurons (see Fig. 3). The

inputs are the length of the three limbs and the octant of point B_i. The outputs are the three angles used to form the rotation matrix ${}^{A}\boldsymbol{R}_{B}$. The ANN is trained with a back propagation algorithm. A training data set of 1000 samples was obtained by varying the angles (one at a time) with a small resolution (1°), and obtaining d_1; d_2 and d_3 using inverse kinematics. For any given length of the limbs and initial position of the platform, it is possible to find an approximate solution to the direct kinematics using the trained ANN. Once the approximate solution is found, Eq. 2 can be written three times (one for each limb) and the system of three nonlinear equations can be solved using the Newton-Raphson's method. A validation set consisting of 100 positions was used to check the method. The resulting error between the two trajectories was negligible.

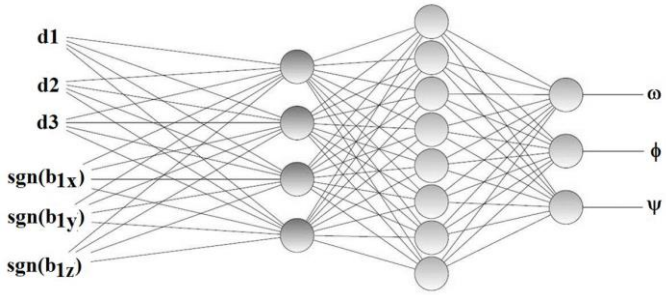

Fig. 3 ANN used to determine forward kinematics

5. Workspace Analysis

The inverse kinematics of the manipulator lets us know the length that the actuators must have in order for the platform to reach a certain orientation; nevertheless, it is important to know if the manipulator can reach that orientation before trying to move it to that position and that is why it is essential to analyse the workspace of the manipulator. There are many ways to represent the workspace of orientation parallel robots, such as for instance the maximal workspace [10]. Recall our Euler angles representing the orientation of the platform, and A_1 the point of interest. The maximal workspace of the 3SPS-1S robot can be defined as all the locations of point A_i that may be reached with at least one combination of the Euler angles. In order for a point to be part of the workspace, three mechanical conditions must be met: max-min limb length, maximum joint angles, and collision between limbs. Prismatic actuators can contract to a minimum length, and expand to a maximum length. The min-max limb length condition states that the lengths d_1, d_2 and d_3 must be greater than or equal to the minimum length of the actuators, and less than or equal to the maximum length of the actuators. Let α_{ij} be the angle between limb i and plane j, being plane 1 and 2 the base and the platform of the robot respectively, and β_j the maximum allowed angle between any limb and plane j. The maximum joint angles condition states that α_{ij} must be less than or equal to β_j. Let $\boldsymbol{n}_j$ be the vector normal to the plane j. The angle α_{ij} can be calculated as follows: $\alpha_{ij} = 90° - \arcsin\left(\left|\boldsymbol{n}_j \cdot \overline{A_1B_i}\right| / |\,\boldsymbol{n}_j\,|\,|\,\overline{A_1B_i}\,|\right)$. If the spherical joints have a full range of motion, then $\beta_j = 90°$, but if the joints have a reduced range of motion, then β_j is less than 90°. The third condition states that if two limbs collide (including the central mast), the point does not belong to the workspace. Tsai and Lin have proposed a method to find

collisions between the legs of a Stewart-Gough platform [11] that can also be used with the 3SPS-1S robot. A first solution computed a non-optimal workspace of the 3SPS-1S parallel robot using the parameters h = 20 cm, r_b = 15 cm, r_p = 15 cm and $\beta_j = 90°$ [19].

5.1. Workspace Optimization

As mentioned earlier in this paper, the biggest problem of parallel manipulators like the 3SPS-1S parallel wrist is the lack of a big workspace; therefore, optimizing the parameters of the robot in order to maximize the workspace is very important. Nevertheless it is also important for the manipulator to have its parameters (r_b, r_p and h) as close as possible to a set of desired parameters, mainly because many times there are size limitations in the location where the manipulator is to be placed. Different methods to optimize the parameters of parallel robots have been proposed [12-16]. Some try to maximize the workspace, others to improve the dynamic behaviour or dexterity of the robot and others to reduce the number of singularities. The following method tries to maximize the workspace and at the same time, keep the robot parameters as close as possible to a set of desired parameters. A genetic algorithm was used to optimize the parameters of the robot:

Genetic algorithm.
Step 1. Fix size of population, n. n = 3 in this case; fix r_b, r_p, h.
Step 2. Create an initial population p of size n.
Step 3. While finishing condition is not satisfied do:
Step 4. Compute a fitness function for each chromosome.
Step 5. Choose chromosome parents by some selection criterion.
Step 6. Create a new generation by recombination to the parents.
Step 7. Apply mutation to the new generation.
Step 8. Replace partially or totally the current population with the members of the new population.
Step 9. End While.

Let $\Delta\omega$, $\Delta\phi$, and $\Delta\psi$ be the range of the pitch, yaw and roll angles respectively, and h_d, r_b and r_p the desired parameters. The fitness function can be written as

$$fitness = a\Delta\phi + b\Delta\psi + c\Delta\omega - d|h - h_d| - e|r_b - r_{bd}| - f|r_p - r_{pd}| \quad (3)$$

where a, b, c, d, e and f are weights given to each parameter. The value of these weights depends on the importance of each parameter for a specific application. Using the desired parameters h_d = 20 cm, r_{bd} = 15 cm, r_{pd} = 15 cm; weights a = 2, b = 2, c = 0.05, d = 10, e = 5, f = 5, the optimized workspace was calculated. The resulting parameters were h = 25.17 cm, r_b = 10.98 cm, r_p = 13.54 cm. The optimized workspace (with $\beta_j = 90°$) can be observed in Fig. 4. Comparing it with the original workspace [19], it is clear that the optimized workspace is bigger than the original. Using the parameters obtained with this optimization method, a sentry device prototype was designed in Fig. 5.

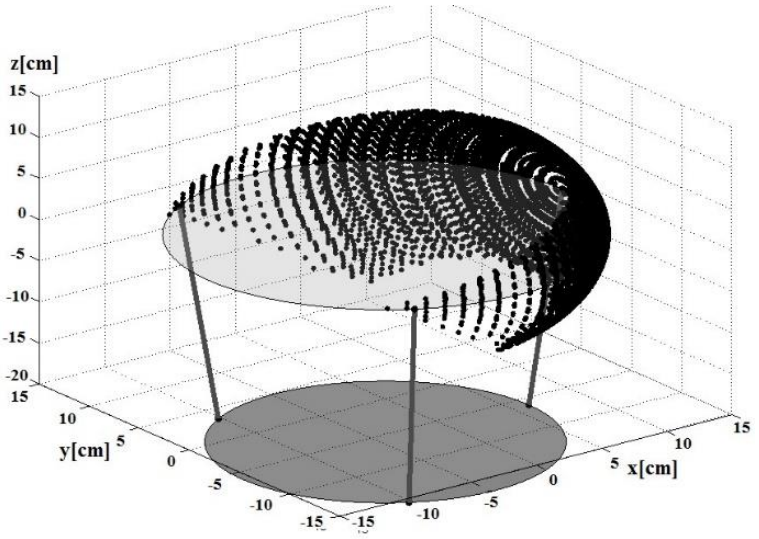

Fig. 4 Optimized Workspace for dynamic control and surveillance device design

Fig. 5 Parallel robot as a surveillance system

6. Jacobian Analysis

While the 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. Defining $d_i = \overline{A_i B_i}$ in Eq. 1 and deriving such Eq. 1 with respect to time we get:

$$d_i \boldsymbol{\omega}_i \times \boldsymbol{s}_i + \dot{d}_i \boldsymbol{s}_i = \boldsymbol{\omega}_p \times \boldsymbol{b}_i \tag{4}$$

where $\boldsymbol{b}_i$ represents the vector between the origin O and point B_i, $\boldsymbol{s}_i$ the unit vector pointing from Ai to Bi, ω_p the angular velocity of the platform, and d_i the angular velocity of the ith leg. Dot multiplying both sides of Eq. 4 by $\boldsymbol{s}_i$ gives:

$$\dot{d}_i = \left(\boldsymbol{b}_i \times \boldsymbol{s}_i\right) \omega_p \tag{5}$$

The matrix multiplying the actuators velocity vector ω_p in Eq. 5 is the Jacobian matrix of the 3SPS-1S parallel robot. Whenever the determinant of the Jacobian matrix becomes zero, it means the robot presents a singularity at that point. By evaluating the determinant of the Jacobian matrix for each point belonging to the workspace, it is

possible to obtain all the singularities presented by the 3SPS-1S robot. The genetic algorithm computed that the set of singular configurations is given by the straight line $\omega = \psi = 0°$, $\phi \neq 0°$ in the (ω,ψ,ϕ) frame [18, 19].

7. Inverse Dynamics

Recall that the inverse dynamics problem can be posed as: Given the trajectory vector of the platform (end effector), determine the actuated joint forces necessaries to get such trajectory vector in the platform. It is possible to orient the manipulator using the kinematics and controlling the length of the actuators, nonetheless, sometimes the size and weight of the bodies may be big enough so that the inertia forces have a considerable impact on the movement of the manipulator. When this happens it is better to control the robot using the inverse dynamics instead of the inverse kinematics. Let $\boldsymbol{x}_p$ be a six dimensional vector describing the position and orientation of the platform, $\boldsymbol{q}$ and $\boldsymbol{\tau}$ the vectors representing the position and the force (or torque) of the actuators respectively, $\hat{\boldsymbol{F}}_p = \begin{bmatrix} \hat{f}_{px} & \hat{f}_{py} & \hat{f}_{pz} & \hat{n}_{px} & \hat{n}_{py} & \hat{n}_{pz} \end{bmatrix}^T$ the six dimensional wrench applied to the platform, and $\hat{\boldsymbol{F}}_i = \begin{bmatrix} \hat{f}_{ix} & \hat{f}_{ipy} & \hat{f}_{iz} & \hat{n}_{ix} & \hat{n}_{iy} & \hat{n}_{iz} \end{bmatrix}^T$ the six dimensional wrench of the *i*th limb. By the principle of virtual work, the following equation can be derived for a general parallel manipulator [10].

$$\delta \boldsymbol{q}^T \tau + \delta \boldsymbol{x}_p \hat{\boldsymbol{F}}_p + \sum \delta \boldsymbol{x}_i^T \hat{\boldsymbol{F}}_i = 0 \tag{6}$$

Since the 3SPS-1S parallel mechanism has only three DOF, $\boldsymbol{x}_p$ becomes a three dimensional vector describing the orientation of the platform. Also, the moving platform can only bear the forces $\boldsymbol{F}_p = \begin{bmatrix} \hat{n}_{px} & \hat{n}_{py} & \hat{n}_{pz} \end{bmatrix}^T$ [10], while the other components are supported by the passive joints. The virtual displacement of the platform and of the *i*th limb is related to that of the platform by the following equations:

$$\delta \boldsymbol{q} = \boldsymbol{J}_p \delta \boldsymbol{x}_p \tag{7}$$

$$\delta \boldsymbol{x}_i = \delta \boldsymbol{x}_p \tag{8}$$

Since Eq. 6 holds for any virtual displacement, using Eq. 6, 7 and 8 we get

$$\boldsymbol{J}_p^T \boldsymbol{\tau} + \boldsymbol{F}_p + \sum_i \boldsymbol{J}_i^T \hat{\boldsymbol{F}}_i = 0 \tag{9}$$

Isolating the input vector term in Eq. 9 the expression for the inverse dynamics is obtained:

$$\boldsymbol{\tau} = -\boldsymbol{J}_p^{-T} \left(\boldsymbol{F}_p + \sum_i \boldsymbol{J}_i^T \hat{\boldsymbol{F}}_i \right). \tag{10}$$

Since each limb is composed by two bodies, we can divide each limb action in two separate wrenches, one for the cylinder and one for the piston, so Eq. 10 can be written as

$$\boldsymbol{\tau} = -\boldsymbol{J}_p^{-T} \left(\boldsymbol{F}_p + \sum_{i=1}^{3} {}^i\boldsymbol{J}_{1i}^{T\,i} \hat{\boldsymbol{F}}_{1i} + {}^i\boldsymbol{J}_{2i}^{T\,i} \hat{\boldsymbol{F}}_{2i} \right) \tag{11}$$

where ${}^{i}\boldsymbol{J}_{1i}$ and ${}^{i}\hat{\boldsymbol{F}}_{1i}$ are the Jacobian matrix and wrench of the *i*th cylinder, respectively; and ${}^{i}\boldsymbol{J}_{2i}$ and ${}^{i}\hat{\boldsymbol{F}}_{2i}$ the Jacobian matrix and wrench of the *i*th piston, respectively. Let *g* be the gravity, ${}^{A}\boldsymbol{I}_{p} = {}^{A}\boldsymbol{R}_{B}\ {}^{B}\boldsymbol{I}_{p}\ {}^{B}\boldsymbol{R}_{A}$, with ${}^{A}\boldsymbol{R}_{B} = {}^{B}\boldsymbol{R}_{A}{}^{T}$, the inertia matrix of the platform expressed in terms of the fixed coordinate system xyz, m_{1i} the mass of the *i*th cylinder, m_{2i} the mass of the *i*th piston. The wrenches can be written as:

$$\mathbf{F}_{p} = \begin{bmatrix} -{}^{A}\mathbf{I}_{p}\dot{\boldsymbol{\omega}}_{p} - \boldsymbol{\omega}_{p} \times \left({}^{A}\mathbf{I}_{p}\boldsymbol{\omega}_{p} \right) \end{bmatrix} \tag{12}$$

$${}^{i}\hat{\boldsymbol{F}}_{1i} = \begin{bmatrix} m_{1i}\,{}^{i}\boldsymbol{R}_{A}\,g - m_{1i}\,{}^{i}\dot{\boldsymbol{v}}_{1i} \\ -{}^{i}\boldsymbol{I}_{1i}\,{}^{i}\dot{\boldsymbol{\omega}}_{i} - {}^{i}\boldsymbol{\omega}_{i} \times \left({}^{i}\boldsymbol{I}_{1i}\,{}^{i}\boldsymbol{\omega}_{i} \right) \end{bmatrix} \tag{13}$$

$${}^{i}\hat{\boldsymbol{F}}_{2i} = \begin{bmatrix} m_{2i}\,{}^{i}\boldsymbol{R}_{A}\,g - m_{2i}\,{}^{i}\dot{\boldsymbol{v}}_{2i} \\ -{}^{i}\boldsymbol{I}_{2i}\,{}^{i}\dot{\boldsymbol{\omega}}_{i} - {}^{i}\boldsymbol{\omega}_{i} \times \left({}^{i}\boldsymbol{I}_{2i}\,{}^{i}\boldsymbol{\omega}_{i} \right) \end{bmatrix} \tag{14}$$

The Jacobian matrices $\boldsymbol{J}_{\mathrm{p}}$, ${}^{\mathrm{i}}\boldsymbol{J}_{1\mathrm{i}}$ and ${}^{\mathrm{i}}\boldsymbol{J}_{2\mathrm{i}}$ as well as all variables from Eq. 12, 13 and 14 can be obtained using the same procedure used by [20] in his 6-UPS dynamics example, taking into consideration that in this case, there are only three limbs and that the platform has only rotational movements. In order to calculate the inertia moments, it is assumed that the cylinder is a hollow cylinder with an internal radius of 1.3 cm, an external radius of 1.5 cm and a length of 16.866 cm. The piston is also assumed to be a hollow cylinder, with an internal radius of 1 cm, an external radius of 1.25 cm and a length of 16.866 cm. The masses were calculated using the supposition that both the piston and the cylinder are made of aluminium [19].

8. Forward Dynamics

Given the actuated joint forced/torques, determine the trajectory, velocities and accelerations of the platform (end effector). It is possible to control the manipulator using inverse dynamics, however, not always a physical model of the manipulator is available, so in order to simulate the behaviour of the system, the forward dynamics of the manipulator is needed [18, 23]. Getting the forward dynamics from Eq. 11 is very complicated; therefore, another method will be used. In order to simplify the model, it is considered that each leg is just one body that changes its length. Let $\boldsymbol{J}_{i}$ be the inertia moment of the leg around its *x* and *y* axis, $\boldsymbol{\tau}_{i}$ be the force produced by the actuator on point B_i along the unit vector $\boldsymbol{s}$i, and $\boldsymbol{f}_{\mathrm{N}}$ a force perpendicular to $\boldsymbol{s}_{i}$, where $\boldsymbol{s}_{i}$ is the unit vector going from point A_i to point B_i, due to the inertia. We have:

$$\boldsymbol{f}_{i} = \tau_{i}\boldsymbol{s}_{i} + \boldsymbol{f}_{N_i} \tag{15}$$

Let $\boldsymbol{M}_{\mathrm{N}}$ be the resultant torque of the forces $\boldsymbol{f}_{\mathrm{Ni}}$ around the centre of the platform. If $\boldsymbol{M}$ is the torque on the end effector, the moment equilibrium equation may be written as:

$$\boldsymbol{M} = \sum_{i=1}^{3} \boldsymbol{\tau}_{i} \left(\overline{\boldsymbol{PB}}_{i} \times \boldsymbol{s}_{i} \right) + \boldsymbol{M}_{N} \tag{16}$$

Following the method used by [10] using Eq. 16 and 17, we can obtain the forward dynamics of the manipulator:

$$\ddot{\boldsymbol{x}}_{p} = \left(\boldsymbol{T}_{1} - \boldsymbol{V}_{1} \right)^{-1} \left(\boldsymbol{J}^{T}\boldsymbol{\tau} - \boldsymbol{T}_{2} + \boldsymbol{V}_{2} \right) \tag{17}$$

where:

$$\boldsymbol{e}\times\boldsymbol{a}=\boldsymbol{e}^{*}\boldsymbol{a},\quad \boldsymbol{e}^{*}=\begin{bmatrix}0 & -z & y\\ z & 0 & -x\\ -y & x & 0\end{bmatrix},\text{for any } \boldsymbol{e}=\begin{bmatrix}x & y & z\end{bmatrix}^{T},\boldsymbol{a}^{T}=\begin{bmatrix}a_x & a_y & a_z\end{bmatrix}^{T}$$

where:

$$\boldsymbol{T}_1=\boldsymbol{I}_p$$

$$\boldsymbol{T}_2=\boldsymbol{\omega}_p\times\left(\boldsymbol{I}_p\boldsymbol{\omega}_p\right)$$

$$\boldsymbol{U}_{1i}=-\boldsymbol{b}_i^{*}$$

$$\boldsymbol{U}_{2i}=\boldsymbol{\omega}_p\times\left(\boldsymbol{\omega}_p\times\boldsymbol{b}_i\right)$$

$$\boldsymbol{V}_1=\sum_{i=1}^{3}\frac{\boldsymbol{J}_i}{d_i^{\,2}}\boldsymbol{b}_i^{*}\boldsymbol{s}_i^{*2}\boldsymbol{U}_{1i}$$

$$\boldsymbol{V}_2=\sum_{i=1}^{3}\frac{\boldsymbol{J}_i}{d_i^{\,2}}\boldsymbol{b}_i^{*}\boldsymbol{s}_i^{*2}\boldsymbol{U}_{2i}$$

And $\boldsymbol{I}_p$ is the inertia matrix of the platform, $\boldsymbol{\omega}_p$ the angular velocity of the platform, $\boldsymbol{b}$i the vector going from point P to point B_i, and d_i the length of the ith leg [23].

9. Dynamic Control

While it is possible to use kinematics as a model to control the orientation of the platform by controlling each leg separately, sometimes this can lead to a non-desired position. Instead if the dynamics of the of the platform's error position is used to control the system, the platform will always reach the desired orientation. In order to use this kind of control, it is necessary to express the dynamics of the manipulator (Eq. 17) in the Euler- Lagrange family of systems form as in Eq. 18 [20, 21].

This implies that either we model a given parallel robot with Euler – Lagrange equations from the beginning or we start applying the D'Lambert representation in order to try to deduce from them an equivalent set of Euler – Lagrange equations. Moreover, even if our robot is suitable to be represented by the Euler – Lagrange equations, most likely we will have to apply their so called first type, which involves Lagrange multipliers. With some exceptions, the resulting model might be quite complicated and impossible to be implemented in any scientific software of simulation as MATLAB/SIMULINK, MAPLE, etc. [10]. This happens as consequence of the constraints of the closed loop kinematic chains of a parallel manipulator. Just to take a look to the mathematical structures described, we illustrate the Euler – Lagrange family of systems which is given by the set of equations shown below [19-21]:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}+\boldsymbol{K}(\boldsymbol{q})=\boldsymbol{\tau} \tag{18}$$

In the latter expression, $\boldsymbol{q}$ is the generalized coordinated vector, $\boldsymbol{M}$ is the inertia matrix function, $\boldsymbol{C}$ is the Coriolis matrix function, $\boldsymbol{K}$ is the gravity terms vector function and $\boldsymbol{\tau}$ is the input vector (forces, torques) [22]. Although not shown, in our case, it was possible to find an equivalent representation from D'Alambert equations to the Euler – Lagrange representation.

9.1 PID Controller

The PID controller is defined by $\boldsymbol{\tau} = \boldsymbol{K}_p\tilde{\boldsymbol{q}} + \boldsymbol{K}_D\dot{\tilde{\boldsymbol{q}}} + \boldsymbol{K}_I\int\tilde{\boldsymbol{q}}dt$ where the error signal $\tilde{\boldsymbol{q}} = \boldsymbol{q}_d - \boldsymbol{q}$, $\boldsymbol{q}_d$ is the desired position and $\boldsymbol{q}$ is the actual platform's position. $\boldsymbol{K}_p$, $\boldsymbol{K}_D$, $\boldsymbol{K}_I$ are symmetric positive definite matrices. The closed loop equation for our robot, represented now by Eq. 18, will result by substituting the PID controller equation in the plant Eq. 18. In addition, global stability of this controlled system can be assured by Lyapunov's theory [22]. Tuning of this simple PID was done by trial and error, tuning first $\boldsymbol{K}_p$ and later $\boldsymbol{K}_I$ and $\boldsymbol{K}_D$. The auto-tuning SIMULINK function could not deal with the robot model nonlinearities.

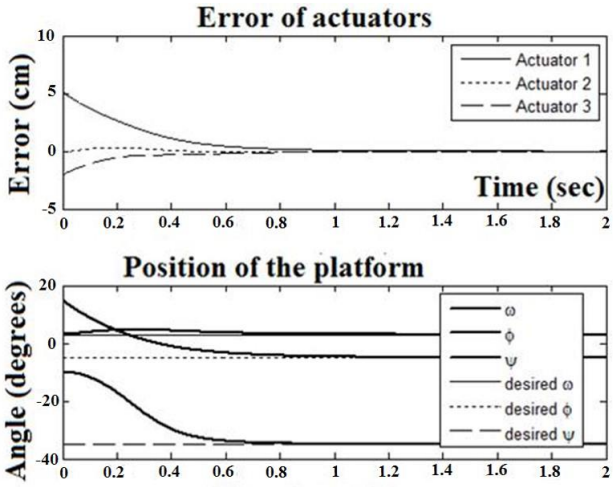

Fig. 6 Resultant position error of the platform with a PID controller

Note in Fig. 6 how the PID achieves a good regulation (tracking a constant trajectory) in the three Euler angles. When not too much precision is required, it is possible to control the manipulator by just using kinematics (i.e. surveillance application), but when such precision has to be high, a dynamical control is advised, as we proceed next (sentry device application).

9.2 PD Fuzzy Controller

The sentry device design is shown in Fig. 7. In order to control it, it will be taken into account that fuzzy controllers have good performance in general [17]. In order to test this, a PD fuzzy controller was designed with a set of plant parameters h = 25.17 cm; r_b = 10.98 cm and r_p =13.54 cm. The PD fuzzy controller is defined by the following set of rules (Tab. 1) where NB = Negative Big, N = Negative, Z = Zero, P = Positive and PB = Positive Big [22], [17]. Tuning was done assuming first a P fuzzy controller (i.e., symmetric table below) and later adjusting the fuzzy values of the D part according to the plant performance (see details in [17]).

Fig. 8 shows a test scenario, in which the trajectory of an intruder is represented by a white line. The twelve points shown in Tab. 2 are the equivalent to the white circles shown in Fig. 8, and represent the X and Y coordinates of the intruder with respect to the manipulator at different times. Furthermore, the last two columns of Tab. 2 represent the

pitch and yaw that the manipulator must have in order to aim at the target, considering that the height of the manipulator is 3 meters above ground, and it aims to an intruder's point, located at 1.5 meters from the ground. It is important to note that the barrel of the device is supposed to be exactly over the *x* axis of the platform; therefore, for any value of the roll, the robot will hit the target. For this reason the roll can be set to any value. However, in order to avoid passing through a singular configurations, it is better to choose a value different than zero; in this case the value chosen was 0.01 radians [22].

Tab. 1 Rules for the fuzzy controller

de/de'	NB	N	Z	P	PB
NB	NB	NB	NB	N	Z
NB	NB	NB	N	Z	P
Z	NB	N	Z	P	PB
P	N	Z	P	PB	PB
PB	Z	P	PB	PB	PB

Fig. 7 Parallel robot used as sentry device

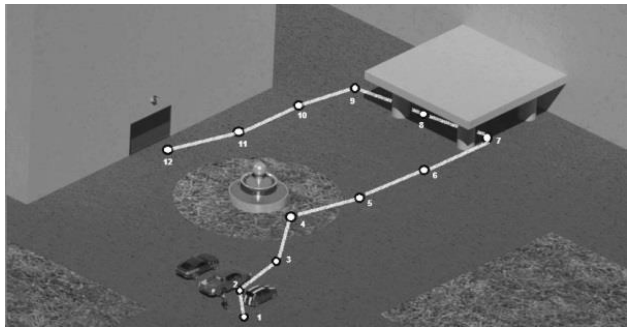

Fig. 8 Intruder's trajectory in a hypothetical scenario (i.e., house, enterprise, etc)

The PD fuzzy controller was used in this case to control the orientation of the robot during a simulation of the trajectory in Simulink using an initial position of 0.01 radians (roll), 0 radians (pitch) and 0 radians (yaw). The errors are shown in Fig. 9. In [9] it is demonstrated that the platform can reach different orientations with the same length of the legs. Since the kinematic controller focuses on reducing the error of the legs and not that of the platform, the reason that the error increased at the beginning of the graph is that the platform was moving towards an incorrect position that has the same leg's

length. Note that the error curves behave well although it takes almost four seconds for the controlled yaw-axis to achieve its desired position.

Tab. 2 Data for creating intruder's path

Point	X(m)	Y(m)	Pitch(deg)	Yaw(deg)
1	30.8	20.8	-2.31	-34.03
2	23.6	15	-3.07	-32.44
3	22	5.6	-3.78	-14.28
4	16.5	0	-5.19	0
5	17.5	-6.5	-4.59	20.37
6	18	-13	-3.86	35.83
7	18.5	-21	-3.06	48.62
8	10	-21	-3.69	64.53
9	5.5	-18.8	-4.38	73.69
10	4.5	-13	-6.22	70.9
11	3.5	-6.5	-11.48	61.69
12	2	-3	-22.58	56.31

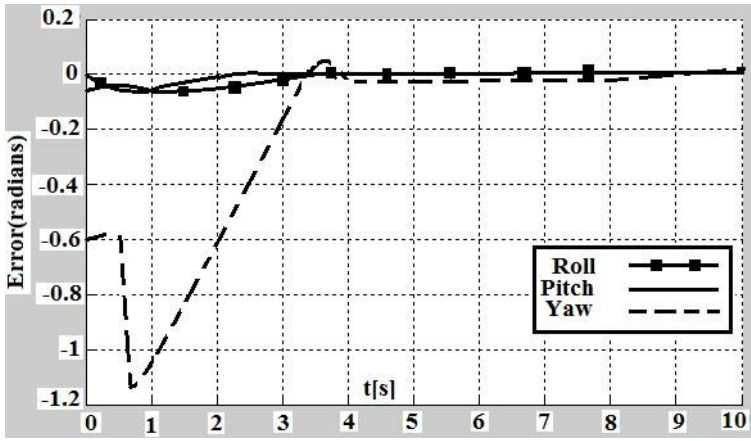

Fig. 9 Platform's error from a PD fuzzy controller (ω = roll, ϕ = pitch, ψ = yaw).

9.3 P-Sliding Mode Controller as Equivalent of a PD-Fuzzy Controller

A proportional fuzzy sliding mode controller is a fuzzy controller which takes advantage of the sliding mode regime, first defined for classical (non-fuzzy but crisp) controllers [21] but modified later to work for fuzzy controllers [17, 20]. The main advantage is that the sliding variable v, $v = \dot{\tilde{q}} + \lambda\tilde{q}$, $\lambda > 0$ permits to define a fuzzy decision vector for each limb in terms of only one variable *v* instead of considering a fuzzy decision matrix as in the fuzzy PD case (compare Tab. 2 and Tab. 3). Thus, tuning process is easier here because we tune a vector v observing w performance (see details in [20], [21]). This is possible as a result of closing the loop around the robot considering the fuzzy controller as a static sector-bounded nonlinearity N(v) which transforms the Euler – Lagrange equation (in terms of generalized coordinates) to a set of Euler – Lagrange equations but in terms of the fuzzy sliding variable *v* (Fig. 10). Compare Fig. 10 with Eq. 18 and the PD controller defined before. Moreover, global stability/tracking analysis was assured by

passivity–based theory and Lyapunov's theory in this case [21, 22]. As a consequence, accuracy could be warranted.

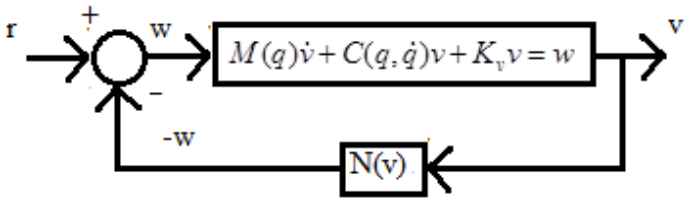

Fig. 10 Resulting closed loop system in terms of v, the fuzzy sliding variable

Tab. 3 P-fuzzy controller for each limb

Sliding condition v	Control output w
PB	PB
PM	PM
PS	PB
ZE	ZE
NS	NS
NM	NM
NB	NB

After implementing the system shown in Fig. 10, the resulting performance is described by the all converging to zero curves given in Fig. 11. Plant's response speed was shaped from the tuning process. It is possible to deal with robustness issues as parametric uncertainty or non-modelled dynamics as a result of the definition of fuzziness in these controllers (decision tables/vectors). See details in [20, 22].

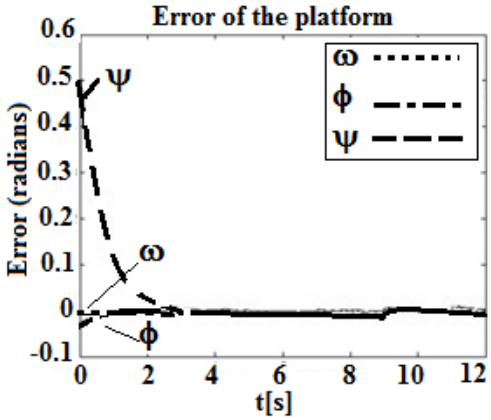

Fig. 11 Error in the platform due to a fuzzy sliding mode controller

10. Conclusions

Parallel manipulators can be used as orientation mechanisms due to their high stiffness and accuracy. Wrists that have the base and the platform connected by a spherical joint

have the advantage of always presenting pure spherical motions; furthermore, they are more tolerant to manufacturing errors than other three DOF parallel wrists. Although there are both analytical and geometrical methods to calculate the forward kinematics, they return more than one solution and therefore cannot be used to keep track of the orientation of the platform during a trajectory. The local forward kinematics method proposed was able to correctly predict the position of the platform. It is important to correctly train the artificial neural network; otherwise the method will likely present some errors.

The optimization method proposed successfully incremented the workspace and kept the size of the manipulator not that far from the desired, however, it may be possible to further increment the workspace by changing the fitness function parameters in the genetic algorithm. It is possible to control the manipulator by using just the kinematic analysis, but while this may work on some applications like a surveillance robot, in other applications like sentry devices a control using the manipulator dynamics is advised, therefore it is important to further analyse this robot.

Inverse and forward dynamics were also deduced in this paper for a parallel robot designed to work as a sentry device.

The resultant dynamical model was obtained in terms of Newton – Euler equations. Such equations were transformed to a more suitable representation for control, the Euler – Lagrange equations. The three control systems were able to orient the manipulator. For illustrative purposes, a test scenario in which an intruder is trying to enter a factory was created. Although the three controllers had a good performance, the classical (crisp) PID showed the poorest performance with respect to the other two, which were able to follow the intruder along the path. Using a fuzzy PID controller, such tracking was faster than the fuzzy sliding mode controller's, but the fuzzy sliding mode controller produced smaller errors, when the intruder changed his trajectory. Workspace-based optimal fuzzy controllers can also be designed and this can be investigated elsewhere.

References

[1] GOSSELIN, C., *These de doctorat McGill University*, [PhD Thesis]. Montreal: McGill: University, Canada, 1988. 115 p.

[2] DI GREGORIO, R., Statics and singularity loci of the 3-UPU wrist, *IEEE Trans. Robot*, 2001, vol. 20, no. 4, p. 630–635.

[3] DI GREGORIO, R., A new parallel wrist using only revolute pairs: the 3-RUU wrist, *Robotica*, 2001, vol. 19, no. 3, p. 305–309.

[4] HERVÉ, J. M, and KAROUIA, M., A family of novel orientational 3-DOF parallel robots. In *Proc. RoManSy 14*. Udine: Italy, Int. Centre for Mech. Sc., 2002, p. 359–368.

[5] KONG, X., and GOSSELIN, C., Type synthesis of 3-dof spherical parallel manipulators based on screw theory, *Journal of Mech. Design*, 2004, vol. 126, no.1, p. 101-108.

[6] ZLATANOV, D. and BONEV, I. and GOSSELIN, C., Constraint Singularities as C-Space Singularities. In *Proc. 8th Int. Symp. on Adv. in Robot Kinematics*. Caldes de Malavella: Spain, Adv. Robot Kin., 2002, p. 1-10.

[7] CUI, G., and ZHANG, Y., Kinetostatic Modeling and Analysis of a New 3-DOF Parallel Manipulator. In *Proc. IEEE Int. Conf. on Comp. Int. and Software Eng.* Wuhan: China, IEEE, 2009, p. 1–4.

[8] FINN, A. and SCHEDING, S., *Developments and Challenges for Autonomous Unmanned Vehicles: A Compendium*. Berlin: Springer, 2010. 230 p.

[9] GAN, D., DIAS, J., and SENEVIRATNE, L., Design and Analytical Kinematics of a Robot Wrist Based on a Parallel Mechanism. In *Proc. IEEE World Automation Congress*. Puerto Vallarta: Mexico, IEEE, 2012, p. 1-6.

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

[11] TSAI, K. and LIN, J., Determining the compatible orientation workspace of Stewart-Gough parallel manipulators, *Mechanism and Machine Theory*, 2006, vol. 41, no. 10, p. 1168–1184.

[12] HUDA, S. and TAKEDA, Y., Dimensional Synthesis of 3-URU Pure Rotational Parallel Mechanism with Respect to Singularity and Workspace. In *Proc. 12th Int. Fed. for the Theory of Mech. and Mach.World Congress*. Besançon: France, C. Français Prom. Sc. Méc. et des Mach., 2007, p. 1–6.

[13] KHAN, S., and ANDERSSON, K., Optimal Design of a 6-DoF haptic device. In *Proc. IEEE Int. Conf. on Mech.* Istanbul: Turkey, IEEE, 2011, p. 713–718.

[14] LOU, Y. and LIU, G. and LI, Z., Randomized optimal design of parallel manipulators, *IEEE Trans. Autom. Sci. Eng*., 2008, vol. 5, no. 2, p. 223–233.

[15] WANG, Z. et al, Optimal design of a linear delta robot for the prescribed cuboid dexterous workspace. In *Proc. IEEE Int. Conf. on Rob. and Biomimetics*. Sanya: China, IEEE, 2007, p. 2183–2188.

[16] ZHANG, L. and SONG, Y., Optimal design of the Delta robot based on dynamics. In *Proc. IEEE Int. Conf. Rob. and Autom.* Shanghai: China, IEEE, 2011, p. 336–341.

[17] DRAIANKOV, D., et al, *Introduction to Fuzzy Control*, Berlin: Springer, 1996. 215 p.

[18] CHAPARRO, D. and ZAVALA-YOÉ, R. and RAMÍREZ-MENDOZA, R., Kinematic and Workspace Analysis of a Parallel Robot used in Security Applications. In *Proc. of the 2013 IEEE Int. Conf. on Mech., Elec. and Automotive Engineering*. Morelos: Mexico, IEEE, 2013, p. 3-8.

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

[20] ZAVALA-YOE, R., Fuzzy Control of Second Order Vector Systems: L2 stability. In *Proc.of the 4th European Cont. Conf.* Brussels: Belgium, IEEE, 1997, p.1-6.

[21] SLOTINE, J.J. and LI, W., *Applied Nonlinear Control*, New Jersey: Prentice Hall, 1991. 461 p.

[22] CHAPARRO, D. and ZAVALA-YOÉ, R. and RAMÍREZ-MENDOZA, R, Dynamics and Control of a 3SPS-1S Parallel Robot Used in Security Applications., In *Proc. 21st Symposium on Mathematical Theory of Networks and Systems*. Groningen: The Netherlands, Univ. Groningen, 2014, p. 1-6.