Int j simul model 14 (2015) 1, 5-16
ISSN 1726-4529 Original scientific paper

SIMULATION OF THE HEXAPOD ROBOT PTINTO WALKING ON IRREGULAR SURFACES

Munoz, P.[*]; Castano, B.[**] & R-Moreno, M. D.[*]
[*] Computer Engineering Department, [**] Physics and Mathematics Department
University of Alcalá, 28801 Alcalá de Henares, Madrid, Spain
E-mail: pmunoz@aut.uah.es, bonifacio.castano@uah.es, malola.rmoreno@uah.es

Abstract
In this article we present our recent work on the simulation of a hexapod robot called PTinto. This robot is a prototype that has been designed to test the six legs locomotion system in the surrounding areas of the Tinto River (Huelva-Spain), which scientists have considered that could potentially have many similarities to the Mars surface. This kind of robots represents a great advance in the planetary research that overcomes the performance of the usual rover when operating in rocky and cumbersome areas. We are developing some of the software PTinto requires to autonomously control it and, in particular, the program that simulates its movements on any surface. Up to now we have a graphic computer program, developed in MATLAB, that represents the robot and allow us to check its movements in a lot of detail. We present in this paper all the elements and resources we use for the simulation and control of PTinto walking on irregular surfaces. Thanks to this simulation program, we could test different autonomous control strategies for the PTinto robot.
(Received in October 2013, accepted in July 2014. This paper was with the authors 3 months for 1 revision.)

Key Words: Walking Robots, Hexapod, Movements Simulation, Irregular Surfaces

1. INTRODUCTION

The use of simulators in robotic systems has been widely proven to be very helpful to test new algorithms and techniques, independently of the hardware availability [1-3]. This article presents all the elements and calculations needed to simulate the movement of a hexapod robot (a six legs spider) called PTinto when it walks through an irregular terrain. The whole simulation system is being developed using MATLAB [4] and is adapted specifically to the geometric characteristics and dynamics of this particular robot. The main interest of this work is that it could be used as a step-by-step guide to implement a suitable simulation system for any other robot similar to PTinto.

This robot is being designed at the Centre for Astrobiology (CAB) of the Institute of Aerospace Technology (INTA) in Madrid (Spain). PTinto is going to be tested in the shallow and inaccessible waters of the Tinto River at Huelva (Spain) before introducing this kind of walking technology in the Mars exploration program. This river is a unique place in the world for the study of biodiversity in extreme conditions (high pH, heavy metals). In this context, the PTinto project aims to create an autonomous and intelligent robot [5, 6] capable of moving over the uneven surface of the Tinto River banks and whose technology could be used on future missions to Mars.

This kind of legged robots is a very useful alternative to the traditional robots moving on wheels or tracks [7-9]. The main reason for this advantage is that rovers became inefficient when ground conditions (e.g. in sandy or boggy regions) are unfavourable. In that case, legged robots are able to operate satisfactorily in this type of land where the rovers are not. Walking robots are used in many applications because of their motion characteristics: high mobility, greater ability to overcome obstacles and adaptability to all fields. In this situation, it is very important the interaction between the robot and the environment, in our case, the only information the robot gets about the land is when its feet lean on the floor or hit against an obstacle.

This article is the continuation of our previous work that described the basic movements of PTinto. These movements were *walk* and *make rotations* on the level floor [5]. Now, we tackle more advanced and realistic situations related to the robot operational performance. First of all, we recall the robot geometrical features and the mathematical elements we use in our simulation. Then, in section 3 we review the main aspects of the simulation of the robot elementary movements on a flat ground. In section 4, we explain how to describe an irregular soil. And, finally we show in great detail all the new features we have developed in our investigation about PTinto evolving on uneven ground. Section 5 shows the main screen and features of our MATLAB program, and the article ends with some conclusions and future work proposals.

2. GEOMETRICAL DESCRIPTION

In this section we describe the geometrical layout of PTinto. Fig. 3 shows the main screen of our MATLAB program and a schematic representation of the robot in the rest position. It can be seen that the robot's body is a hexagonal prism and each one of its legs is located at the prism vertical edges. The vacuum inside the body will be used to place both components that allow its movements and the potential burden to carry in terms of its use.

2.1 PTinto layout

In the rest position the total length is 168 cm, the maximum width 233 cm and the height 143 cm. Each one of its six legs consists of two straight sections of fixed length joined by other two sections of varying length. The change in length for each variable component is achieved by a step-by-step motor. These changes allow the leg to go up and down. Furthermore, the connection between the body and each leg has a third engine that changes the angle between the plane of the leg and the body allowing the leg moves forward or backward.

We show a schematic representation of a PTinto's leg in Fig. 1. We also display the name of its main elements: the three angles (A_1, A_2 and A_3), the insertion points in the body (V_1 and V_2), the knee (K) and the foot (F). It is important to realize all the elements of each leg are always in the same plane.

A leg has three degrees of freedom. These are, on the one hand the three angles that determine exactly its position and the foot coordinates. And, on the other hand the three feet coordinates that bind the angles values.

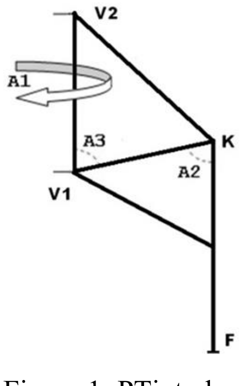

Figure 1: PTinto leg.

These three angles have a range of possible values. This range is shown in Table I.

Table I: Angle ranges.

Angles	A_1	A_2	A_3
Minimum	-30	60	60
Maximum	30	90	90

The angle A_1 is special, its value is positive if the leg is in an advanced situation relative to the body and negative in the opposite case.

Each time we compute a new position for one foot, we check if the new angles are inside these ranges. If they are, we accept that position, otherwise we reject it and leave the robot in its previous state (we say the new position is inadmissible or inacceptable).

2.2 The reference systems

In order to control the movement, location and spatial orientation of a robot like Ptinto we have developed two different rectangular coordinates system: one of them called “global system” is fixed in the ground and the other known as “local system” is fixed in the robot's body.

The global system corresponds to a fixed Cartesian system in which the robot moves [10] and the local system has its origin of coordinates at the geometric centre of the robot's body and its three perpendicular axes coincide with its three main directions.

In the robot rest position, also called starting position, the leg angles values are in the centre of its total range and the robot's feet are leaned on the $Z=0$ plane of the global system. At the rest position the global and local coordinate axes are parallel, have the same orientation and the origin of the local coordinate system in relation to the global system is (0, 0, H), $H=114{,}26$ cm. H is the height of the centre of the robot's body over the plane in the global system $Z=0$.

In general, we will compute changes upon feet positions and the corresponding leg angles in the local system and then, we will transform them into the global system. This requires always knowing the position of the local origin of coordinates and an orthonormal local basis in the global system. We represent the local origin as:

$$\overrightarrow{Ol} = (Ol_x, Ol_y, Ol_z) \tag{1}$$

and an orthonormal local basis as:

$$\mathrm{B} = \left\{\overrightarrow{Lx}, \overrightarrow{Ly}, \overrightarrow{Lz}\right\} \tag{2}$$

where:

$$\overrightarrow{Lx} = (Lx_x, Lx_y, Lx_z),\ \overrightarrow{Ly} = (Ly_x, Ly_y, Ly_z),\ \overrightarrow{Lz} = (Lz_x, Lz_y, Lz_z) \tag{3}$$

The objects in the local coordinate system will be named in small letters and in the global one in capitals. That is, j foot coordinates in local and global systems will be:

$$\vec{f}(j) = (f(j)_x, f(j)_y, f(j)_z), \quad \vec{F}(j) = (F(j)_x, F(j)_y, F(j)_z) \tag{4}$$

In this context if:

$$\vec{R} = \left(R_x, R_y, R_z\right) \text{ and } \vec{r} = \left(r_x, r_y, r_z\right) \tag{5}$$

are the position vectors of the same point in the global system and local system respectively, there are the following relations between them:

$$\vec{R} = \vec{r}\begin{pmatrix} Lx_x & Lx_y & Lx_z \\ Ly_x & Ly_y & Ly_z \\ Lz_x & Lz_y & Lz_z \end{pmatrix} + \overrightarrow{Ol} \quad \text{and} \quad \vec{r} = \left(\vec{R} - \overrightarrow{Ol}\right)\begin{pmatrix} Lx_x & Ly_x & Lz_x \\ Lx_y & Ly_y & Lz_y \\ Lx_z & Ly_z & Lz_z \end{pmatrix} \tag{6}$$

2.3 Elementary movements

In our previous work [5] we showed all the details about the elementary movements of PTinto.

These movements were two: walk forward and rotate on a flat ground.

In that work we paid a big attention to the relations between the two set of degrees of freedom for each leg: the three angles A_1, A_2 and A_3 and the three foot coordinates f_x, f_y and f_z. When we compute the angles of a leg from the coordinates of its foot we call it *inverse geometry* and in the opposite case *direct geometry*.

At any time, we develop a strategy that we follow in this work. First of all we start from an admissible position, then we compute the feet coordinates for a new position, according to the movement we want to simulate, and then we check if the new angles are admissible. If they are, we accept the changes, otherwise we reject them.

This article extends this approach to more complicated movements on an irregular surface.

3. GROUND DEFINITION

The next step in this work is to describe an irregular surface and determine how the robot interacts with it. In this article we have chosen a floor formed by square blocks 50 cm wide. The size of the total ground covered by them will be large enough for the robot to take some steps on it. The height of each one of these blocks will be specified using an array of integer values where each item coordinates (x, y) will correspond with the location of the bottom-right corner of the block in the global system. The initial ground in the starting position will be flat and negative heights are not considered. The first six lines of the matrix that defines the ground of Fig. 2 are as follows:

$$\begin{pmatrix} 5 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 10 & 5 & 0 & 0 & 0 & 5 & 5 & 5 & 5 \\ 10 & 10 & 10 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 20 & 20 & 10 & 10 & 10 & 10 & 10 & 10 & 5 & 5 \\ 10 & 20 & 20 & 20 & 10 & 10 & 10 & 20 & 10 & 5 \\ 10 & 10 & 10 & 10 & 10 & 20 & 20 & 20 & 10 & 5 \end{pmatrix} \tag{7}$$

The interaction between the floor and the robot feet will be accomplished using a function that calculates, in the global system, the height of the ground for each position of a robot foot $\vec{F}(j) = (F_x(j), F_y(j), F_z(j)), \ \forall 1 \le j \le 6$. In addition, each foot is assumed to have a 4 cm × 4 cm square cross section (we will call it footprint). Then, for each foot j, we calculate the ground heights for the four corners of the footprint as follows:

$$(F_x(j)+2, F_y(j)+2), (F_x(j)+2, F_y(j)-2), (F_x(j)-2, F_y(j)+2), (F_x(j)-2, F_y(j)-2) \tag{8}$$

and the highest one of these four values will be the floor level under that foot. We will call this number $Floor(\vec{F}(j))$ and we determine whether foot j is under or above the floor, by comparison with $F_z(j)$.

In order to avoid the legs hit against the sides of the blocks all the movements of the feet (up and down) are strictly vertical.

Fig. 2 shows an image of PTinto going through the ground described by the previous matrix. We can appreciate the blocks making up the ground and the small cross at the end of some of the legs that means the leg is leaned.

4. MOVEMENTS ON AN UNEVEN GROUND

After describing the ground, this article deals with the problem of the robot movement on it. At the very beginning, we assume that the robot starts in the rest position and over a flat floor. Then, the robot will walk straight ahead always in the same direction. All along its transit on this irregular floor, PTinto will adapt its position to the characteristics of the environment and will try to keep the angles of their legs closer to the equilibrium, i.e. their medium value. In this way it will have more freedom of movement. The robot will always have at least three alternate legs resting on the ground. According to the numbering of the legs, the movement scheme will be: odd legs (1-3-5) leaning on the floor and even legs (2-4-6) lifted or vice versa. Between each change of support there will be a situation where the robot will be resting on its six legs.

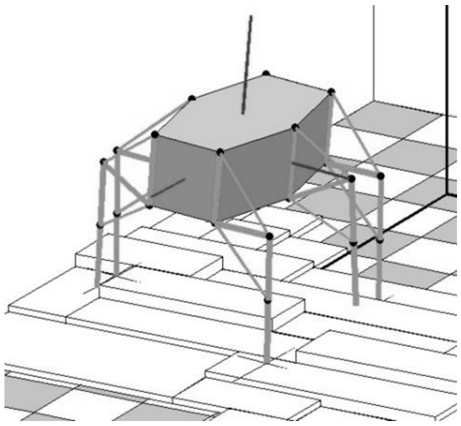
Figure 2: PTinto walking on an irregular surface.

A complete pace forward of the robot will be divided into a series of five individual and independent movements. This independence is not strictly required; however, we consider it in the current development to simplify the global study.

The five individual movements, or movement sections, that make up a complete pace of PTinto are:

1. Raise feet: odd or even.
2. Progress on the leaned feet.
3. Bringing down raised feet.
4. Relocate angles.
5. Relocate height.

These five movements can be grouped into two different actions. The first three ones make up a complete forward motion between two fully six-legged supported positions and the second two represent the relocation of the PTinto's body, after a full step to get an intermediate position.

To manage all of these movements and the different situations we define three vectors. These vectors determine the performance of the legs of the robot at any moment. These three vectors are:

1. *Supp*(j), $1 \leq j \leq 6$. The elements of this vector take binary values. *Supp*(j) = 1 means foot j leans on the ground (i.e. the robot is resting on it) and *Supp*(j) = 0 indicates foot j is lifted.
2. *Adv*(j), $1 \leq j \leq 6$. The elements of this vector take the values: 1, 0, and –1. These values indicate respectively: the foot can move forward, backward or must be motionless.
3. *GoUp*(j), $1 \leq j \leq 6$. The elements of this vector can take the values: 1, 0, and –1. These values indicate respectively: the foot can go up, down or must be motionless.

The movements will be done in a step-by-step process with the aim of simulating the operation of the step by step robot engines. One step will usually be one centimetre for linear movements and one sexagesimal degree for rotations. These values are stored in the variables *Step* and *Deg*. Before each step, we will assume that PTinto is in an admissible position and we will accept the resulting position, after each step, according to its feasibility.

In the following paragraphs, we describe in detail each one of the individual movements listed above.

4.1 Raise feet: odd or even

This is the first movement of the series. The starting position of PTinto is the robot resting on its six legs whether on a flat floor or on an uneven ground.

For this movement we have to determine which group of legs, odd ones or even ones, must rise and which must remain on the floor. If the robot starts at the rest position we can choose any set of legs, otherwise the election depends on the previous movement. This option will be monitored by the vector *Supp*. For instance, if we decide to raise odd legs this vector has to be: *Supp* = (1,0,1,0,1,0). Besides, *GoUp* has to be *GoUp* = (–1,1,–1,1,–1,1).

We should take into account two very important things:

1. In order to lift the raising feet to maximum height, raising legs will shrink and leaning ones will stretch out.
2. In order to avoid hitting the side faces of the ground blocks, the movement of the raising legs will be strictly vertical in the global system (i.e. along the global vector $\vec{Z}=(0,0,1)$, that is vector $\vec{z}=(Lx_z, Ly_z, Lz_z)$ in the local system).

In that way, if $\vec{f}(j)_i$ is the initial position of *j* foot the final one $\vec{f}(j)_f$ is computed as:

$$\vec{f}(j)_f = \vec{f}(j)_i + GoUp(j)\cdot Step\cdot\vec{z} \tag{9}$$

If $GoUp(j)\neq 0$ a new foot vector $\vec{f}(j)_f$ will be obtained, then the corresponding leg angles $A_1(j)$, $A_2(j)$ and $A_3(j)$ will be calculated and, if all of them are inside their range, both foot coordinates and leg angles will be updated. Otherwise, foot coordinates and leg angles will not be accepted, we will set $GoUp(j)=0$ and from this moment on the leg *j* will be motionless, in the local system, until the end of this elementary section.

If the stopped leg *j* is leaned then the lifting movement of the robot's body will be also ended and only the raised legs will be able to follow their motion. When the stopped leg is a lifted one, it will remain still as well.

When the feet positions are updated and only if the leaned legs are not blocked we have to lift vertically the body a distance equal to *Step*. In that way leaned feet will remain in the same global place. This can be accomplished by the expression:

$$Ol_{z_f} = Ol_{z_i} + Step \tag{10}$$

where Ol_{z_i} and Ol_{z_f} represent the third coordinate of the centre of the body before and after lifting the robot.

After one step is completed, and all the elements of the robot have been calculated in the local system, if there is any variation with respect to the previous position, we perform a change coordinates into the global system and represent the robot in the new position.

This process continues until all of the legs have reached the end of their movements ranges.

4.2 Progress on the leaned feet

In this section we describe the forward movement of PTinto's body by the action of the engines of its leaned legs moving them backwards with regard to the body. At the same time, the engines of the raised legs will move forward. As in the previous section this movement is also divided into a set of elementary steps.

In order to control this forward or backward movement we use the vector *Adv*. This vector indicates the type of movement for each foot. Its behaviour is analogous to the *GoUp* vector described in the previous movement section: *Adv* = 1 means the movement of the *j* foot is forward, *Adv* = –1 backwards and *Adv* = 0 the *j* foot is motionless. When all values *Adv*(*j*),

$1 \leq j \leq 6$ become null this section will be ended, PTinto will be stopped and the simulation will move on the next section.

The forward and backward movement of each leg is made in accordance to the vector $\vec{x} = (1,0,0)$ in the local system (vector $\overrightarrow{Lx} = (Lx_x, Lx_y, Lx_z)$ in the global system). This choice makes the robot to move forward along a straight line in the direction of the *X*-axis of the local system.

In this context, the new position of the *j* foot according to the movement described above is expressed by the expression:

$$\vec{f}(j)_f = \vec{f}(j)_i + Adv(j) \cdot Step \cdot \vec{x} \tag{11}$$

Then, after getting the new coordinates for the six feet, we compute the angles to place each foot in its new place. If the six angles are admissible we accept the new values. Otherwise, we have two possible situations. If a leaned leg cannot acquire its computed angles we return all the leaned feet to their previous places and stop the motion of the robot. In the case of a raised leg *j* we only return it to its previous position and stop the leg by setting $Adv(j) = 0$.

In addition, there are another two situations that stop the advance of the robot. These are:

1. When any of its raised feet hits against the front of a block because it is too high to be overcome.
2. One foot hits the floor.

If we consider leg *j*, we detect both situation by comparison between the global foot coordinate $F_z(j)$ and the ground height $Floor(\vec{F}(j))$ under it. If $F_z(j) < Floor(\vec{F}(j))$ (the foot is under the ground) we discard all new positions, stop the movement of the robot and wait for the next movement section.

If the movement of the robot is not stopped, we must advance its body a *Step* distance in the direction of the global system vector (Lx_x, Lx_y, Lx_z). This is done by the expression:

$$\overrightarrow{Ol}_f = \overrightarrow{Ol}_i + Step \cdot \overrightarrow{Lx} \tag{12}$$

obviously, $\overrightarrow{Ol}_i$ and $\overrightarrow{Ol}_f$ are the vectors of the centre of PTinto in the global system, before and after the advance movement.

As well as in the previous movement section we compute the new position of the robot in the global system and make a graphical representation.

This movement can reach a total impasse when one of the raised legs can't get over an obstacle after trying twice. In that case the simulation algorithm stops and we admit that we have come to a point that needs more advanced treatment than the presented in this paper.

4.3 Bringing down raised feet

This action ends a complete forward movement of the robot. In this section all the raised legs are lowered until they reach the ground. At the same time, the leaned ones are shrunken in order to bring down the robot's body. The movement ends with the robot leaning on its six legs.

It may also happen that any of the lifted legs go down without reaching the ground. In that case the robot would be blocked and the simulation would end. This situation would require a more advanced treatment than the outlined in this paper.

In this movement section the feet will move vertically in the global coordinate system to avoid hitting the sides of the blocks of the ground. The vectors that direct the movement are $\vec{Z} = (0,0,1)$ in the global system. We also use the vector $GoUp(j)$ (i.e. if leg *j* is leaned $GoUp(j) = 1$ and if leg is lifted $GoUp(j) = -1$), the variable *Step* and the motion equations are the same as in the section 4.1:

$$\vec{f}(j)_f = \vec{f}(j)_i + GoUp(j) \cdot Step \cdot \vec{z} \tag{13}$$

After computing these new feet positions we check whether the values of the angles are possible or not. When possible we accept them. Otherwise, we rule out the wrong values and keep the corresponding leg in its previous position. Besides, if the leg is leaned we do not change any other supporting leg and stop the vertical movement of PTinto's body.

Otherwise, if the robot's body is able to move we charge its height by the expression:

$$Ol_{z_f} = Ol_{z_i} + Step \tag{14}$$

In this section we have also considered that when a raised leg reaches eventually the ground, it becomes leaned and its foot gets the height of the ground under it. It is important to realize that because the movements of the legs are strictly vertical, the height of the ground beneath each one is the same throughout the entire movement.

To deal with this matter, whenever a new height $F_z(j)$ is computed for a raised j leg in the global system we compare this value with the ground level $Z_G = Floor(\vec{F}(j))$. Then if $F_z(j) \leq Z_G$ we set $F_z(j) = Z_G$ and consider the j leg resting on the floor.

This movement section ends when all the raised legs are leaned or when any of them has gone down without getting a support and is unable to descend more because it has exhausted the range of its angles.

4.4 Adaptation of angles

In this movement section we relocate the robot so that the body became parallel to the plane determined by the three feet going to be leaned during the next advance. This movement is a rotation of the robot around its geometric centre. This supposes a change of the local reference system, but in such a way the robot maintains the direction of its forward motion. So we have to compute the new local basis and the rotation angle θ between the current and the new basis. After that, we will divide θ into equal elementary angles of amplitude $Deg \leq 1°$ and perform the complete rotation step by step.

In this change, the feet will not change their position in the global system, so that we have to turn them, in the local system, in the opposite direction of the robot body to simulate its rotation movement.

To begin with this section we have to determine the three feet that are going to remain leaned in the next full step. Those three feet define a plane and the robot body must be placed parallel to it. If we assume, for instance, that these three feet are the odd ones, we can calculate the perpendicular vector to the plane defined by the feet as follows:

$$\vec{N} = \frac{(\vec{F}(3) - \vec{F}(1)) \times (\vec{F}(5) - \vec{F}(1))}{\left\| (\vec{F}(3) - \vec{F}(1)) \times (\vec{F}(5) - \vec{F}(1)) \right\|} \tag{15}$$

where × is the cross product. This is one of the three new local basis vectors.

The second vector is the one in the direction of the motion. Because it must be within the global XZ-global plane and perpendicular to $\vec{N}$, we compute it as follows:

$$\vec{X} = \frac{(N(3), 0, -N(1))}{\left\| (N(3), 0, -N(1)) \right\|} \tag{16}$$

and the third local basis vector is:

$$\vec{Y} = \vec{N} \times \vec{X} \tag{17}$$

So, the new orthonormal local basis is:

$$\mathrm{B}' = \left\{ \vec{X}, \vec{Y}, \vec{N} \right\} = \left\{ \overrightarrow{Lx}', \overrightarrow{Ly}', \overrightarrow{Lz}' \right\} \tag{18}$$

The rotation matrix for changing from the old to the new basis is:

$$\mathrm{M} = (\mathrm{B})^t \times \mathrm{B}' \tag{19}$$

where $(\mathrm{B})^t$ is the transpose of matrix B and × the product of matrices.

This rotation matrix M has an associate axis $\vec{E}$ and an angle of rotation θ that we compute as follows:

$$\theta = 2\cdot\cos^{-1}(t_1) \text{ and } \vec{E} = \frac{(t_2,t_3,t_4)}{\sin(\theta/2)} \tag{20}$$

where

$$t_1 = \frac{\sqrt{|\mathrm{M}(1,1)+\mathrm{M}(2,2)+\mathrm{M}(3,3)+1|}}{2}$$
$$t_2 = \mathrm{sign}(\mathrm{M}(3,2)\text{-}\mathrm{M}(2,3))\frac{\sqrt{|\mathrm{M}(1,1)\text{-}\mathrm{M}(2,2)\text{-}\mathrm{M}(3,3)+1|}}{2}$$
$$t_3 = \mathrm{sign}(\mathrm{M}(1,3)\text{-}\mathrm{M}(3,1))\frac{\sqrt{|\text{-}\mathrm{M}(1,1)+\mathrm{M}(2,2)\text{-}\mathrm{M}(3,3)+1|}}{2}$$
$$t_4 = \mathrm{sign}(\mathrm{M}(2,1)\text{-}\mathrm{M}(1,2))\frac{\sqrt{|\text{-}\mathrm{M}(1,1)\text{-}\mathrm{M}(2,2)+\mathrm{M}(3,3)+1|}}{2} \tag{21}$$

Now we are ready to start the simulation of the robot rotation. We chose to rotate the body by an angle θ around the axis $\vec{E}$ and the feet the same angle around the axis $-\vec{E}$, we divide this angle into equal small angles *Deg* and use quaternions [11] to perform rotations.

An elementary rotation around axis $\vec{E}$ can be achieved by the quaternion:

$$Q = (\cos(\frac{Deg}{2}), \sin(\frac{Deg}{2})\cdot\vec{E}) \tag{22}$$

Then the new orthonormal $\{\overrightarrow{Lx_f}, \overrightarrow{Ly_f}, \overrightarrow{Lz_f}\}$ basis after rotating the basis $\{\overrightarrow{Lx_i}, \overrightarrow{Ly_i}, \overrightarrow{Lz_i}\}$ will be computed by the expressions:

$$\begin{aligned}(0,\overrightarrow{Lx_f}) &= Q\circ(0,\overrightarrow{Lx_i})\circ Q^*\\ (0,\overrightarrow{Ly_f}) &= Q\circ(0,\overrightarrow{Ly_i})\circ Q^*\\ (0,\overrightarrow{Lz_f}) &= Q\circ(0,\overrightarrow{Lz_i})\circ Q^*\end{aligned} \tag{23}$$

where Q^* is the quaternion conjugate and $\circ$ the quaternion product.

Respectively, for each foot *j* the change can be computed with the quaternion:

$$-Q = (\cos(\frac{Deg}{2}), \sin(\frac{Deg}{2})\cdot(-\vec{E})) \tag{24}$$

and the new foot coordinates $\vec{f}(j)_f$ as:

$$(0,\vec{f}(j)_f) = -Q\circ(0,\vec{f}(j)_i)\circ(-Q)^* \tag{25}$$

Similarly to the previous sections, we calculate all the angles for all the legs and, if they are admissible, we accept the change and obtain the new coordinates of the robot. Then we draw it in the new position. We follow this process, step by step, until we complete the angle θ.

If at any moment any of the angles is not admissible, we leave the robot in the previous position and continue the process with the next movement section.

4.5 Relocate height

Now we are going to relocate the robot in height so that the angles of its legs will get an intermediate position and it will be able to tackle the next movement with maximum guarantees of amplitude and stability. We have to take into account that in the starting position the geometric centre of the robot is 114,26 cm above the plane of the six legs.

At this moment, the robot has six legs leaned and three of them will stay on the ground to undertake the next step. The feet of these three legs determine a plane and now the robot's body is parallel to it. In this position, PTinto will move its body up or down to get the average *z* coordinate of all its feet equal to 114,26 cm in the local system. This movement represents a change in the position of the feet and body in the direction of the local *z*-axis.

The overall vertical distance *d* we have to move the robot is computed as:

$$d = \frac{1}{6}\left(\sum_{j=1}^{6} f(j)_z\right) - 114,26 \tag{26}$$

As before, we divide *d* into small equal distances called *Step* and change the *z* coordinate of each foot as follows:

$$f(j)_{zf} = f(j)_{zi} - Step \tag{27}$$

Then, we compute the new angles and if all of them are in range we accept the change, and move the PTinto's body the same distance *Step*, in the opposite direction, by the formula:

$$\overrightarrow{Ol}_f = \overrightarrow{Ol}_i + Step \cdot \overrightarrow{Lz} \tag{28}$$

After that, we update the coordinates into the global system and draw the robot. If any of the angles is not acceptable we discard the new coordinates, leave PTinto in the previous position and the current movement section stops.

This process continues until the robot completes the distance *d* or it reaches an inadmissible position.

5. THE IMPLEMENTATION

Using all the theoretical results described above, we have made a complete program in MATLAB that can manipulate the robot to perform all the movements described in this paper. Fig. 3 shows the main screen of the program with PTinto in the start position, the reference systems, the ground layout described by eq. (7) and the elements to control the robot evolution.

We have used the graphical MATLAB tool GUIDE [12] and we have created a user interface that allows us to control the robot and send the different movements order to perform. Under this control, PTinto evolves and the graphical interface presents a detailed picture of its movements. This simulation of PTinto will allow us to test control autonomous capabilities [13] in different kind of terrain, check how the robot evolves and the difficulties it could have without using the real robot.

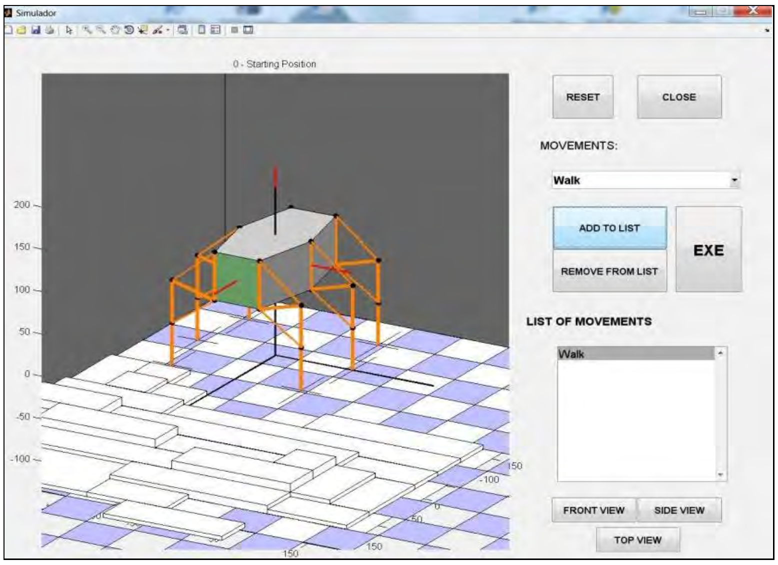

Figure 3: The main screen of the MATLAB simulation program.

With our software we are able to create a list of different movements and tell the robot to follow them. We can also change the floor layout into any square blocks configuration we want. For instance, it is possible to create a scenario with blocks that are too high to overcome for PTinto and see how it deals with them. This experimentation is very important because it will allow us to see the way PTinto interacts with a wide range of obstacles.

On top of that, the simulation program is able to display images of the PTinto motion with three different points of view: Front, Side and Top. This way we can study the position of the robot related to its environment in a great detail. This became very necessary when we want to see feet leaned on the ground, in this case we draw a small black cross at the end of each foot to indicate that the foot should be resting on the floor. Using this technique we are able to check the match between our mathematical computation and the graphical representation in every movement.

Another important issue we have to consider is how to avoid the "virtual robot" could be deformed through all software manipulations it has to undergo. This is done by checking, each time it moves, the rigidity of the body, the planarity of the legs and their relative positions with respect to the planes of the body. These extra calculations extend the computational effort of our program but they are absolutely essential to ensure its validity and applicability to the real robot.

6. CONCLUSIONS

In this paper we have shown our simulation program of a six-legged robot called PTinto moving on an uneven ground. We have described all the mathematical computation that we have used and the way we have implemented in a MATLAB program with the GUIDE graphical interface. We have addressed a lot of the purely kinematic robot motion aspects. Many of them have been settled and controlled but still there are others very important that we will address in future work.

First of all, it is necessary to solve the situation that appears when the robot finds an obstacle impossible to save (i.e. a block too high or too low) and detect it. In this context the robot has to be able to go back, to a previous situation, and searches for other alternatives, such as other directions of movement. One of these options could be the robot rotation on rough terrains.

Besides, it will be also important to study the dynamic aspects of the movement of the robot, that is, the forces and moments to be borne by the legs and the joints, between the moving parts, according to their intrinsic weight and the distribution of the load PTinto carries. In this context, it will be important to take into account the work demanded to the engines depending on the position of the legs and the weight it transports as well.

ACKNOWLEDGEMENT

This work was partially supported by the Junta de Comunidades de Castilla-La Mancha project PEII-2014-015A (PEII11-0079-8929).

REFERENCES

[1] Michel, O. (2004). Cyberbotics Ltd. – Webots^{TM}: Professional mobile robot simulation, *International Journal of Advanced Robotic Systems*, Vol. 1, No. 1, 39-42

[2] Poulakis, P.; Joudrier, L.; Wailliez, S.; Kapellos, K. (2008). 3DROV: A planetary rover system design, simulation and verification tool, *Proceedings of the 10th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Hollywood, 8 pages

[3] Curkovic, P.; Jerbic, B.; Stipancic, T. (2013). Coordination of robots with overlapping workspaces based on motion co-evolution, *International Journal of Simulation Modelling*, Vol. 12, No. 1, 27-38, doi:10.2507/IJSIMM12(1)3.222

[4] The MathWorks, Inc. MATLAB 7, Creating Graphical User Interfaces, from *http://www.mathworks.com/help/pdfdoc/matlab/buildgui.pdf*

[5] Muñoz, P.; R-Moreno, M. D.; Gallego, P.; Castaño, B. (2011). A cognitive architecture and simulation environment for the PTinto robot, *Proceedings of the 4th IEEE International Conference on Space Mission Challenges for Information Technology*, Palo Alto, 129-136

[6] Muñoz, P.; R-Moreno, M. D.; Castaño, B. (2010). Integrating a PDDL-based planner and a PLEXIL-executor into the PTinto robot, *Proceedings of the 23rd IEA-AIE 2010*, Part I (LNAI 6096), Córdoba, 72-81, doi:10.1007/978-3-642-13022-9_8

[7] Krzic, P.; Pusavec, F.; Kopac, J. (2013). Kinematic constraints and offline programming in robotic machining applications, *Technical Gazette*, Vol. 20, No. 1, 117-124

[8] Lopez, J.; Brenosa, J.; Galiana, I.; Ferre, M.; Gimenez, A.; Barrio, J. (2012). Mechanical design optimization for multi-finger haptic devices applied to virtual grasping manipulation, *Strojniski vestnik – Journal of Mechanical Engineering*, Vol. 58, No. 7-8, 431-443, doi:10.5545/sv-jme.2011.141

[9] Tomasic, T.; Demetlika, A.; Crnekovic, M. (2012). Self-balancing mobile robot tilter, *Transactions of FAMENA*, Vol. 36, No. 3, 23-32

[10] Budanov, V. M. (2007). Algorithms of motion planning for a six-legged walking machine, *Journal of Mathematical Sciences*, Vol. 146, No. 3, 5931-5937, doi:10.1007/s10958-007-0407-8

[11] Hamilton, W. R. (1969). *Elements of Quaternions*, Vol. I, 3rd ed., Chelsea Publishing Company, New York

[12] Marchand, P.; Holland, O. T. (2003). *Graphics and GUIs with MATLAB*, Chapman and Hall/CRC, Boca Raton

[13] Muñoz, P.; R-Moreno, M. D. (2013). Deliberative systems for autonomous robotics: A brief comparison between action-oriented and timelines-based approaches, *Proceedings of the ICAPS Workshop on Planning and Robotics (PlanRob 2013)*, Rome, 45-53