Differential Kinematics of a Multisection–Robot

Fabian C. Castro[1(✉)], Nicolás J. Walteros Vergara[2], Cesar A. Cardenas[1], Juan C. Amaya[1], Emiro-De-la-Hoz-Franco[3], Paola A. Colpas[3], and Carlos A. Collazos[1(✉)]

[1] Vicerrectoria de investigación, Universidad Manuela Beltrán, Bogotá, Colombia
{fabian.castro,carlos.collazos}@docentes.umb.edu.co
[2] Departamento de matematicas, Universidad Distrital, Bogotá, Colombia
[3] Departamento de Ciencias de la Computación y Electrónica, Universidad de la Costa, Barranquilla, Colombia

Abstract. The multi-section robots also called variable geometry robots (VGT), are formed by different modules and have multiple degrees of freedom (DOF); These robots are a new class that can be defined as systems adaptable to different environments, unlike conventional robots, multi-section robots allow greater flexibility and adaptability to carry out tasks with restricted space conditions, their locomotion has a high degree of manipulation and dexterity in environments with difficult access and very closed spaces where maneuverability must be high, these characteristics are very similar to those exhibited by the movements of snakes, elephant trunks, and octopus tentacles, capabilities beyond them reach of traditional handlers of rigid link, multi-link robots can adapt their shape to navigate through complex environments. In this work, we show the implementation of the Lie Matrix Theory of the rigid movements of a body in a multi-link Robot so that through kinematics and with the planning of trajectories through third-order polynomials this resembles curves smoothed by Bezier to generate different deformations in the robot in such a way that its movements elude obstacles in a given one within the workspace. The developed algorithm was implemented on a simulated virtual platform in a robotics environment. The motivation of the work was to be able to demonstrate a planning of robot trajectories with multiple degrees of freedom using deterministic algorithms and not focused on computational intelligence such as neural networks or reinforced learning.

Keywords: Multi-section robot · Screw transformation · Path planning · Curvature discretization

1 Introduction

In recent years, a growing interest in bio-inspired robotics has emerged, currently great advances have been made in numerous applications at an industrial level

O. Gervasi et al. (Eds.): ICCSA 2022, LNCS 13375, pp. 158–172, 2022.
https://doi.org/10.1007/978-3-031-10522-7_12

and in the field of medicine, continuous and multi section robots inspired by biological tubes, tentacles and snakes, can vary Its curvature, a feature called VGT (Variable Geometry Truss), has recently seen an increase in efforts aimed at taking advantage of these qualities to improve frontiers in the field of medicine mainly in minimally invasive surgical interventions. Several designs have now been commercialized, which are inspiring and allow a traditional change from the traditional robot that is in greater demand in today's market, however surgical approaches towards flexible access routes, for example, through natural orifices such as the nose, where The multi-section robots can easily access.
The main objective of this work is to introduce some mathematical - analytical concepts applied to model the kinematics of a particular case of Multisection Robot and a simulation that involves the ROS, MATLAB and GAZEBO computational environments, in such a way as to show the advantages of the implemented mathematical model. and visualize in 3d a construction of a robot with 20 cylindrical links 15 cm long and 1.5 cm in diameter with ball-type joints.

Exponential and its multiplications can parameters angles present in each joint of the device of this paper, the design here has a nice approach taking into account product of 4×4 square matrices at most. Bezier curves and splines are introduced to describe paths in the Multi section-Robot.

2 Rigid Body Movements

Rigid motions in the space are modeled in a efficient way using *Twist theory*. An affiance map $T : \mathbb{R}^3 \to \mathbb{R}^3$ is called a *Rigid motion* if: (1) preserve distances, and (2) preserve orientations (in the sense that, the map T preserves cross product [1,2]. Given O a non-empty subset of the space $\mathbb{R}^3$, the set of all rigid motions restricted on O are rotations followed by translations such that conforms the 6-dimensional Lie group: *the 3-Special Euclidean group* SE(3), where its Lie algebra vectors are named *Twists*. Let SO(3) the Lie group of rotations that preserve lengths in $\mathbb{R}^3$ named 3-*Special Orthogonal group*, *Chasles' theorem* means that all rigid motions can be viewed like a composition between a rotation and a translation about a fixed axis, hence, in symbols, the 3-Special Euclidean group is the semi direct product [3]

$$SE(3) = SO(3) \times R^3.$$

2.1 Twists Lie's Theory

Is straightforward that, each element of SO(3) is a rigid motion in the space. An important strategy here is introducing all element of SO(3) by pick a rotation axis and a measure of an angle. To give an adequate usage for coordinates, the first one step is fix an inertial frame and next, fix other positive oriented reference system about the rigid body with an axis parallel to the rotation axis, where, if p is an arbitrary point of a rigid body O rotated since a point $p(0)$ by a matrix R_{IO}, suppose p_I the coordinates of the point p respect to the inertial frame and

p_O the coordinates of p respect to the body frame [4]. The relation between the coordinates are given by the following formula

$$p_I = R_{IO} p_O.$$

Let $\mathfrak{so}(3)$ the Lie algebra of SO(3) which is re presentable by the skew-symmetric matrices with Lie bracket the commutator such that is isomorphic to the natural Lie algebra structure of $\mathbb{R}^3$ with cross product operation. An identification Isomorphic between the Lie's algebras $\mathbb{R}^3$ and $\mathfrak{so}(3)$ is given in the following way

$$w := \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} \rightarrow \hat{w} := \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix} \tag{1}$$

Also, for each pair $\omega = [\omega_1, \omega_2, \omega_3]^T$ and $\nu = [\nu_1, \nu_2, \nu_3]^T$ then $\omega \times \nu = \hat{\omega}\nu$. In all cases, $\omega \in \mathfrak{so}(3)$ will give the direction of a rotation fixing a measure of an angle, choosing $|\omega| = 1$ by the magnitude of angular spatial velocity, *Rodrigues' formula* says that the relation

$$e^{\hat{\omega}\theta} = I + \sin\theta\hat{\omega} + (1 - \cos\theta)\hat{\omega}^2 \tag{2}$$

gives rotations around the axis described by $\hat{\omega}$ with a measure θ, indeed; giving an arbitrary point $p = p(0)$ in a rigid body O and a parametrization for determinate rotation of t radians, is generated the following differential equation

$$\dot{p}(t) = \hat{\omega} p(t) \tag{3}$$

where, the solution is given by $p(t) = e^{\hat{\omega}t} p(0)$ (here, $\omega \times p(t) = \hat{\omega} p(t)$). Because exponential map $\exp : \mathfrak{so}(3) \rightarrow \mathrm{SO}(3)$ is surjective [2,9] all rotation matrix is the exponential of a skew-symmetric matrix.

Onwards, we denote the Lie algebra of SE(3) by $\mathfrak{se}(3)$. Twists give a way to parametrize motions of the joints in the Multi–section Robot of this paper (our Robot are a set of links and joints that form a kinematical chain: an assembly of rigid bodies connected by joints to provide motion that is the mathematical model for the mechanical system). Robot modeled here looks like a hypnotized cobra that eludes obstacles by motions of its links product of setting adequate configurations of its joints.

2.2 Screw Theory

One of the multiple ways to introduce twists is representing elements by matrices 6×1 in the form $\left[v^T, \omega^T\right]^T$, where v denotes linear velocity and ω spatial angular velocity. Another way consists in the following, let $\omega \in \mathbb{R}^3$ the vector that describes the direction of the axis of rotation with $|\omega| = 1$, and $q \in \mathbb{R}^3$ a point on the axis [5]. Assuming that the link rotates with unit velocity, then the velocity of a point $p(t)$ in a rigid body O, is

$$\dot{p}(t) = \omega \times (p(t) - q). \tag{4}$$

This equation allows representing each element $\xi \in \mathfrak{so}(3)$ in homogeneous coordinates by defining the 4×4 matrix

$$\hat{\xi} := \begin{bmatrix} \hat{w} & v \\ 0 & 0 \end{bmatrix} \tag{5}$$

where $v = -\omega \times q$. Since that the exponential map $\exp : \mathfrak{se}(3) \to \mathrm{SE}(3)$ can work in matricial sense by the information given above, $e^{\hat{\xi}\theta} \in \mathrm{SE}(3)$ produces rotations followed by translations in both cases of magnitude θ radians (translations parallel to the rotation axis scaled by ω), *Chasles' theorem* says that, *all rigid motion can split by a rotation following by a translation.* There are available computational advantages by using in this way twist's techniques, as optimize the computational cost in contrast to *Denavid–Hatenberg's Algorithm.*

A geometric approach to twists is given by *Screws.* A *Screw* is a triple (l, h, θ) where l denotes an axis, h the pitch, and θ the magnitude of angle around of l. A screw motion represents a rigid bdy motion by a magnitude of rotation about the axis l followed by translation by an amount $h\theta$ parallel to the axis l. If $h = \infty$ then the corresponding screw motion consists of a pure translation along the axis of the screw by a distance θ. In other words, in the case that the pitch $h < \infty$ the motion of rotation and translation (could be a pure rotation around the axis when $h = 0$) is named *finite.* When $h = \infty$ the motion is named *infinite* or *prismatic.* Other important facts about Screws are: (1) Rigid motions associated with screws are in surjective correspondence with motions generated by twists, and (2) exponential map $\exp : \mathfrak{so}(3) \to \mathrm{SO}(3)$ is a surjective map.

To can change coordinates in a twist, there exists a map called *Adjoint Representation* denoted by Ad. In general, suppose G is a Lie group, $C_g : G \to G$ *the inner automorphism of* $g \in G$, $\mathfrak{g}$ the Lie algebra of G and $\mathrm{GL}(\mathfrak{g})$ *the general linear group of* $\mathfrak{g}$ (i.e. group of linear automorphism of $\mathfrak{g}$). Supposse that $g \in G$, adjoint representation is defined as

$$\begin{aligned} \mathrm{Ad} : G &\longrightarrow \mathrm{GL}(\mathfrak{g}) \\ g &\longmapsto dC_g. \end{aligned}$$

In this particular case, making identifications, for each $g \in \mathrm{SE}(3)$ with a twist ξ, occurs that $\mathrm{Ad_g} = g\xi g^{-1}$ is a *twist change of coordinates of* ξ.

2.3 Product of Exponential Formula

The manipulator described here is a sequence of links and joints such that, they go out of a device setting its joints for eluding obstacles. Until passing an obstacle, the manipulator takes a determinate sequence of configurations to avoid. Election of pure rotational motions induces a restriction to a subgroup of associated screws in SE(3), in this case, joints are named *Spherical* or of *Socket Ball Joint.* The forward kinematics here is based in achieve different configurations for each joint to can describe the configuration of the last link of the sequence of joints, in this last join, is installed a camera that visualizes

the locally available motion space to can elude obstacles, and the initial part emerges from a device that puts out a determinate number of links and above is located a camera that maps globally around the manipulator. More precisely, let I the inertial reference frame, and given a sequence of $n+1$ links, then are n joints, hence; for the i joint $(1 \leq i \leq n)$, i should denote the joint that lie between the $i-1$–nth and i–nth link. Then, the relationship that involves the configuration with the initial joint and the configuration of the nth joint is [5]

$$g_{I,n}(\theta) = g_{I,1}(\theta_1)g_{1,2}(\theta_2)...g_{n,n+1}(\theta_n) \tag{6}$$

where $\theta_1, \theta_2, ..., \theta_n$ are the sequence of the amounts of the rotations. We can construct recursively that in this case

$$g_{I,n+1}(\theta) = e^{\hat{\xi}_1\theta_1}e^{\hat{\xi}_{1,2}\theta_2}...e^{\hat{\xi}_{n,n+1}\theta_n}g_{I,n+1}(0) \tag{7}$$

Choosing an adequate reference frame, is possible pick $g_{n,n+1}(0) = Id$. Using adjoint representation is possible to prove that $\hat{\xi}_i$ is the twist that corresponds to the i-nth joint $(1 \leq i \leq n)$.

An important step to modeling the manipulator of this paper is, given a determined net configuration, gives each rotation involved in the motion, for this objective, we apply *Paden-Kahan subproblems* to design the inverse kinematic by focus in Rotation about two subsequent axes [6–8].

3 Path Planning

It is a path where certain time distributions are specified, for example in terms of speeds and accelerations at each of the different points. Thus, a trajectory is related to the time in which each part of the path is completed, so depending on the speed with which it is carried out, the trajectory will change. The different trajectories are selected considering the physical restrictions of the drives, as well as certain criteria path quality such as precision and smoothness.

This path planning is made using the point to point method, also using third degree polynomial for control the position and velocities. To carry out the planning process it is necessary that the inverse kinematics of the robot have been designed. I also know must implement a path generation method as point-to-point movement, sequence movement points, among others.

For this movement a third degree polynomial equation is used, this together with a set of parameters is performed to obtain the position and acceleration of the links. These parameters were chosen in a way that the user can modify both the position and the speed that he expects the actuator to have. This path planning is made using the point to point method, also using third degree polynomial for control the position and velocities. To carry out the planning process it is necessary that the inverse kinematics of the robot have been designed. I also know must implement a path generation method as point-to-point movement, sequence movement points, among others.

For this movement a third degree polynomial equation is used, this together with a set of parameters is performed to obtain the position and acceleration of the links. These parameters were chosen in a way that the user can modify both the position and the speed that he expects the actuator to have.

3.1 Path Point to Point

This is a movement method where the manipulator must move from an initial position (θ_0) to a final position (θ_n) in a time (t_f). For this case the path of the effector is not taken into account. For this movement a third degree polynomial equation is used, this together with a set of parameters is performed to obtain the position and acceleration of the links. These parameters were chosen in which you can modify both the position and the speed that you expect the actuator to have.

Parameters for positions:

$$\begin{aligned} \theta(0) &= \theta_0 \, |rad| \\ \theta(T_f) &= \theta_f \, |rad| \end{aligned} \tag{8}$$

Parameters for speeds:

$$\begin{aligned} \dot{\theta(0)} &= \dot{\theta_0} \, |rad/s| \\ \dot{\theta(T_f)} &= \dot{\theta_f} \, |rad/s| \end{aligned} \tag{9}$$

Polynomial for position:

$$\theta(0) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{10}$$

Deriving Eq. 10 gives the speed polynomial:

$$\dot{\theta(0)} = a_1 + 2a_2 t + 3a_3 t^2 \tag{11}$$

Deriving Eq. 11 the acceleration polynomial:

$$\dot{\theta(0)} = 2a_2 + 6a_3 t \tag{12}$$

For the initial time $t_0 = 0$, substituting in 10, 11 obtain $a_0 = \theta_0$ and $a_1 = \dot{\theta_0}$ Replacing $t = T_f$ in to find the parameters a_2 and a_2 arises the following equations.

$$a_2 = \frac{\dot{\theta_f} - \dot{\theta_o}}{2} - \frac{3}{2} a_3 t \tag{13}$$

$$a_3 = \frac{2}{t^3}(\theta_f - \theta_0) + \frac{2}{t^2}\dot{\theta_0} + \frac{1}{t}(\dot{\theta_f} - \dot{\theta_0}) \tag{14}$$

Equations can be summarized

$$\begin{cases} a_0 = \theta_0 \\ a_1 = \dot{\theta_0} \\ a_2 = \left(\frac{\dot{\theta_f} - \dot{\theta_0}}{2}\right) t - \frac{3}{t^2}\left(\theta_f - \theta_0\right) + \frac{2}{t}\dot{\theta_0} + \left(\dot{\theta_f} - \dot{\theta_0}\right) \\ a_3 = \frac{2}{t^3}\left(\theta_f - \theta_0\right) + \frac{2}{t^2}\dot{\theta_0} + \frac{1}{t}\left(\dot{\theta_f} - \dot{\theta_0}\right) \end{cases} \tag{15}$$

In the Fig. 1 the q_i that represent the angles of the rotational joints and the P_x, P_y, P_z are the points respectively in the Cartesian Plane

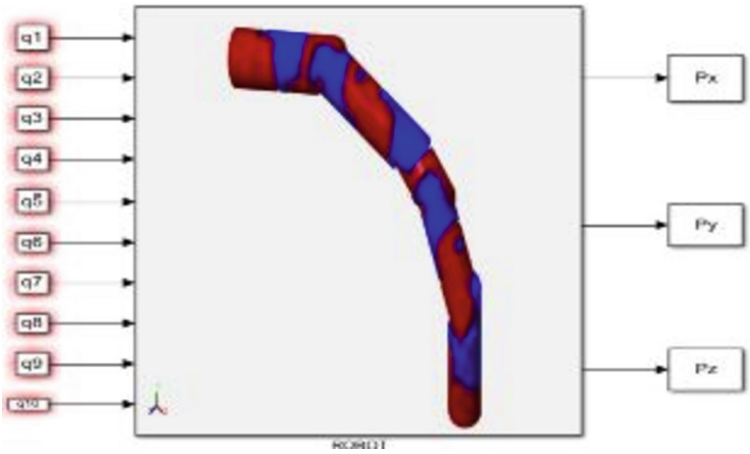


Fig. 1. Diagram of a servant robot with 10 degrees of freedom

3.2 Implemented Points

A servant-type multi section robot with 10 degrees of freedom was generated. As a first measure, a simplification of the robot's block diagram was carried out where the direct kinematics equations were implemented.

- Goal points
  - $P_{01} = (0, 0.6750, 0)$
  - $P_{02} = (0, 0.1219, -0.0646)$
  - $P_{03} = (0.43, 0.2464, 0.3)$
  - $P_{04} = (-0.3, 0.53, 0.1)$

In Fig. 2 you can see the goal points in three-dimensional space.

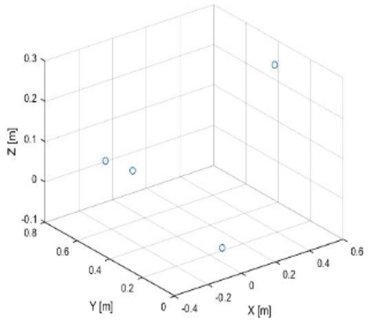


Fig. 2. Goal points

With the target points defined, the following is to calculate the inverse kinematics to know which points the obtained angles will reach, and a comparison was made of the angles found using the Paden-Kahan sub problem algorithm and the real angles of the robot. See Table 1.

Table 1. Position error for each target point

Goal point	Point kinematic direct	Error
0, 0.6750, 0	−0.0038, 0.6421, 0.111	0.11%
0, 0.1219, −0.0649	0.0195, 0.4422, −0.2484	36.98%
0.4300, 0.2464, 0.3	0.4190, 0.0083, −0.1589	27.70%
−0.3, 0.53, 0.010	−0.2752, 0.5743, −0.0320	8.49%

In Fig. 3 you can see the difference between the desired target point (blue circle) with respect to the point obtained with the inverse kinematics with Paden-Kahan (red circle).

It is important to take into account the angles with which they were achieved, for this it presents Table 2, where you can see all the angles for the desired trajectories, in this table you can see the number of articulation and the objective angles of this for each point, with which it will go from angle P_0 to angle P_1 a if until reaching P_4,

As previously described for point-to-point movement, it is necessary that the trajectory is defined in a period of time . Table 3 shows the selected time for each way point.

However, this is not the only parameter that must be taken into account. In addition, all the initial and final speeds were assigned a zero range, with which the robot will have an infinitesimal time in which it will not move.

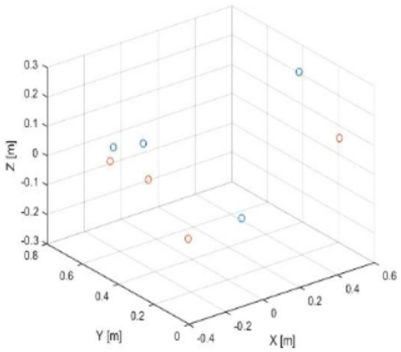


Fig. 3. Goal point and obtained point (Color figure online)

Table 2. Joint movement for all trajectories

Joint	P_1	P_2	P_3	P_4
1	−0.0740	0.4118	0.4569	0.1549
2	0.22633	−0.0740	−0.8782	0.2470
3	0.0445	−0.3343	0.1862	−0.251
4	0.0538	0.2459	−0.8732	0.0538
5	−0.2660	−0.6566	0.7926	0.1135
6	−0.0740	0.4118	0.4569	0.1549
7	−0.4281	−08652	−0.1151	0.1096
8	−0.0824	−0.2602	0.0724	0.3057
9	0.0751	0.0913	−0.9756	0.0167
10	−0.1827	0.1072 8	−0.1373	−0.2001

Table 3. Times for each trajectory

	P_1	P_2	P_3	P_4
T(S)	0	3	4	10

4 Architecture System and Algorithms Implemented and Simulation

The environment is developed in GAZEBO and integrated with ROS *(Robot Operating System)* and MATLAB. For the generation in the world, the robot has been designed as a chain of links taking into account a dynamic (i,e inertia, friction, and mass), each actuator has a servomotor. This means that the position and speed of each of them can be used by a PID, the final effector has a camera that allows determining the correct distance. The virtual environment receives the angles of each link generated by a twist.

To avoid obstacles in a static environment was used the *algorithm A** or *Astar type search of graphs*. The algorithm uses an evaluation function, where represents the heuristic value within a point in the work-space (*C-space*) evaluate from the n step and the end, the real cost of the path traveled to reach the goal n from the initial point. Algorithm A^* uses an evaluation function, where $f(n) = g(n) + h'(n)$ Thus $h'(n)$ represents the heuristic value of the node to be evaluated from the current one n $g(n)$ the actual cost of the path to reach that node, until the end. The Fig. 2 and Fig. 3 we show the algorithm implemented in Matlab for different pathways with restrictions given an initial and final point. Thus Fig. 2 you can see a trajectory with little deformation since the points are very close, but unlike figure ?? the trajectory needs deformation due to the restrictions and distance of the points.

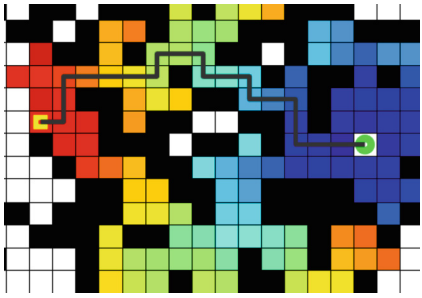


Fig. 4. Path planning using A*

For softening the trajectory the Bezier curves algorithm is used in a polynomial way of order 3. The idea of geometrically defining the shapes is not too complex: a point on the plane can be defined by coordinates. For example, an initial point P_o has some coordinates (x_1, y_1) and a final point P_f corresponds to it (x_2, y_2). To draw a curve between both, it is enough to know its position and the essential elements of a Bezier curve; the points are called "reference points" or "nodes". P_r. The shape of the curve is defined by invisible points in the plane, called "control points", "handlers" or "hands".

The generalization would become the curves called "Spline", that is, the Bezier curve is a third-degree Spline. The Bezier curve of degree n can be generalized as follows. Given the points P_0, $P_1, ..., P_n$, the Bezier curve is of the type:

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} P_i (1-t)^{n-i} t^i, t \in [0, 1]$$

In Fig. 4 we show how the curve softens after applying the algorithm A, according to the deformations found, the nodal points are produced which control the curvature in the robot with the previous expressions, we easily obtain the positions in which we must place the 3 additional vertices to achieve specific initial conditions, depending on the position of the point and the first and second discrete derivatives in it, for each component (in the flat, x and y).

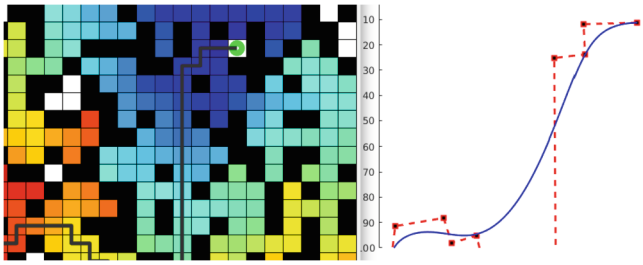
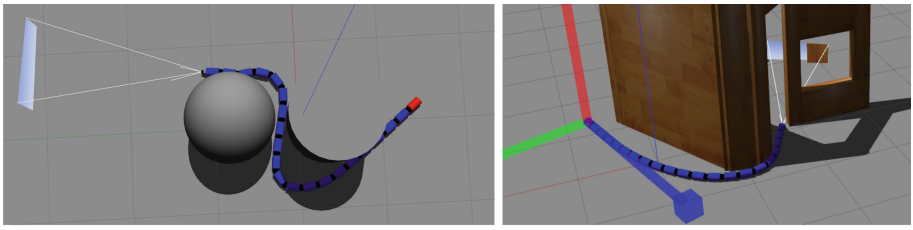


Fig. 5. Spline curve softens after applying the algorithm A*

For the simulation of the robot, cylindrical links are connected through revolution joints, the initial link of red color is followed by the blue cylinders, in the end effector is located a camera, this corrects the final angles that the Bezier algorithm delivers. After the curve is found with the Bezier algorithm, we proceed by inverse kinematics to find the angles that will be the control variables for each twist. The simulation here has twenty connected links. The simulation is tested with two events, the first one step is with two objects as obstacles and several zones of constraints forcing the robot to generate an s, the second one generates a virtual environment of a house where the robot searches for the exit for this test. Environments were controlled, giving curvature parameters and restriction zones.

The Fig. 6(a) shows the robot generating a curve to avoid different obstacles and 6(b) generating a curve to embroider an area.



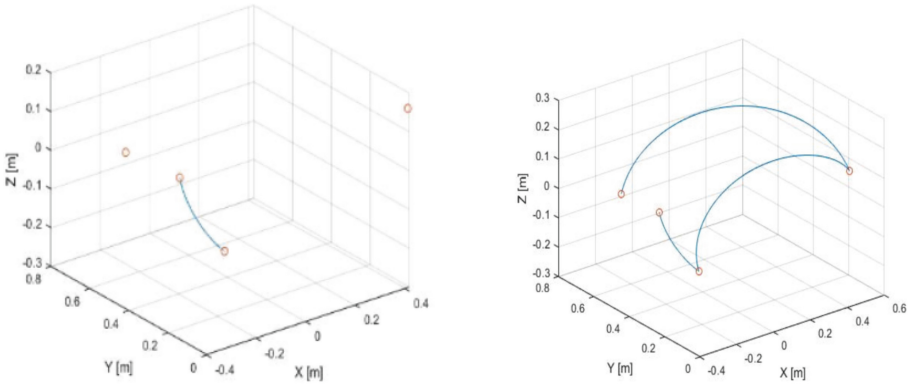
(a) 1 (b) 2

Fig. 6. curvature of the robot in the virtual world

5 Results

In the first trajectory, it was calculated for the links outside the point P_1 to P_n Table 2, this movement was carried out in time shown in Table 3 In the Fig. 7(a) you can see the movement in three-dimensional space the from movement P_1 to P_2 in 7(b) we can see all the complete routes (the sum of all the routes), since This mode made the robot move from the starting point. P_1 to the end point P_4 passing through the intermediate points P_2 and P_3. It can also be seen that the curves are made with a smooth movement, this is due to combining it with

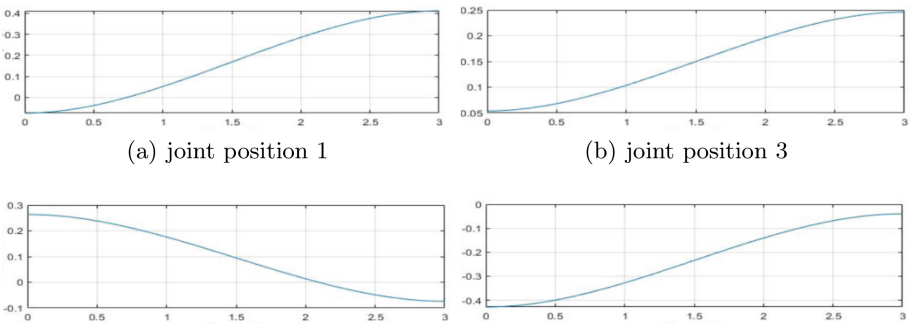
a bezier allowing the robot not to generate sudden movements, protecting the actuators.



(a) Path end effector first point (b) Complete path of the end effector

Fig. 7. Path end effector

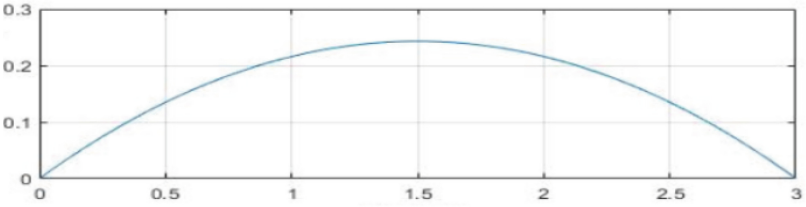
In the Fig. 8 see the position of joints with respect to time, in each of these figures it can be seen that the primary requirement of having a smooth movement in each joint was met, in addition to this, the desired parameter of the initial and final velocity was met, since it starts at 0 rad/s and ends in 0 rad/s. In the Fig. 9 see the velocity of joints with respect to time, in each of these figures, it is important to observe the behavior of the velocity, this tells us that it stops when it reaches the point end. It can also be observed that a straight path was not made between point and point, since there were no intermediate points between these points. Now, if you want to improve this implementation, you should generate a trajectory with more intermediate points. From Fig. 10 the total movement of each of the links can be observed, in these figures it can be observed how, by modifying the time in which the task is required to be carried out, the speed is modified, making the robot have more abrupt movements.



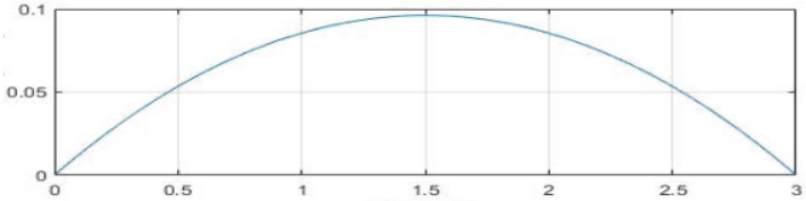
(a) joint position 1 (b) joint position 3

(c) joint position 5 (d) joint position 9

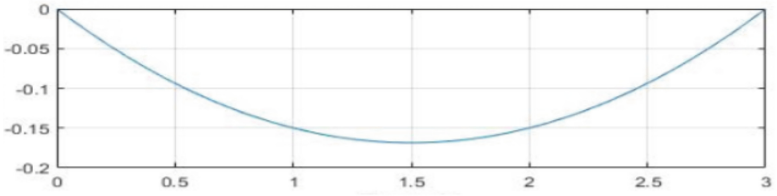
Fig. 8. Joint positions



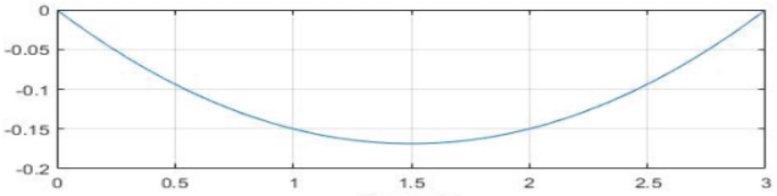
(a) joint speed 1



(b) joint speed 3

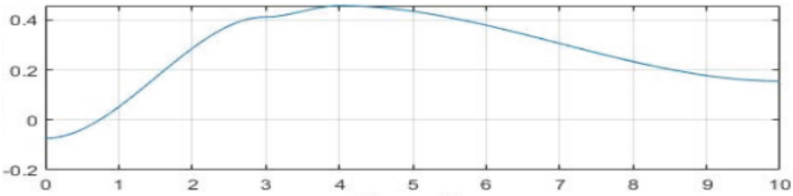


(c) joint speed 5

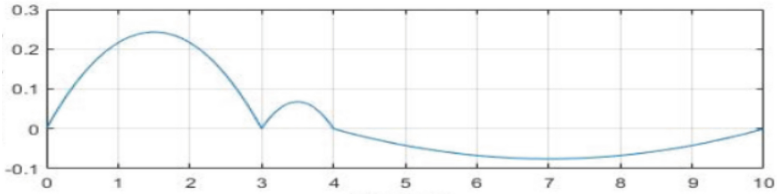


(d) joint speed 9

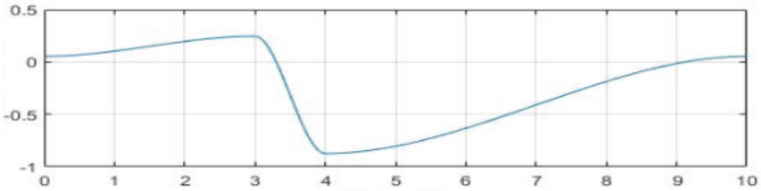
Fig. 9. Joint speeds



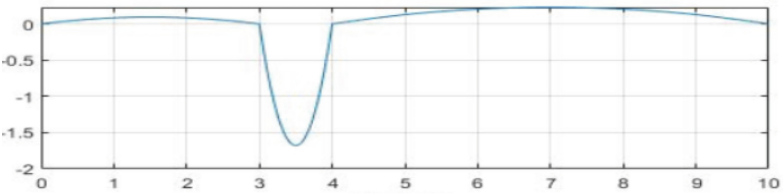
(a) Total joint position in trajectory 1



(b) Total joint speed in trajectory 1



(c) joint position in trajectory 2



(d) joint speed in trajectory 2

Fig. 10. Joint speeds

6 Conclusion

This article showed how the screw theory is an alternative method to the traditional methods to find the kinematics of a robbery, as is the case of the Denavit-Hartenberg parameters applied to a robot with many links, although in the clarification of the equations inverse kinematics there was an error in the angles found with respect to the desired ones. We integrate several algorithms including route planning that controls the rotation smoothness of each link through a third order polynomial, you can see how the robot reaches the inverse kinematics point in the desired time, for each link it is necessary to find a Polynomial control global path planning was also implemented to avoid obstacles in a static environment and together with Bezier's suvizdo algorithm it is possible to generate a curve that the robot can copy. A problem with this methodology is that you have to know the work environment and the obstacles must be static, without the ability to navigate in an unknown environment for future work, it is proposed to work with more recent methods in the field of artificial intelligence as a learner. Reinforced to teach the robot how to move according to different obstacles and see the behaviors in a dynamic environment.

References

1. Denavit, J., Hartenberg, R.S.: A kinematic notation for lower-pair mechanisms based on matrices (1955)
2. Duistermaat, J.J., Kolk, J.A.: Lie groups. Springer Science & Business Media (2012). https://doi.org/10.1007/978-1-4614-8024-2
3. Featherstone, R.: Rigid Body Dynamics Algorithms. Springer (2014). https://doi.org/10.1007/978-1-4899-7560-7
4. Kim, Y., Cheng, S.S., Diakite, M., Gullapalli, R.P., Simard, J.M., Desai, J.P.: Toward the development of a flexible mesoscale MRI-compatible neurosurgical continuum robot. IEEE Trans. Robot. **33**(6), 1386–1397 (2017)
5. Niu, G., Zhang, Y., Li, W.: Path planning of continuum robot based on path fitting. J. Control Sci. Eng. 2020 (2020)
6. Oliver-Butler, K., Till, J., Rucker, C.: Continuum robot stiffness under external loads and prescribed tendon displacements. IEEE Trans. Rob. **35**(2), 403–419 (2019)
7. Ouyang, B., Liu, Y., Tam, H.Y., Sun, D.: Design of an interactive control system for a multisection continuum robot. IEEE/ASME Trans. Mechatron. **23**(5), 2379–2389 (2018)
8. Paden, B., Sastry, S.: Optimal kinematic design of 6R manipulators. Int. J. Robot. Res. **7**(2), 43–61 (1988)
9. Vergara, N.J.W., et al.: Un ejemplo de aplicación momento en dimensión infinita. Master's thesis, Uniandes