Rev. Roum. Sci. Techn.– Électrotechn. et Énerg.
Vol. **65**, *1-2*, pp. 117–121, Bucarest, 2020

EFFICIENT TRAJECTORY RECONSTRUCTION FOR ROBOT PROGRAMMING BY DEMONSTRATION

REDA HANIFI ELHACHEMI AMAR[1], LAREDJ BENCHIKH[2], HAKIMA DERMECHE[1], OUAMRI BACHIR [3], ZOUBIR AHMED-FOITIH[1]

Key words: Programming by demonstration, Interpolation, Motion capture, Trajectory reconstruction.

The reproduction of hand movements by a robot remains difficult and conventional learning methods do not allow us to faithfully recreate these movements because it is very difficult when the number of crossing points is very large. Programming by demonstration gives a better opportunity for solving this problem by tracking the user's movements with a motion capture system and creating a robotic program to reproduce the performed tasks. This paper presents a programming by demonstration system in a trajectory level for the reproduction of hand/tool movement by a manipulator robot; this was realized by tracking the user's movement with the ArToolkit using a normal camera (low cost) and reconstruct the trajectories by using the constrained cubic spline, which gives better results comparing the conventional cubic spline interpolation. And finally the obtained trajectories will be simulated in a virtual environment on a robot (Puma 600) based on a computed torque controller.

1. INTRODUCTION

Since the invention of the first robots, the reproduction of human movement is still a challenging subject in robotics. This reproduction can be divided into two categories: the first one is imitation of the human movement as it is by the robot to realize the task. References [1, 2] they imitate the human movement to realize different tasks like writing and opening doors, their approach was derived from the human functioning based on the body schema and the percept. The second reproduces the human movement by taking as reference only the hand (the end-effector) and neglecting how and where the other joints are positioned (shoulder, elbow, and wrist). In Benchikh's work, in which he realized a system of a synchronous reproduction of the human movement, following the movement of a single point of interest [3].

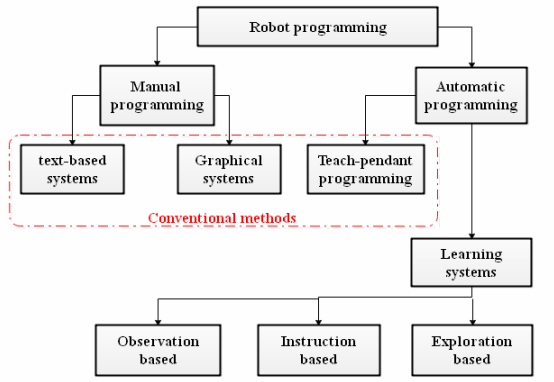

Fig. 1 – Robot programming methods.

One can notice two categories of robot programming methods, the first one is manual like the text based systems, graphical systems and the teach-pendant programming; the second one is the automatic programming such as the instruction and the observation based programming. The Fig. 1 represents the categorization of these methods made by [4].

For applications such as imitation and reproduction of human movement; conventional learning methods as text programming, graphical systems or the teach-pendant programming are not suitable. Especially when the trajectories to reproduce are complex and the number of crossing points are high. Most of the recent works for human movement reproduction use the robot observation based programming. In this category, we find what is called programming by demonstration or learning by watching in other readings [1–3, 5, 6].

Programming by demonstration (PbD) is a robot programming method based on the extraction of data directly from the visualization of the user's performance. It is a promising automatic method, which can permit to a user with little or no expertise to program robot tasks [7].

In PbD systems, the teacher performs the task while a learning interface records the movement and actions carried out during the performance. Different interfaces can be used kinesthetic guidance teacher moves the robot links manually and the trajectories are recorded [6]. The direct control and teleoperation interfaces are also used in PbD systems [8]. Interfaces based on sensors (vision, magnetic and inertia) are widely used in PbD systems. There is a variety of tracking techniques in vision systems [9, 10] and it has an advantage over the other sensor-based methods, because no sensors need to be attached to the teacher. Vision systems allow the teacher to have more natural performance without being disturbed by the material.

In this paper, we propose a PbD system in trajectory level based on visual tracking system (ARToolKit) to track the hand / tool movement and reproduce these movements by a robot manipulator in a simulated environment. In the next section, we present our visual system.

2. THE TRACKING SYSTEM

For the hand/tool tracking, we propose to use a vision-based tracking system. These systems use image-processing methods to calculate the camera pose relative to real world objects and give the position and orientation of these objects. In our work, we use the ARToolKit for tracking the user's hand/tool.

We fixed the markers on the faces of a cube (the same marker for all faces). This will allow the camera to see at least one marker and permit us to avoid occlusions. Based on the work [11], we have used simple markers with a 30 % border

[1] Université des Sciences et de la Technologie d'Oran, El Mnaouar, BP 1505, Bir El Djirs 31000 (Oran) Algérie
[2] Université d' Evry-Val-d'Essonne, IBISC, Paris, France
[3] Université de Bechar, Faculté de Technologie, Département de génie électrique, Bechar 08000 Algérie

width to have a better detection of the marker and avoid at the maximum the false identifications. In Fig. 2 we can see an example of the markers that are using.

The operator will perform the movements and the system will track the position of the marker for the hand / tool and gives the position (x,y,z) every sampling time (the position is calculated from the centre of the cube).

The acquisition camera was set at 20 frames/s to limit marker miss identifications. The slow frame rate and some miss identifications creates gapes and amplitudes peaks. An interpolation of acquired tracking data is mandatory. In literature, different interpolation have been used, non-uniform rational B-splines (NURBS) in [12] for the trajectory approximation, [13] used the L1 splines for the interpolation and preservation of the trajectory. In Section 5, cubic and constrained interpolations are both used and compared.

3. THE CONSTRAINED CUBIC SPLINE

The principle behind constrained cubic spline is to prevent overshooting and eliminating oscillation by sacrificing smoothness. In this interpolation, we replace the equality of the second order derivatives at every point by a specified first order derivative [14]. The construction of the constrained cubic spline function F_i is based on the following criteria:

- Curves are third order polynomials

$$F_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i; \tag{1}$$

- Curves pass through all the known points;
- First order derivative, is the same for both functions on each side of a point

$$F_i'(x) = F_{i+1}'(x); \tag{2}$$

- Boundary conditions are the same as for the natural cubic spline

$$F_1''(x_0) = F_n''(x_n) = 0; \tag{3}$$

- The second order derivative is replaced by a specified first order derivative at every point.

$$F_i'(x) = F_{i+1}'(x) = F'(x). \tag{4}$$

The main step becomes the calculation of the slope for each point. Naturally, we know the slope will be between the slopes of the adjacent straight lines, and it should approach zero if the slope of either line approaches zero.

Fig. 2 – Example of the ArToolKit markers

$$F'(x) = 2 / \left(\frac{x_{i+1} - x_i}{y_{i+1} - y_i} + \frac{x_i - x_{i-1}}{y_i - y_{i-1}} \right) \tag{5}$$

$F'(x) = 0$, if the slope changes sign at this point.

The equation (5) is used only for the intermediate points; in the end points, we use equations (6) and (7):

$$F_1'(x_0) = \frac{3(y_1 - y_0)}{2(x_1 - x_0)} - \frac{F'(x_1)}{2}, \tag{6}$$

$$F_n'(x_n) = \frac{3(y_n - y_{n-1})}{2(x_n - x_{n-1})} - \frac{F'(x_{n-1})}{2}. \tag{7}$$

In this interpolation, there is no necessity to solve a system of equation because the slope at each point is known. Based on the two adjacent points on each side, we can calculate every spline function; as given by equation (1) by using the equations (8) to (13).

$$F_1''(x_{i-1}) = \frac{2\left(F_i'(x_i) + 2F_i'(x_{i-1})\right)}{(x_1 - x_{i-1})} + \frac{6(y_i - y_{i-1})}{(x_1 - x_{i-1})^2}, \tag{8}$$

$$F_1''(x_i) = \frac{2\left(2F_i'(x_i) + F_i'(x_{i-1})\right)}{(x_1 - x_{i-1})} + \frac{6(y_i - y_{i-1})}{(x_1 - x_{i-1})^2}. \tag{9}$$

Finally, every polynomial is calculated from the following parameters

$$a_i = \frac{F_i''(x_1) - F_i''(x_{i-1})}{6(x_i - x_{i-1})}, \tag{10}$$

$$b_i = \frac{x_i F_i''(x_{i-1}) - x_{i-1} F_i''(x_i)}{2(x_i - x_{i-1})}, \tag{11}$$

$$c_i = \frac{(y_n - y_{n-1}) - b_i\left(x_i^2 - x_{i-1}^2\right) - a_i\left(x_i^3 - x_{i-1}^3\right)}{2(x_n - x_{n-1})}, \tag{12}$$

$$d_i = y_{i-1} - a_i x_{i-1}^3 - b_i x_{i-1}^2 - c_i x_{i-1}. \tag{13}$$

4. THE ROBOT'S MODEL OF MOTION

In our work, we use a manipulator robot model, the Puma 600. Parameters of this robot are shown in table I [15, 16]. Motion model of such a mechanism is usually described by the following matrix equation:

$$\Gamma = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + F(\dot{q}), \tag{14}$$

where:

Γ – vector of actuator joint torque;
$M(q)$ – inertia matrix;
$C(q, \dot{q})$ – vector of centrifugal and Coriolis torque;
$G(q)$ – vector of gravitational torques;
$F(\dot{q})$ – vector of actuator joint friction forces;
$q, \dot{q}, \ddot{q}$ – are respectively, the joint angle, velocity, and acceleration vectors.

To ensure the linearization of the nonlinear system described by the equation (14) in closed loop, we introduce a linearization control system based on exacting knowledge of the robot model and its implementation. In this control system, the loop of the linearization is achieved by choosing a torque Γ applied to the robot, as follow

$$\Gamma = M\left(q\right)\Gamma_0 + C\left(q,\dot{q}\right)\dot{q} + G\left(q\right) + F\left(\dot{q}\right), \quad (15)$$

where Γ_0 is an auxiliary input of the selected controller. A proportional derivative (PD) controller is a typical choice and it is given by the equation

$$\Gamma_0 = \ddot{q}_d + K_v\left(\dot{q}_d - \dot{q}\right) + K_p\left(q_d - q\right). \quad (16)$$

By replacing $\ddot{q} = \Gamma_0$ in the equation (16), we get:

$$\Gamma_0 = \ddot{e} + K_v\dot{e} + K_p e, \quad (17)$$

where:

$e = q_d - q$ – vector of the position error;

$\dot{e} = \dot{q}_d - \dot{q}$ – vector of the velocity error;

$\ddot{e} = \ddot{q}_d - \ddot{q}$ – vector of the acceleration error;

$q_d, \dot{q}_d, \ddot{q}_d$ – are vectors of desired position, respectively velocity and acceleration.

K_p, K_v – gain matrices of the PD controller.

The error eq. (17) is a linear differential equation of second order where K_p and K_v are defined positive diagonal matrices, so the closed-loop system becomes linear decoupled

$$K_p = \begin{pmatrix} 350 & 0 & 0 \\ 0 & 350 & 0 \\ 0 & 0 & 350 \end{pmatrix}, \quad K_v = \begin{pmatrix} 35 & 0 & 0 \\ 0 & 35 & 0 \\ 0 & 0 & 35 \end{pmatrix}. \quad (15)$$

In Fig. 3 we can see the implementation of the computed torque controller on the Puma robot.

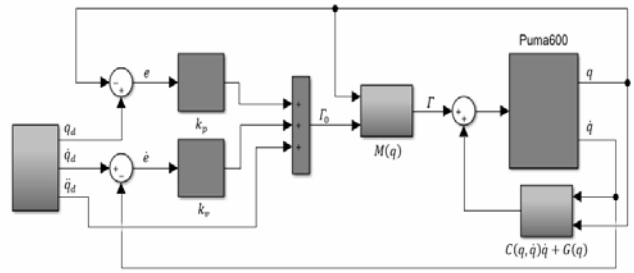

Fig. 3. – Implementation of the computed torque controller.

Table 1

Parameters of the Puma 600 manipulator robot

Parameters	Values
Mass of the first body	10,521 kg
Mass of the second body	10,236 kg
Mass of the third body	8,767 kg
Coefficient of viscous friction	2,52 N.m.s/rad
Coefficient of viscous friction	7 N.m.s/rad
Coefficient of viscous friction	1,75 N.m.s/rad
Coefficient of dry friction	3,6 N.m.s/rad
Coefficient of dry friction	10 N.m.s/rad
Coefficient of dry friction	2,5 N.m.s/rad
Length of the first body	0,149 m
Length of the second body	0,432 m
Length of the third body	0,431 m
Mass of the first body	10,521 kg
Mass of the second body	10,236 kg
Mass of the third body	8,767 kg
Coefficient of viscous friction	2,52 N.m.s/rad

5. SIMULATION AND RESULTS

Since the trajectories obtained from the tracking system should be interpolated, we started by testing the constrained cubic spline interpolation on different data sets. The Figs. 4 and 5 show a comparison between the constrained cubic in blue line and spline and the conventional cubic spline interpolation in red line, red circles are the interpolated data points.

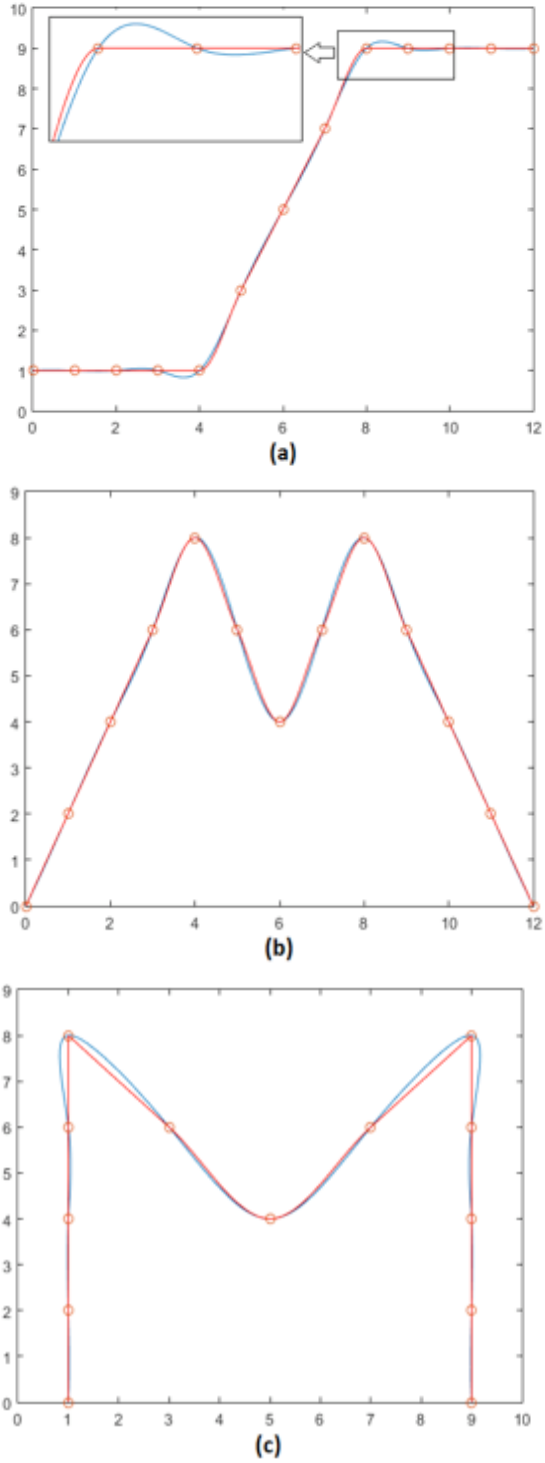

Fig. 4. – Trajectory reconstruction using cubic spline and constrained cubic spline interpolations.

In the first tests, we used both of the interpolation methods to reconstruct handwriting trajectory made at almost the same speed. We have chosen to reconstruct the letter M using both of the cubic spine interpolation and the constrained cubic spline because this letter contains sudden directional changes, which will help us to see the behavior of the used interpolation method.

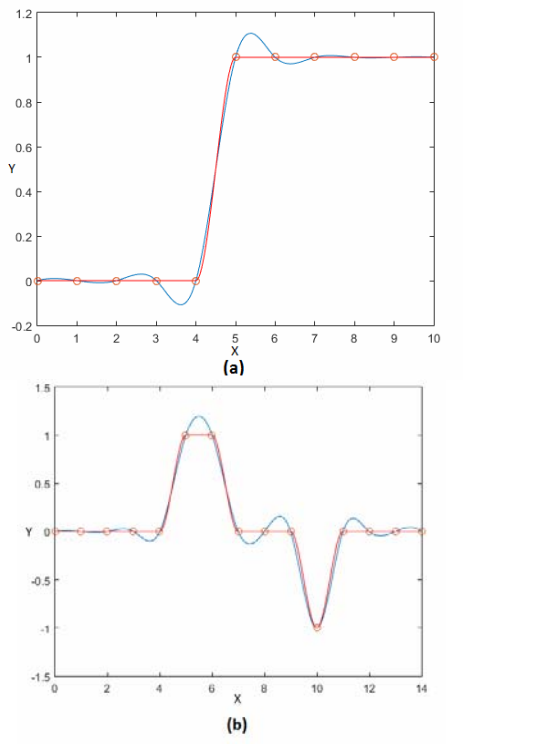

Fig. 5. – Interpolation of different data sets.

For the results in the Fig. 4 b we see that both of the methods gave the same results this due to the fact that there was no large data variation but in the Fig. 4 a we can see an oscillation and an overshooting of the reconstructed data. Figure 4 c shows the result of the reconstructed trajectory where we can clearly see that constrained cubic spline gave the better results and the oscillation of the cubic spine interpolation affected the obtained trajectory. For Fig. 5 we have used the following data:

(a): $x = [0,1,2,3,4,5,6,7,8,9,10]$; $y = [0,0,0,0,0,1,1,1,1,1,1]$;

(b): $x = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]$;
$y = [0,0,0,0,0,1,1,0,0,0,–1,0,0,0,0]$.

In Fig. 5 it is noted that the constrained cubic spline does not oscillate after large amplitude variation and reduces the overshooting comparing to the cubic spline. This fact makes the constrained cubic spline more stable and has less oscillation, which makes it better for the reconstruction of the trajectories.

In Fig. 6 one can see a handwriting trajectory reconstructed with the constrained cubic spline interpolation.

The interpolation programs were made with Matlab on a computer with the following configuration: I7 3770 3.4 GHz and 8 Gb RAM. Table 2 shows interpolation execution time according to interpolated point number.

Table 2

Parameters of the Puma 600 manipulator robot

Number of points	Constrained spline (ms)	Cubic spline (ms)
25	1.383	2.332
50	1.900	2.481
100	2.502	3.121
200	4.084	3.390

One can notice two behaviours. The first is for number of points below 200: the constrained cubic spline is faster than the cubic spline since it does not solve a system of equations. The second case is when the number of interpolated points exceeds 200 the cubic spline overcomes the constrained cubic spline.

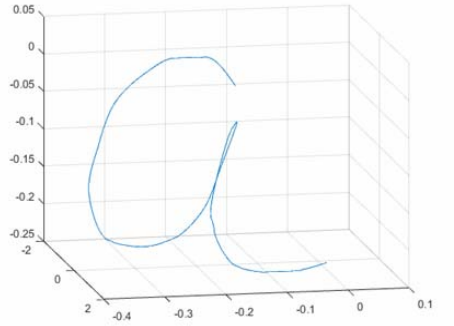

Fig. 6 – 3D Trajectory reconstructed by the constrained cubic spline.

Figure 7 shows a simulation of Puma 600 robot executing the same trajectory. The advantage of the simulation process is reconstructed path visualization. The proposed system depends on the teacher, and do not generate collision-free trajectories.

6. CONCLUSIONS

In this article, we presented a robot programming by demonstration system in a trajectory level. The system is based on human movement reproduction by following a single point of interest (the hand/tool). We used ARToolKit as a visual tracking system. The slow frame rate and the miss identifications of the markers was creating gapes in the trajectories. The natural cubic spline and constrained cubic spline interpolations were used to correct these weaknesses and to reconstruct the trajectory. A comparison

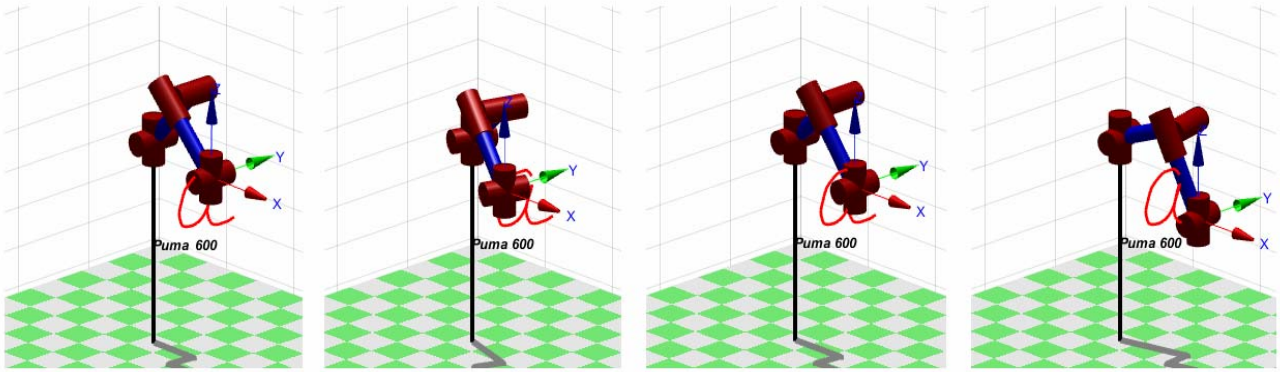

Fig. 7. – Simulation of robot trajectory reproduction.

was made between those both interpolation methods for 3D trajectories reconstruction, 1D data sets and as per execution time. Tests show that constrained cubic spline interpolation overcomes the cubic spline interpolation. The constrained cubic spline interpolation shows good results and the fact that it generates less oscillation and prevents overshooting makes it suitable for the trajectory reconstruction. In the future works this it will be used in a real time trajectory reconstruction.

Received on July 2, 2018

REFERENCES

1. C. A. Acosta-Calderon, H. Hu, *Robot imitation: Body schema and body percept*, Applied Bionics and Biomechanics, **2**, pp. 131–148 (2005).
2. C. A. Acosta-Calderon, H. Hu, *Robot imitation from human body movements*, The 3rd International Symposium on Imitation in Animals and Artifacts, UK, 2005, pp. 1–9.
3. L. Benchikh, *Method for training a robot or the like, and device for implementing said method*, U.S. Patent Application No 12/812 792 (2011).
4. G. Biggs, B. MacDonald, *A survey of robot programming systems*. Proceedings of Australasian Conference on Robotics and Automation, Brisbane, Australia, 2003, pp. 1–10.
5. Y. Kuniyoshi, M. Inaba, H. Inoue, *Learning by watching: Extracting reusable task knowledge from visual observation of human performance*, IEEE Transactions on Robotics and Automation, **10**, pp. 799-822 (1994).
6. S. Calinon, A. Billard, *A probabilistic programming by demonstration framework handling constraints in joint space and task space*, International Conference on Intelligent Robots and Systems IROS, Nice, France, 2008, pp. 367–372.
7. J. Aleotti, S. Caselli, M. Reggiani, *Leveraging on a virtual environment for robot programming by demonstration,* Robotics and Autonomous Systems, **47**, pp. 153–161 (2004).
8. M. Shimizu, W. Yoon, K. Kitagaki, *Experimental validation of task skill transfer approach using a humanoid robot*, International Symposium on Assembly and Manufacturing ISAM'07, 2007, pp 141–146.
9. A. Tomescu, F. M. G. Tomescu, *Existence and uniqueness of weak solutions of the induced current reaction problem* (*Part I*: *Electric field problem*), Rev. Roum. Sci. Techn. – Électrotechn. Et Énerg., **39**, *1*, pp. 25–44 (1993).
10. M. Kurien, M. K. Kim, M. Kopsida, I. Brilakis, *Real-time simulation of construction workers using combined human body and hand tracking for robotic construction worker system.* Automation in Construction, **86**, pp. 125–137 (2018).
11. P. P. Valentini, , *Natural interface for interactive virtual assembly in augmented reality using Leap Motion Controller,* International Journal on Interactive Design and Manufacturing (IJIDeM), pp. 1–9 (2018).
12. D. Khan, S. Ullah, I. Rabbi, *Factors affecting the design and tracking of ARToolKit markers*, Computer Standards & Interfaces, **41**, pp. 56–66 (2015).
13. J. Aleotti, S. Caselli, *Robust trajectory learning and approximation for robot programming by demonstration*, Robotics and Autonomous Systems, **54**, pp. 409–413 (2006).
14. F. Hernoux *et al.* , *Leap Motion pour la capture de mouvement 3D par spline L1*, Journées du Groupe de Travail en Modélisation Géométrique, Marseille, 2013, France.
15. C.J. Kruger, *Constrained cubic spline interpolation,* Chemical Engineering Applications, 2003.
16. B. Ouamri, Z. Ahmed-Foitih, *Adaptive neuro-fuzzy inference system based control of Puma 600 robot manipulator*, International Journal of Electrical and Computer Engineering, **2**, pp. 90–97 (2012).
16. B. Ouamri, Z.Ahmed-Foitih, *Computed Torque Control of a Puma 600 Robot by using Fuzzy Logic,* International Review of Automatic Control, **4**, pp. 248–252 (2011).