2010 Latin American Robotics Symposium and Intelligent Robotics Meeting

# Smooth trajectory tracking interpolation on a robot simulator

Rafael Renan Pacheco
LAboratory for Research on Visual Applications
Santa Catarina State University
Joinville, Brazil
E-mail: rafael.renan.pacheco@gmail.com

Marcelo S. Hounsell, Roberto S. U. Rosso, Jr.
LAboratory for Research on Visual Applications
Computer Science Department
Santa Catarina State University, Brazil
E-mail: {marcelo, rosso}@joinville.udesc.br

André B. Leal
Electrical Engineer Department
Santa Catarina State University
Joinville, Brazil
E-mail: leal@joinville.udesc.br

*Abstract*—**When there is the need of a robotic manipulator to follow a certain smooth and continuous trajectory, it involves the specification of the curve to be followed and the way the gripper should behave alongside it. This work aims to identify and analyze a technique to be used in a 3D virtual robotic simulator that executes smooth and continuous movements. It also aims to develop a method for trajectory programming using a solution that covers the main problems of trajectory tracking (smoothness of path and gripper approach). An efficient and flexible method to embed trajectory tracking was indentified and implemented which allows its integration with existing standard (linear or circular) movements.**

***Keywords: Robot manipulator; simulator; trajectory tracking; smooth curves.***

## I. INTRODUCTION

One of the problems regarding robotic manipulator controllers is when there is the need to follow a trajectory in a continuous and smooth way [1]. Most commercial robotic controllers use linear or circular interpolations which allow them to calculate the trajectory length and to define a trajectory speed profile [2]. However, for most of the tasks which needs trajectory tracking, like welding, the manipulator speed is limited by the task process reducing the use of the robot dynamic variables [2].

A task such as the standard pick-and-place when a manipulator capture, move and place a specific object in a desired position can become a lot more complex if there is the need for continuity and smoothness of movements [4].

According to [3], the use of curve interpolation methods has become default in most CNC machines programming and, companies such as FANUC and SIEMENS have developed interpolation methods for NURBS curves for their CNC systems. CNCs are robot-like machines that already use curve interpolation methods to perform more complex tasks with better performance at lower costs.

Smooth curves increases robot tasks domain for operations in curved surfaces, for instance. The trajectory definition to realize this kind of task without smooth curves would be of higher complexity since it would be necessary to define all points where the end-effector should operate on.

The trajectory definition is usually made at the Cartesian space, specifying the path which the end-effector shall pass and the orientation the gripper shall present. However, defining Cartesian points only are not enough to realize such manipulator constraints. The manipulator controller needs to find all angle values of the robot joints given the Cartesian points and the gripper orientation. To solve this problem an inverse kinematic calculation is used [5][14]. For spline movements, the trajectory is generated by a small set of control points through a spline interpolation rule which allows more freedom and flexibility for robot tasks [6].

Robot programming can be accomplished in two ways: on-line and off-line. In off-line programming, the tasks are programmed apart from the robot, usually in a computer, to later be sent to the manipulator controller. Whilst in on-line programming, the task is realized with the robot, through joysticks or even grabbing and moving the manipulator arm [5]. Therewith, precision and smoothness problems in the on-line programming can be found since the entire trajectory is manually defined. On the other hand, off-line programming has advantages such as trajectory simulation, smooth trajectory generation and collision simulation [7].

The use of robot simulators have advantages: represent lower risks to an operator; avoids equipment wearing as well as long preparations to use the real robot; can be integrated with CAD or CAM to detect collisions risk with obstacles or even with the manipulated part [7].

To define and control a trajectory in a smooth and efficient way an analysis is required regarding the performance, the behavior of the manipulator movement and, a method to define the trajectory and movement the manipulator.

This paper presents a way to incorporate an efficient spline-based curve that allows a smooth interpolation for a gripper path and orientation on a 5 ODF robot simulator. The paper is divided as follows: next section presents some related work; Section III presents the simulator chosen for incorporating the smooth interpolation; Section IV presents the performance and functional tests and analysis, and; Section V concludes this text.

## II. Related Works

### A. Trajectory tracking with inverse kinematic

The work of [8] proposes an analytic solution to solve the inverse kinematic of a 5-DOF robot manipulator, specifying a desired approach for the end-effector. It was pointed out that controlling a robot with less than 6-DOF through a trajectory, with a fixed approach, can be very difficult. To solve this problem, an analytic solution was developed [8], based on the PArm (Pioneer Robotic Arm) robot, which works for any position and approach reachable by the end-effector besides providing a solution for 99% of the cases which the end-effector orientation is not reachable.

To analyze and validate it, a trajectory tracking in a PArm robot simulator was realized. The end-effector orientation was defined by an orientation matrix, and the trajectory defined by an analytic curve. The tracking process was processed as follows: First, the inverse kinematic is calculated to find the link angles, based on the point of the curve and the defined approach orientation. After that, the forward kinematic was calculated using the link angles returned by the inverse kinematic, being able to find the end-effector position. Also, those angles were applied in a real PArm robot for further results comparison.

The results showed a continuous movement of the manipulator through the trajectory, in a smooth way and tests realized in the real PArm robot had shown close similarity to the simulator environment. Although the simulator precision to generate the trajectory was substantially higher than the real robot and specifying a curve with an analytic formulae and orientation matrix is not a trivial task, the manipulator was able to move with continuity and smoothness.

### B. Movement under parametric curves.

A solution to create movement under parametric curves using B-Splines was presented by [9], which is easier than using analytic ones. The work shows a few techniques to represent the movement of rigid bodies over a defined trajectory, being divided in four types:

- Representation of the rigid body through Cartesian product of position and orientation.
- Use of kinematic mapping, where the positions are mapped into a higher degree space, using quaternion and B-Spline to define the movement.
- Methods that consider the location as elements of the Lie group [10 apud 9].
- Methods which uses circular arches to create orientation or curves.

The proposal of [9] has two characteristics: The first is that the orientation is locally parameterized with geometric vectors. The second is that the Frenet movement system has a geometric interpretation and kinematic properties. Through these two characteristics, it is ensured a kinematic movement [9].

Through this method, it is possible to create smooth interpolation of the end-effector orientation over a trajectory. However, [9] work took in account an interpolation in all reference axis of the rigid object, limiting its usage on robots that are possible to reach different approaches for the end-effector relatively to the three robot base axis. This way, the solution is not applicable to robot manipulators which the end-effector orientation is limited by the robot plane, formed by the robot shoulder and the end-effector, thus limiting the orientation in two axes.

### C. Robot Simulator in X3D

A robot simulator with 5-DOF (Degrees of Freedom) has been developed [7]. The simulator is based on an ER-4PC 5-DOF robot, in which three are for anthropomorphic arm movement and two for the end-effector. It was developed using Java [11] and X3D [12], integrated by a SAI (Scene Access Interface) API (Application Programming Interface) [13 apud 7].

The forward and inverse kinematics of this simulator has been developed [14], where an analytic method was used to calculate the inverse kinematic, which has the advantage to be a fast method and able to identify the solutions for both possible elbow position: above and under. To solve this problem, [14] developed the inverse kinematic algorithm which needs a parameter to define what position to use in the calculation, allowing a continuous solution.

Through this simulator we will implement the smooth trajectory functionality, adding a new command line to the simulator programming language and changes in the existing interfaces. Also new features regarding smooth trajectory, like visualization of the control points and trajectory generation on the user interface for preview, were added.

## III. Virbot4u - A smooth robot simulator

### A. The curve

To generate a smooth trajectory in a virtual environment, we will use the Catmull-Rom spline curve [15], which is a family of the cubic splines interpolator curves whose tangent of each control point is calculated based on the control point before and after, represented by: T(pi+1 – pi-1), where “T” stands for tangent tension and, “p” for a control point.

The Catmull-Rom spline has the fallowing characteristics [15]: C1 continuity; local control; the final curve does not lie in the convex hull but interpolate all control points. The use of a cubic curve allows computing a smooth trajectory in an efficient way. Also, the tension (T) parameter allows achieving different results of smoothness for a same group of control points, increasing the domain of generated trajectories.

Note that, by definition, the tangent of the first control point does not exist, since there is no control point before that. The same applies to the last control point, as no control point exists after that, featuring a curve which does not interpolate the extreme control points. To avoid this problem, the extreme points are repeated to allow a complete interpolation of the control points.

The parameter T used in the matrix formulation of the Catmull-Rom spline (1) affects the tension of the curve near the control point. Usually a value of 0.5 for T is applied, which results in a smooth curve, but can be any value [15]. The parameter “u” represents the parametric value of the point, ranging from 0.0 to 1.0, and "p" is the current control point.

$$p(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -T & 0 & T & 0 \\ 2T & T-3 & 3-2T & -T \\ -T & 2-T & T-2 & T \end{bmatrix} \begin{bmatrix} p_{i-2} \\ p_{i-1} \\ p_i \\ p_{i+1} \end{bmatrix} \quad (1)$$

## B. *Interfaces*

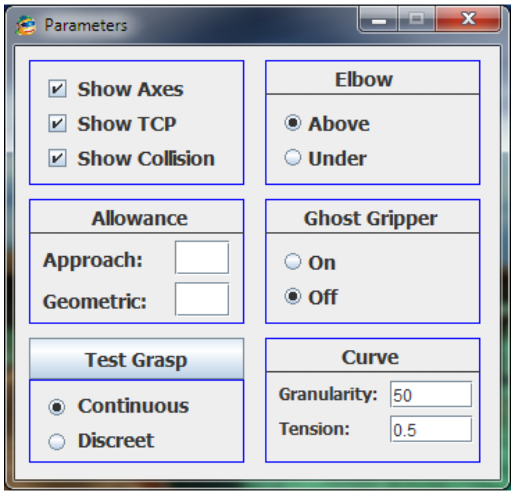
During the implementation, the VirBot4u user interfaces have been changed. Fig. 1. shows the new parameter interface that meets the new trajectory functionalities. The first change was in the parameter window of the simulator, where options to choose the elbow position, to allow a continuous trajectory tracking was added. Also an option to display a ghost version of the gripper of each saved control point was added, which allows seeing the approach of the end-effector during the trajectory definition. The last options added in the parameter window were the granularity, which defines how many points will be generated between each pair of control points, and the tension, which allows change the smoothness of the Catmull-Rom spline curve. Extra options can be found to control the exposure of reference axes, Tool Central Point (TCP), collision detection and parameters for grasping analysis. All these last options are out of the scope of the present paper. The elbow position will only be used during the manipulator movement, not conflicting with the trajectory definition. Also, granularity value will be used for trajectory generation preview as well as for manipulator movement.

The simulator window that saves control points and postures was also modified to allow choosing which points become control points. Each point represents a posture of the robot, and contains information like all axis angles of the robot and the end-effector Cartesian position and orientation calculated by forward kinematic.

Fig. 2 shows the Points window, divided in two contexts: the left part define posture positions which can be sent to the right part, responsible for storing the smooth curve's control points. Also, the user can easily manipulate the control points by reordering, adding, removing and testing points for singularities through inverse kinematic validation.

For the control points, there are three buttons on the top right of the screen, which allows reordering the control points, followed by a GS button, which generates a command line in the Programming window using the selected control points. Note that, in order to generate a trajectory and to move the manipulator, at least, four control points are necessary since a cubic curve is used in this process.

The Points window allows visualizing the control points through the View Points button as well as validating them to detect unreachable positions by the inverse kinematic calculation. The visualization renders a small sphere in the position of each end-effector from the Control Points list, shown in Fig. 3, and has its color changed by the validation process, turning green to represent a reachable position and, red to represent an unreachable position according to the inverse kinematic. Therefore, the user can foresee an invalid control point during the trajectory definition.



[✓] Show Axes
[✓] Show TCP
[✓] Show Collision

Elbow: [✓ Above] Under

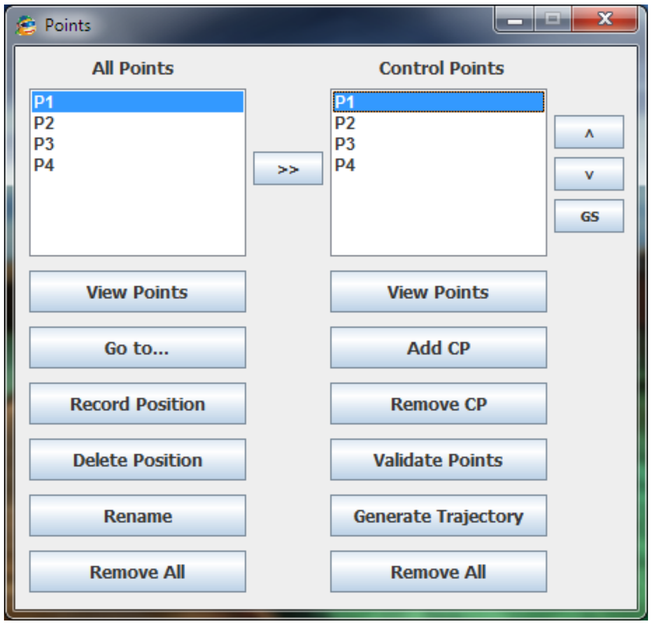
Allowance: Approach: Geometric:

Ghost Gripper: On [✓ Off]

[✓ Continuous] Discreet

Curve: Granularity: 50 Tension: 0.5

Figure 1. Parameter window of the robot simulator.



All Points: P1, P2, P3, P4

Control Points: P1, P2, P3, P4

Figure 2. Points and Control Points window.

The Generate Trajectory button (Fig. 2) generates a smooth trajectory outlook on the simulator environment, using the pre-defined control points and the granularity set in the parameter interface. This functionality allows a fast preview of the smooth trajectory, having the trajectory updated after each modification in the control points list like changing the position of a control points.

Fig. 3. Shows the simulator environment, with the ER-4PC robot, five control points, TCP and ghost gripper parameters both set to ON, trajectory generation activated and granularity set to 50 points.

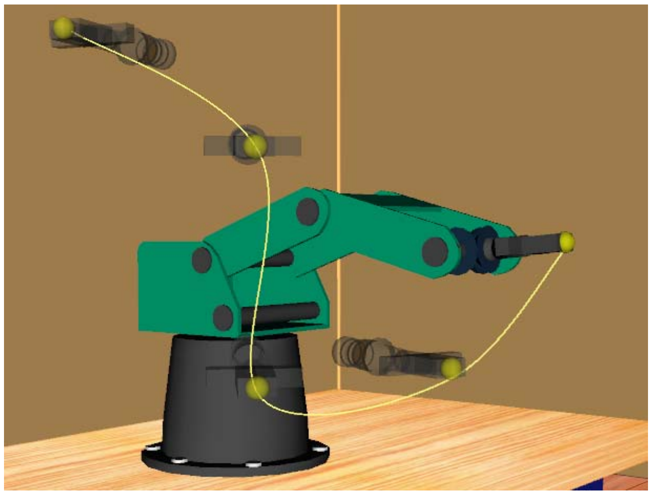


Figure 3. Robot simulator environment showing a smooth trajectory.

## C. *Programming*

To move the manipulator through the smooth trajectory, a new command line to the programming simulator window was developed to look like existing SCORBASE [16] commands. The command line allows defining the end-effector approach and the list of control points to be interpolated, for example:

GS FREE P1 P2 P3 P4 P5;

- Where GS stands for Go Smooth.
- The FREE keyword identifies the end-effector approach from the following: FRONT, UP, BACK, FIXED and FREE. FRONT, UP and BACK are pre-defined approaches, overriding the approach stored when the control point was gathered. FIXED approach uses the current end-effector approach in the simulator, also overriding the control point approach. FREE results in a smooth end-effector orientation interpolation, calculated altogether with the curve interpolation, using the orientation originally gathered at each control point. For that end, a Catmull-Rom spline interpolation is also used.
- Following the approach definition is the control points list.

All control points used in this command should have been already defined at the Points window. If some invalid control point or less than four control points are selected, the simulator will throw an error message, identifying the error so the user can fix the command.

## IV. TESTS AND RESULTS

Performance and functional tests were carried out to evaluate the resulting implementation.

Two strategies to calculate the inverse kinematics and movements of the manipulator were tested.

The first strategy consists in separating the trajectory tracking process in two parts:

- First the simulator calculates the entire trajectory, saving the smooth points in memory, to later calculate the inverse kinematic of each point and move the manipulator. This strategy is labeled as T1 for the first step and T2 for the second step;
- The second strategy integrates and unites T1 and T2: For each calculated smooth point, inverse kinematic is applied to move the manipulator. This strategy is labeled as T3.

The performance tests were made in a computer running Windows 7, with an Intel Core 2 Duo 2.3 GHz processor and 2 GB of RAM. In all tests memory swap was not necessary which would cause performance loss in the validation. The tests were taken using 200, 400, 600, 800 and 1000 control points and similar values for granularity, where each set of control points were tested with all values of granularity, resulting in 25 performance results for each strategy. Each result is an average of 10 cases of the same test.

The analysis of the T1 results (Fig. 4) shows a linear behavior for most of the tests, confirming that increasing both granularity and control points does not cause an exponential time increase for the curve interpolation. However, note that for the worst case, which uses 1000 control points and 1000 points of granularity, there is a break in the linearity of the performance because T1 strategy saves a vector in memory with all calculated points, causing a loss of performance to reallocate memory for a huge amount of data in this vector. Therewith, the time of the worst case was 7773 ms (almost 8 seconds) to calculate the interpolation of the control points.

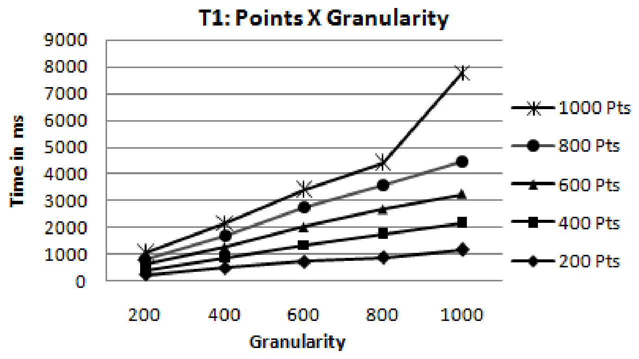


Figure 4. Performance test for the T1 strategy.

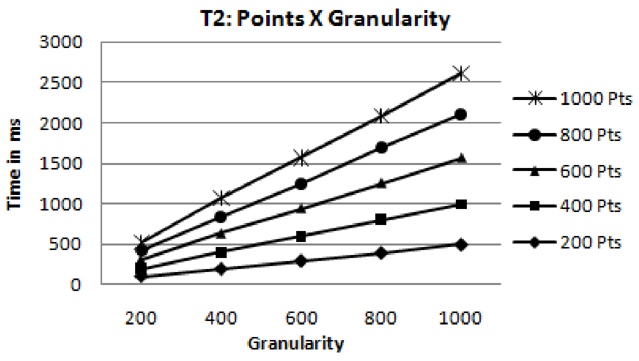


Figure 5. Performance test for the T2 strategy.

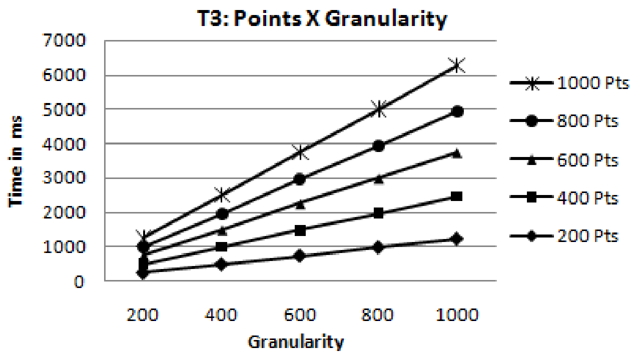


Figure 6. Performance test for the T3 strategy.

Comparing T3´s with T1´s and T2´s worst cases, there was a performance gain of 4214ms. In other words, T3 strategy was 40% more efficient. However, this comparison takes in account the linearity brake caused by T1 and higher memory allocation. Thus, it is worth making another comparison using the second worst T1 and T2 cases, which is complete linear, with 800 control points with 1000 points of granularity. In this comparison, the total time of T1 and T2 was 6555ms, whereas T3 had a performance of 4925ms. Therefore, even where T1 and T2 show a complete linear behavior, the T3 strategy was 25% more efficient than T1 and T2 which is consistent to all other cases.

Even using a united strategy, visual discontinuity and jerky movements were not detected during de movement of the manipulator. This discontinuity could be caused by the excessive calculations of curve interpolation and inverse kinematics calculation for each point during the manipulator movement.

It all means that the unified strategy, which unites the interpolation with inverse kinematics calculus, showed better performance for huge amount of data. However, normal usage of the simulator does not require such amount of points and such granularity magnitude used in the tests. Thus, both strategies did perform efficiently for reasonable amount of points, allowing us to choose the strategy with better flexibility. To this end, the strategy which calculates separately the smooth points from inverse kinematics shows more flexibility for adding validation during the calculations and for changing the curve shape as a whole. Therefore it was chosen as the default method for trajectory tracking for the Virbot4u simulator.

Functional tests were also carried out in order to verify programming flexibility and problems domain: Through the programming interface of the simulator, more complex trajectories could be created, alternating between linear and smooth trajectories, as shown in the Fig. 7.

To achieve this result, a mix between the command GP (Go Position) and the new command, GS, was used as shown in the example code:

- GS FRONT P1 P2 P3 P4;
- GP P5;
- GS FREE P5 P6 P7 P8;
- GP P9;
- GS FRONT P9 P10 P11 P12;

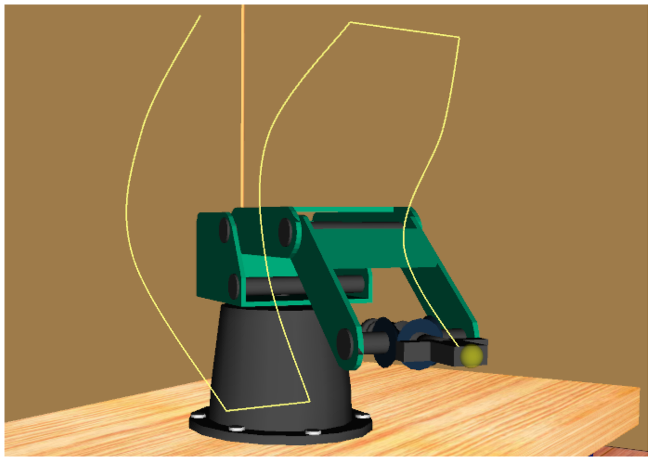


Figure 7. Mixed trajectory using linear and smooth commands.

Note that GS command is a little different from existing GP command as the former includes an approach specification that overrides the configuration gathered with the specific point. This is a flexibility which should be incorporated into the GP command.

Final tests regarding the generated trajectory were made and identified that a value of 20 for the granularity is enough to result in a smooth curve with high quality (no visible corners) in the simulator environment. However, small granularity for the trajectory tracking can result in a jerky movement, requiring higher values to achieve smooth movements. A reasonable tradeoff seems to be the value 50 for granularity which still accounts for fast response when a control point is manipulated in the Control Points list. But, if a value higher than 50 is set in the Parameter window for granularity, the simulator will use 50 as granularity for the trajectory preview to save time and get a fast visual feedback.

## V. CONCLUSION

A method to control a robot manipulator through a smooth trajectory at the Euclidean space has been presented. The Catmull-Rom spline parametric curve was selected because it interpolates control points (which sometimes are mandatory for robot tasks) and allows different results of curves to be achieved by changing a tension parameter. The parametric matrix formulation was used which allowed an easy implementation and modification of the curve definition.

An adaptation of the Virbot4u robot simulator [7] was performed which allowed to study and develop the functionalities of a smooth trajectory tracking. Through the developed user interface, the simulator allows visualizing trajectory tracking features such as points validation (if they can be reached and inverse kinematically calculated), end-effector visualization and a smooth trajectory outlook. This way, many visual information and feedback are available before start programming the trajectory, allowing identifying problems and errors.

Regarding the calculation strategy, two distinct ways of trajectory tracking implementation were analyzed and the strategy that calculates separately the interpolation points from the inverse kinematic calculation was chosen due to its performance and flexibility.

A new command line to execute the smooth trajectory tracking, GS (Go Smooth), was defined resembling commands of the existing real robot´s language. It can easily be mixed up alongside existing trajectory specifications by C0 continuity and overrides gripping approach if necessary Options like elbow position and granularity were left to be defined at the user interface, keeping the programming as simple as possible..

With the development of smooth trajectories to the Virbot4u simulator, a greater flexibility to perform complex tasks was achieved. Combined with the simulator programming interface, the usage of smooth curves increases considerably the tasks domain of the robot in a simple and efficient way. The simulator code written in Java with Xj3D [18] can be found at [17] for public and free download.

## REFERENCES

[1] N. A. Martins. “Projeto de um controlador adaptativo para robôs manipuladores no espaço de juntas” Acta Scientiarum, Maringá, 23(6):1481-1494, 2001. (In Portuguese).

[2] T. Horsh and B. Jüttler. “Cartesian spline interpolation for industrial robots”, Computer-Aided Design, 30(3):217-224, 1998.

[3] W. Yongzhang, L. Yuan, H. Zhenyu and S. Zhongxi. “Integration of a 5-axis Spline Interpolation Controller in an Open CNC System”, Chinese Journal of Aeronautics, 22:218-224, 2009.

[4] C. Müller-Karger, A. G. Mirena and S. López. “Hyperbolic Trajectories for Pick-and-Place Operations to Elude Obstacles”, IEEE Transactions on Robotics and Automation, 16(3):294-300, 2000.

[5] F. L. Lewis, D. M. Dawson, and C. T. Abdallah. “Robot Manipulator Control: Theory and Practice”, NewYork, NY: Marcel Dekker, Inc., 2004.

[6] D. Verscheure. “Contributions to Contact Modeling and Identification and Optimal Robot Motion Planning” Ph.D. thesis, Mec. Eng., Faculteit Ingenieurswetenschappen, 310 p, 2009.

[7] A. Hoss, M. S. Hounsell and A. B. Leal. “Virbot4u: Um Simulador de Robô usando X3D”. I Simpósio de Computação Aplicada, Passo Fundo, pp. 1-15, 2009. (In Portuguese).

[8] D. Xu, C. A. A. Calderon, J. Q. Gan H. Hu, and M. Tan. “An analysis of the inverse kinematics for a 5-DOF manipulator”, International Journal of Automation and Computing, 2(2):114-124, 2005.

[9] A. P. Pobegailo. “Design of motion along parameterized curves using B-splines”, Computer-Aided Design, 35:1041-1046, 2003.

[10] F. C. Park and B. Ravani. “Bezier curves on Riemannian manifolds and Lie groups with kinematics applications”, ASME Journal of Mechanical Design, 117(1):36-40, 1995.

[11] Sun Microsystems. (2010, May 30). Java Technology. Available: http://www.sun.com/java/

[12] WEB3D Consortium. (2010, May 30). What is X3D?. Available: http://www.web3d.org/

[13] V. Geroimenko and C. Chen, “Visualizing Information Using SVG and X3D”, Springer, 2004.

[14] R. A. Zwirtes. “Cinemática Inversa para Controle da Abordagem de Órgãos Terminais de Robôs Manipuladores”. Undergrad Dissertation, Computer Science. Universidade do Estado de Santa Catarina, 2004. (In Portuguese).

[15] C. Twigg. Catmull-Rom splines [Online]. Available: http://graphics.cs.cmu.edu/nsp/course/15-462/Fall04/assts/catmullRom.pdf. 2003.

[16] ESHED ROBOTEC. Scorbase User Manual. Catalog #100342 Rev. E, 92 p, 2003.

[17] Virbot4u. (2010, Jul 10). Um simulador de robô usando X3D. Available: http://www2.joinville.udesc.br/~larva/virbot4u/

[18] The Xj3D Project. (2010, Jun 7). A toolkit for VRML97 and X3D written in Java. Available: http://www.xj3d.org/