Flexible Path Planning based on the deformation of a Bézier curve through a field of vectors.

L.Hilario[1],N.Montés[1], M.C.Mora[2], A.Falcó[1]
1Cardenal Herrera CEU University, C/San Bartolomé 55 46115,Alfara del Patriarca, Spain.
2 Mechanical Engineering and Construction Department, Universitat Jaume I, Castellón,Spain.
luciah@uch.ceu.es , nimonsan@uch.ceu.es, mmora@emc.uji.es , afalco@uch.ceu.es

1.- ABSTRACT.

This article presents a new technique for flexible path planning based on the deformation of a Bézier curve through a field of vectors. This new technique is called Bézier Shape Deformation (BSD). This deformation is computed with a constrained optimization method (Lagrange Multipliers Theorem). The main advantage of this method is how the solution is obtained. A linear system is solved to achieve the result. As a consequence, the deformed curve is computed in a few milliseconds where the linear system can be solved offline if the Bézier curve order is maintained constant during the movement of the robot. This method allows the use of these trajectories in dynamic environments where the computational cost is critical. This technique can be combined with any collision avoidance algorithm that produces a field of vectors. In particular, it is appropriate for artificial potential field methods. At the end of the paper, the presented methodology is combined with an artificial potential fields algorithm recently proposed, the Potential Field Projection method (PFP). This method is based on the combination of the classical Potential Fields method and the multi-rate Kalman filter estimation and takes into account the uncertainties on locations, the future trajectory of the robot and the obstacles and the multi-rate information supplied by sensors. As shown in the simulation results, flexible trajectories for collision avoidance are generated with smooth curves.

2.- INTRODUCTION.

A robot can be defined as a machine that should be able to collect information as well as to interact in a natural way with the surrounding environment. The key idea is how to obtain a proper trajectory that must be smooth and must interact with the environment in order to guarantee that the path is free of collisions in real-time.

Collision avoidance is one of the main goals in the research carried out in industrial applications. In this sense, parametric curves are one of the most used options to represent the trajectories because produce inherently smooth paths. The most commonly used in robotics are: B-Splines, NURBS, Bézier and Rational Bézier.

A B-Spline (defined in [2]) is a piecewise polynomial function that can have a locally very simple form, yet at the same time be globally flexible and smooth. Splines are very useful for modelling arbitrary functions, and are used extensively in computer graphics design. There exit a few path planning algorithms which incorporate a spline-based methodology. It is possible to find some works, as those explained next.
One of the first publication that incorporated the B-Spline as a parametric curve in path planning is the work published by Komoriya and Tanie [32]. In this case, they added straight-line segments to make the overall trajectory close to the intended one. However,

the trajectory did not pass through the exact waypoint. After that it is possible to find an study discussed by Vazquez and Humberto [33]. They used B-Splines in path planning by adding a time variable. The speed of the mobile robot was controlled by the given B-Spline itself. Zhang et. al [34] created a fuzzy trajectory controller, which emulates the B-Spline. Eren et. al [35] presented an implementation of a Spline Curve Algorithm in a mobile robot path planning domain. They generated several interior points within a Spline and let the robot follow the points in succession. The path could easily be updated and modified as the robot moves. This algorithm was suitable for dynamic control using multiple mobile robots operating in the same environment. Yamamoto et. al [36] considered the dynamic and kinematic constraints in B-Spline path planning algorithms to find the best possible time trajectory in a static environment. Connors and Elkaim, [37], [38], [39] developed a method for solving path planning problems by bending smooth cubic Splines to avoid obstacles. This method iteratively refines the path to compute a feasible and collision-free path in real time through an unstructured environment. The use of cubic Splines guarantees continuous vehicle heading and turning angle which is ideal for many ground based vehicles.

A parametric curve defined by rational coordinate functions is called a rational curve. Rational B-Splines curves are obtained from B-Splines. They are generally referred to as NURBS which stands for Non-Uniform Rational B-Splines. This kind of parametric curves is used for trajectory reconstruction. Piegl [8] and Tatematsu and Ohnishi [19] described the NURBS curves as the best parametric curves for path planning for 2D and 3D curve approximation. Alleotti and Caselli [20], [21], [22] have been working with NURBS curves to approach the trajectory described by a robot arm PUMA 560. To obtain the robot behaviour they use the Programming by Demonstration (PbD) method, a promising solution for automatic transfer of knowledge from a human to a robot. The NURBS curves are used for trajectory reconstruction to generate a smooth path that approximates the actual motion.

Bézier curves were defined by Pierre Bézier in 1962 to design automobile bodies. Nowadays, these kind of parametric curves are widely used in computer graphic design and animation. Furthermore, Bézier curves are also used as trajectories in mobile robots. Khatib et. al [23], proposed the bubbled band concept with an appropriate metric, connecting the bubbles with Bézier curves. They use Bézier curves by their interesting properties, for example, the convex hull's property. A Bézier curve lies in the convex hull envelop of its control points. It implies that if the control points are located inside the bubble, then the path approximation is inside the bubble. After this publication, there is another work of Graf et. al [24]. This work presents a flexible path planning method for nonholonomic mobile robots. The generated path is smoothened and eventually modified in reaction to dynamic obstacles using the method of elastic band. This elastic band is represented as a sequence of connected bubbles. A Bézier algorithm is used to track the path smoothly. Moreover Nagatani et. al [25], used a path planning algorithm based on Bézier curves considering the minimum radius of curvature of the vehicle. Subsequently, Hwang et. al [26] presented a new interfacing method using a touchpad/screen to control a mobile robot, capable of real-time dynamic obstacle avoidance as well. They developed two algorithms: a significant points extraction algorithm for noisy input data and an on-line piecewise cubic Bézier curves trajectory generation algorithm for a mobile robot. Skrjanc and Klancar [27], developed a new cooperative collision avoidance method for multiple nonholonomic robots with constraints and known start and goal velocities based on Bernstein-Bézier curves. The minimization problem used is an inequality optimization problem. The objective function minimizes the sum of all absolute maximal times subjected to the distances between the robots. One year later, Lizarraga and Elklaim [28] used Bézier curves for generating spatially deconflicted paths for multiple UAVs (unmanned aerial vehicles). The critical issue addressed is that of guaranteeing that all the paths lie inside a predefined airspace volume. Bézier curves represent a natural tool for meeting this requirement (convex hull's property). The path generation problem is formulated as a constrained optimization problem over a finite optimization set and solved using standard MATLAB optimization tools. In this case, the cost function penalizes excessive lengths; the vehicles must use the shortest possible path. The constraint of the problem is the distance between the multiple vehicles, as the generated paths must have a minimum separation among them. The problem is non-linear and the solution is obtained by numerical techniques. Recent works that we can find are discussed by Choi et. al [29], [30], [31]. They presented two path planning algorithms based on Bézier curves for autonomous vehicles with waypoints and corridor constraints. In this paper a mathematical problem of constraint optimization is used. The cost function is the curvature of the Bézier curve to obtain resulting paths with larger radii of curvature.

Besides this, there exists a parametric curve defined by rational coordinate functions, called a Rational Bézier Curve (RBC). This curve was used by Montés et al. [18] in order to obtain the best possible approximation of a clothoid using the weights of the RBC. The clothoidal path planning was easily developed using the Curve properties because is invariant under rotation, translation and scaling.

The main difference between all the parametric curves (B-Splines, NURBS, RBC and Béziers) is the complexity of their mathematical definition. While Bézier curves are the simplest ones, B-Splines or NURBS are more complex although they can more accurately represent objects in CAD programs. However, in mobile robotics real-time applications are required and, for this reason, more researchers use Bézier curves.

3.- OBJECTIVE AND OUTLINE.

Through all these publications we can conclude that an ideal solution is having a mathematical tool to obtain the modification of the predefined trajectory, parametric curve, obtaining the modified curve that ensures collision-free navigation in real-time.

In other research fields like CAD/CAGD programs, the shape modification of the parametric curves is a recently research topic, see Meek et. al [13] for NURBS shape modification and Fowler and Bartels [14] for B-Spline shape modification.

In Xu et. al [15] the Bézier shape modification by constrained optimization based on the discrete coefficient norm is discussed. The problem of parametric curves shape modification by constrained optimization was proposed recently. Wu and Xia [16] developed a new technique to modify a Bézier curve by minimizing the changes of the shape. This result was used by Montés et. al [17] and it was improved and developed for Liquid Composite Moulding Processes. This is our previous work, defined as Bézier Shape Deformation (BSD). This method computes the modification of the shape defining a constraint optimization problem solved by the Lagrange Multipliers theorem. This theorem needs a cost function that, in this case, is defined to minimize the distance between two Bézier curves and a set of constraints are used to obtain our aim.

The principal constraint is how the modified curve passes through the “Target Point”. The rest of constraints are necessary to obtain a smooth curve, for example, continuity and derivability on the full curve.The advantage of this method is the possibility to add as many different constraints as necessary to obtain the best solution.

The current paper proposes a new path planning algorithm that has the ability to interact with the environment. This fact is able due to the combination of a proposed mathematical tool, called BSD, and an algorithm that produces forces to guide the robot to the goal avoiding the obstacles in the environment. In the present work we use a very recent study in this field, Mora et al. [40], [41].

This paper is organized as follows: Section 3 defines the Bézier curve and its useful properties for path planning. Section 4 explains the objective and the outline of the study. Section 5 develops the new technique called Bézier Shape Deformation (BSD) applied for mobile robots. In Section 6, simulation results are given. Finally, Section 7 provides the conclusions and future work. At the end of the work as cited the references and the acknowledgments.

4.- DEFINITIONS AND PRELIMINARY NOTATION.

The Bézier curve of degree n, given $(n+1)$ control points, $\mathbf{P}_i$, is defined to be (see [1], [2], [3], [4]).

$$\boldsymbol{\alpha}(t)=\sum_{i=0}^{n}\mathbf{P}_i\cdot B_{i,n}(t);t\in[0,1] \tag{1}$$

where t is a normalized time variable, $0\leq\frac{t}{T_{\max}}\leq 1$, and $B_{i,n}(t)$, are called the Bernstein polynomials or Bernstein Basis functions of degree n.

$$B_{i,n}(t)=\begin{cases}\binom{n}{i}(1-t)^{n-i}t^{i};0\leq i\leq n\\ 0,\text{ otherwise}\end{cases} \tag{2}$$

Bézier curves have useful properties for path planning because:

1.- A Bézier curve always pass through the first and the last control points.
2.- A Bézier curve lies within the convex hull of the control points.
3.- A Bézier curve is tangent to the vector of the difference $\mathbf{P}_1-\mathbf{P}_0$ at the start point and the vector of the difference $\mathbf{P}_n-\mathbf{P}_{n-1}$.
4.- A Bézier curve can be translated and rotated by performing the last properties on the control points.

Note: The intrinsic parameter, t, is a non-dimensional parameter and could be re-defined in order to have dimensional sense. In this case, the definition of the Bézier curve changes as shown in (3) and (4).

$$\boldsymbol{\alpha}(u) = \sum_{i=0}^{n} \mathbf{P}_i \cdot B_{i,n}(u); u \in [u_0, u_f] \tag{3}$$

$$B_{i,n}(u) = \binom{n}{i} \cdot \left(\frac{u_f - u}{u_f - u_0} \right)^{n-i} \cdot \left(\frac{u - u_0}{u_f - u_0} \right)^{i}; i = 0,1,..,n \tag{4}$$

The displacement of every control point is denoted as: $\underline{\underline{\boldsymbol{\varepsilon}}} = [\boldsymbol{\varepsilon}_0 \cdots \boldsymbol{\varepsilon}_n]$. The Bézier curve obtained modifying its controls points, $S_{\varepsilon}(\boldsymbol{\alpha}(t))$, is defined to be

$$S_{\varepsilon}(\boldsymbol{\alpha}(t)) := \sum_{i=0}^{n} (\mathbf{P}_i + \boldsymbol{\varepsilon}_i) \cdot B_{i,n}(t) \tag{5}$$

5.- BÉZIER SHAPE DEFORMATION FOR MOBILE ROBOTS.

5.1.- MATHEMATICAL MODEL.

To deform a given Bézier curve, we only need to change the control points, and then it is necessary to compute the perturbation ε_i of every control point. For that, a constraint optimization problem is proposed. Then, it is necessary to define a cost function to optimize and a set of constraints to obtain the desirable solution.
The cost function to optimize is defined as follows;

$$\min_{\underline{\underline{\boldsymbol{\varepsilon}}}} \int_0^1 \left\| S_{\varepsilon}(\boldsymbol{\alpha}(t)) - \boldsymbol{\alpha}(t) \right\|_2^2 dt = \min_{\underline{\underline{\boldsymbol{\varepsilon}}}} \int_0^1 \left\| \sum_{i=0}^{n} \boldsymbol{\varepsilon}_i \cdot B_{n,i}(t) \right\|_2^2 dt = \min_{\underline{\underline{\boldsymbol{\varepsilon}}}} \int_0^1 \left(\sum_{i=0}^{n} \boldsymbol{\varepsilon}_i \cdot B_{n,i}(t) \right)^2 dt \tag{6}$$

This function minimizes the changes of the shape minimizing the distance between both curves see Figure 1.

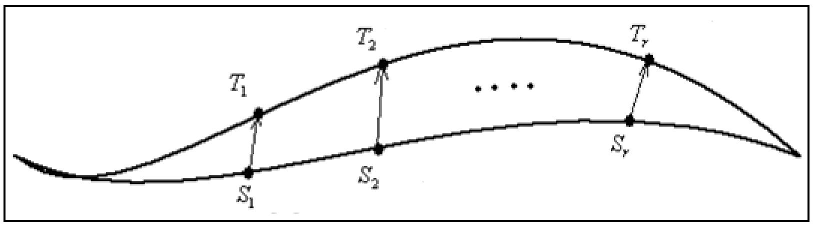

Figure 1
The deformation of a Bézier curve.

Bézier curve constructed using a large number of control points is numerically unstable, see Skrjanc and Klancar, [27]. In addition to this, the order of the Bézier curve must not be third or higher because it produces a loop or a cusp depending of the geometrical location of the control points. In order to avoid this undesirable property it is necessary to concatenate two or more Bézier curves of second order to represent the full trajectory joining the initial and final positions of the mobile robot. Then, the cost function must be defined as follows,

$$\min_{\underline{\boldsymbol{\varepsilon}}^{(1)}\cdots\underline{\boldsymbol{\varepsilon}}^{(k)}} \int_0^1 \sum_{l=1}^{k} \left\| \sum_{i=0}^{n_l} \boldsymbol{\varepsilon}_i^{(l)} \cdot B_{i,n_l}(t) \right\|_2^2 dt = \min_{\underline{\boldsymbol{\varepsilon}}^{(1)}\cdots\underline{\boldsymbol{\varepsilon}}^{(k)}} \sum_{l=1}^{k} \int_0^1 \left(\sum_{i=0}^{n_l} \boldsymbol{\varepsilon}_i^{(l)} \cdot B_{i,n_l}(t) \right)^2 dt \tag{7}$$

In order to develop a proper navigation algorithm some set of constraints must be satisfied:

1.-As we can see in Figure 1 a field of vectors connect the Start Points (points on the curve), $\mathbf{S}_i$, with the Target Points (points on the modified curve), $\mathbf{T}_i$. The modified Bézier curve $\mathbf{S}_{\boldsymbol{\varepsilon}}(\alpha(t))$ passes through the Target Point **T**. Thus, the curve must curve must guarantee the following constraint:

$$\mathbf{T}_1^{(l)} - S_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_l\left(t_1^{(l)}\right)\right) = 0, \cdots, \mathbf{T}_k^{(l)} - S_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_l\left(t_k^{(l)}\right)\right) = 0 \tag{8}$$

2.-To maintain the derivative property of the curve, derivative restrictions in the start and end points of the resulting concatenated curve must be imposed to the constrained optimization method,

$$\boldsymbol{\alpha}_k{}'(1) - \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_1'(1)\right) = 0 \Rightarrow n_k \cdot \left(\boldsymbol{\varepsilon}_{n_k}^{(k)} - \boldsymbol{\varepsilon}_{n_k-1}^{(k)}\right) = 0 \tag{9}$$

3.- Continuity has to be imposed on the joint points of the concatenated curves,

$$\mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_1(1)\right) - \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_2(0)\right) = 0, \cdots, \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_{k-2}(1)\right) - \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_{k-1}(0)\right) = 0 \tag{10}$$

The evaluation of these equations developed like this generic case,

$$\mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_l(1)\right) = \mathbf{P}_{n_l}^{(l)} + \boldsymbol{\varepsilon}_{n_l}^{(l)}, \cdots, \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_{l+1}(0)\right) = \mathbf{P}_0^{(l+1)} + \boldsymbol{\varepsilon}_0^{(l+1)} \tag{11}$$

4.- Derivability in the joint points of the concatenated curves is also required to guarantee the smoothness of the trajectory,

$$\mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_1'(1)\right) - \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_2'(0)\right) = 0, \cdots, \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_{k-2}'(1)\right) - \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_{k-1}'(0)\right) = 0 \tag{12}$$

The evaluation of these constraints is compute as,

$$\begin{aligned} \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_l'(1)\right) &= n_l \cdot \left[\left(\mathbf{P}_{n_l}^{(l)} - \mathbf{P}_{n_l-1}^{(l)}\right) + \left(\boldsymbol{\varepsilon}_{n_l}^{(l)} - \boldsymbol{\varepsilon}_{n_l-1}^{(l)}\right)\right] \\ \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_{l+1}'(0)\right) &= n_{l+1} \cdot \left[\left(\mathbf{P}_1^{(l+1)} - \mathbf{P}_0^{(l+1)}\right) + \left(\boldsymbol{\varepsilon}_1^{(l+1)} - \boldsymbol{\varepsilon}_0^{(l+1)}\right)\right] \end{aligned} \tag{13}$$

The Lagrange Multipliers theorem is applied to solve the constrained optimization problem. The constraints $r_i, i = 1,..,4,$ are added to the cost function, g_1, resulting in the Lagrange function defined as,

$$L(\boldsymbol{\varepsilon}_i) = g_1 + r_1 + r_2 + r_3 + r_4 \tag{14}$$

The cost function is:

$$g_1 = \int_0^1 \sum_{l=1}^{k} \left\| \sum_{i=0}^{n} \boldsymbol{\varepsilon}_i^{(l)} B_{n_l,i}(t) \right\|_2^2 dt \tag{15}$$

The constraints are included in the Lagrange function in the following form,

$$r_1 = \sum_{l=1}^{k} \sum_{j=1}^{r_l} \left\langle \boldsymbol{\lambda}, \mathbf{T}_j^{(l)} - \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_l\left(t_j^{(l)}\right)\right)\right\rangle \tag{16}$$

$$r_2 = \left\langle \boldsymbol{\lambda}, \boldsymbol{\alpha}_1^{'}(0) - \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_1^{'}(0)\right)\right\rangle + \left\langle \boldsymbol{\lambda}, \boldsymbol{\alpha}_k^{'}(1) - \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_k^{'}(1)\right)\right\rangle \tag{17}$$

$$r_3 = \sum_{l=1}^{k-1} \left\langle \boldsymbol{\lambda}, \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_l(1)\right) - \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}_{l+1}(0)\right)\right\rangle \tag{18}$$

$$r_4 = \sum_{l=1}^{k-1} \left\langle \boldsymbol{\lambda}, \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}'_l(1)\right) - \mathbf{S}_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\alpha}'_{l+1}(0)\right)\right\rangle \tag{19}$$

Thus, the L function depends on the previous variables,

$$L = L\left(\boldsymbol{\varepsilon}^{(1)}, \ldots, \boldsymbol{\varepsilon}^{(k)}, \boldsymbol{\lambda}\right) \tag{20}$$

The number of the constraints depends on the number of the concatenated Bézier curves,

$$\sum_{l=1}^{k} r_l + 1 + 1 + (k-1) + (k-1) \tag{21}$$

The problem is solved making zero the partial derivatives of the Lagrange function,

$$\begin{cases} \dfrac{\partial L}{\partial \boldsymbol{\varepsilon}^{(l)}} = 0 \\ \dfrac{\partial L}{\partial \boldsymbol{\lambda}} = 0 \end{cases} \tag{22}$$

In this way, a linear system of equations is obtained. The matrix formulation of this system defined as,

$$\mathbf{A}\cdot\mathbf{X}=\mathbf{b} \tag{23}$$

The characteristics of this linear system are:

1.- The variable vector is a column vector,

$$\mathbf{X}^T=\left[\boldsymbol{\varepsilon}^{(1)},\cdots,\boldsymbol{\varepsilon}^{(k)},\boldsymbol{\lambda}\right] \tag{24}$$

2.- The independent vector is a column vector too,

$$\mathbf{b}^T=\left[\mathbf{0},\mathbf{v}^{(1)}\cdots\mathbf{v}^{(k)},\mathbf{0},\mathbf{C}\right] \tag{25}$$

The terms of $\mathbf{b}^T$ are defined as,

$$\mathbf{v}^{(l)}=\left[\left(\mathbf{T}_1^{(l)}-\mathbf{S}_1^{(l)}\right)\cdots\left(\mathbf{T}_{r_l}^{(l)}-\mathbf{S}_{r_l}^{(l)}\right)\right] \tag{26}$$

$$\mathbf{C}=\left[\mathbf{C}_{0,(2,1)},\mathbf{C}_{1,(2,1)},\cdots,\mathbf{C}_{0,(k,k-1)},\mathbf{C}_{1,(k,k-1)}\right] \tag{27}$$

$$\begin{gathered}\mathbf{C}_{0,(i,(i-1))}=\mathbf{P}_0^{(i)}-\mathbf{P}_{n_i}^{(i-1)}\\ \mathbf{C}_{1,(l,(l-1))}=n_{l-1}\cdot\left(\mathbf{P}_{n_{l-1}-1}^{(l-1)}-\mathbf{P}_{n_l}^{(l-1)}\right)+n_l\cdot\left(\mathbf{P}_1^{(l)}-\mathbf{P}_0^{(l)}\right)\end{gathered} \tag{28}$$

3.- The matrix of the system is defined in (29). It is a square matrix defined as a block matrix. This form allows adding new blocks that belong to other constraints.

$$\mathbf{A}=\begin{pmatrix}\mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} & \mathbf{A}_{14}\\ \mathbf{A}_{21} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{A}_{31} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{A}_{41} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{pmatrix} \tag{29}$$

The solution of this system is obtained as,

$$\mathbf{X}=\mathbf{A}^{-1}\mathbf{b} \tag{30}$$

The most important advantage of the proposed method is that the solution obtained to get the deformed curve is not an approximation. It is an accurate solution. Another important thing is its low computational cost. The deformation of the initial robot path can be done in real time because the problem to solve is a linear system and the inverse matrix $\mathbf{A}^{-1}$ can be computed in advance.

In Figure 2 it is shown a numerical result of the BSD algorithm. The modified trajectory is computed in 0.23 ms in a Pentium IV 2.4 Ghz.

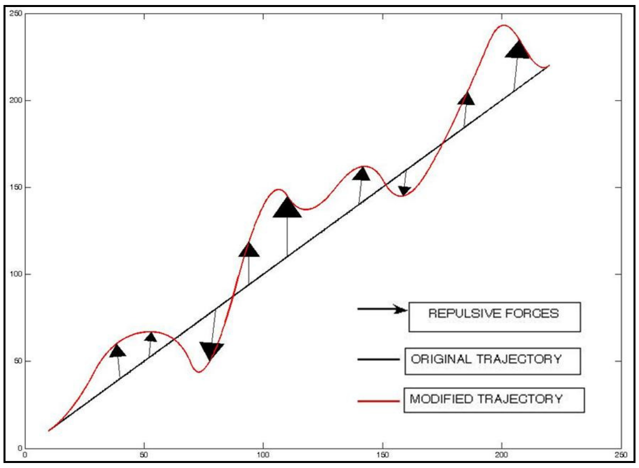

Figure 2
Numerical Result of the BSD with ten Bézier curves of second order.

The method developed in this paper guarantees the accuracy of the solution proposed if and only if the number of vectors is less than the order of the Bézier curve deformed by vectors. If the number of vectors is equal or upper than the order of the Bézier curve there are too many constraints to obtain the solution.

5.2.- POTENCIAL FIELD PROJECTION FOR GENERATING A FIELD OF VECTORS.

The technique explained above requires an algorithm that generates the field of vectors to ensure that the deformed trajectory is collision-free. In the present study, the technique explained above is evaluated through a predictive path planning method described in Mora et al. [40] and Mora and Tornero [41], the Potential Field Projection method (PFP). This method is based on the combination of the classical Potential Fields method Khatib [42] and the multi-rate Kalman filter estimation Tornero et al. [43] and Pizá [44] and takes into account the uncertainties on locations, the future trajectory of the robot and the obstacles and the multi-rate information supplied by sensors. The particularities of this approach are described in this section.

Dynamic models of moving objects are essential in the estimation procedure. In this case, particle kinematic models have been used for the robot and the obstacles. Using Euler approximation, this model corresponds to the equations (31) and (32), where the state vector is composed of position *x, y* and velocity v_x, v_y in XY coordinates, *T* is the control or estimation period and the control input is the object acceleration a_x, a_y.

$$\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_{k} + \begin{bmatrix} \frac{T^2}{2} & 0 \\ 0 & \frac{T^2}{2} \\ T & 0 \\ 0 & T \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \end{bmatrix} \tag{31}$$

$$\begin{bmatrix} x \\ y \end{bmatrix}_{k} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_{k} \tag{32}$$

This model responds to the linear state space representation (33), where $\mathbf{x}\in\Re^{n}$ is the state vector, $\mathbf{u}\in\Re^{m}$ is the input vector, $\mathbf{z}\in\Re^{p}$ is the measurement vector, **w** and ***v*** are the process noise and the measurement noise, respectively. On the other hand, $\mathbf{A}\in\Re^{nxn}$, $\mathbf{B}\in\Re^{nxm}$ and $\mathbf{C}\in\Re^{pxn}$ are the state space matrices for linear systems.

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{A}\cdot\mathbf{x}_k+\mathbf{B}\cdot\mathbf{u}_k+\mathbf{w}_k\\ \mathbf{z}_k &= \mathbf{C}\cdot\mathbf{x}_k+\mathbf{v}_k\end{aligned} \tag{33}$$

Predicted future positions and uncertainties are obtained from the prediction equations of the multi-rate Kalman filter (34) for every object in the environment, where $\hat{\mathbf{x}}\in\Re^{n}$ is the state estimation vector, $\mathbf{P}\in\Re^{n\times n}$ is the error estimation variance matrix, $\mathbf{K}\in\Re^{n\times p}$ is the Kalman gain and $\mathbf{Q}\in\Re^{n\times n}$ and $\mathbf{R}\in\Re^{p\times n}$ are, respectively, the process noise and the measurement noise covariance matrices.

$$\begin{aligned}\hat{\mathbf{x}}_{k+1/k} &= \mathbf{A}\cdot\hat{\mathbf{x}}_{k/k}+\mathbf{B}\cdot\mathbf{u}_k\\ \mathbf{P}_{k+1/k} &= \mathbf{A}\cdot\mathbf{P}_{k/k}\cdot\mathbf{A}^{\mathrm{T}}+\mathbf{Q}\\ \mathbf{K}_k &= \mathbf{P}_{k/k-1}\cdot\mathbf{C}^{\mathrm{T}}\cdot\left[\mathbf{C}\cdot\mathbf{P}_{k/k-1}\cdot\mathbf{C}^{\mathrm{T}}+\mathbf{R}\right]^{-1}\cdot\mathbf{\Delta}_k\\ \hat{\mathbf{x}}_{k/k} &= \hat{\mathbf{x}}_{k/k-1}+\mathbf{K}_k\cdot\left[\mathbf{z}_k-\mathbf{C}\cdot\hat{\mathbf{x}}_{k/k-1}\right]\\ \mathbf{P}_{k/k} &= \mathbf{P}_{k/k-1}-\mathbf{K}_k\cdot\mathbf{C}\cdot\mathbf{P}_{k/k-1}\end{aligned} \tag{34}$$

The delta function Δ modifies the expression of the Kalman gain indicating the presence (unit Δ matrix) or the absence (zero Δ matrix) of measurements in one particular estimation instant. Predicted future positions are derived imposing zero Δ matrices for future instants, as measurements are not available.

Regarding to the robot, the sequence of predicted positions is considered as the Start Points, S_i, of the reference trajectory for the Bézier Shape Deformation method explained above. Regarding to the obstacles, their future trajectories and associated uncertainties are considering as restricted areas for path planning.

These predicted trajectories are used in the generation of the potential field $U(\mathbf{q},i)=U_{att}(\mathbf{q},i)+U_{rep}(\mathbf{q},i)$, defined in Mora et. al [40] even in the instants of time without measurements of the environment $(i>0)$.

The potential field generates a force in every prediction instant i, $\mathbf{F}(\mathbf{q},i)=-\nabla U(\mathbf{q},i)$, that modifies the initial prediction depending on the location of the goal and the future trajectories of the obstacles. In fact, these set of forces are transformed into displacements taking into account a particle dynamic model. These displacements affect the shape of the initial parametric trajectory.

5.3.- COMBINING BOTH METHODS TO CREATE A FLEXIBLE AND COLLISION-FREE TRAJECTORY.

In Section 5.1, it has been defined a mathematical technique, BSD. In addition to this, a method to generate a set of forces that will guide the robot to the goal without colliding with the obstacles in the environment has been chosen. These vectors are used in BSD. The key idea is how to join both techniques to design a new procedure of path planning. It is therefore necessary to characterize the parameters required in BSD to apply it in a mobile robotics environment. These characteristics are described as follow.

In order to illustrate the required definitions, an example of prediction is used. This example corresponds to eight points along the prediction horizon with the set of vectors associated to every predicted point. The time span used in this prediction 14 seconds, that is, 2 seconds between each position point.

1. **The order of Bézier curves**. It is necessary to work with second order for mobile robots trajectories. With an order higher than second order, a Bézier curve could generate geometrical effects like loops or cusps. It is produced on the geometrical location of the control points. It means, $n_l = 2, \forall l = 1, \cdots, k$.

2. **The number of the Bézier curves**. This number must to be equal to the number of field vectors associated. The number of the Bézier curves on BSD is equal to k. The set of vectors associated to the predicted points i. Then $i = k$.
   In the proposed example, described below, the number of the predicted locations of the robot is $i = 8$, so the Bézier curves number is $k = 8$.

3. **The computation of the control points**. The control points are equally distributed along the predicted trajectory. In this study, the tested model is holonomic. Thus, Then, the predicted positions are located on a straight line and can be formulated as:

First curve	Second curve
$\mathbf{P}_0^{(1)} = \hat{x}(1)$ $\mathbf{P}_1^{(1)} = \hat{x}(1) + \frac{1}{3} \cdot \left(\hat{x}(2) - \hat{x}(1) \right)$ $\mathbf{P}_2^{(1)} = \hat{x}(1) + \frac{2}{3} \cdot \left(\hat{x}(2) - \hat{x}(1) \right)$	$\mathbf{P}_0^{(2)} = \mathbf{P}_2^{(1)}$ $\mathbf{P}_2^{(2)} = \hat{x}(2) + \frac{1}{2} \cdot \left(\hat{x}(3) - \hat{x}(2) \right)$ $\mathbf{P}_1^{(2)} = \mathbf{P}_0^{(2)} + \frac{1}{2} \cdot \left(\mathbf{P}_2^{(2)} - \mathbf{P}_0^{(2)} \right)$
The $j-$curve such that $3 \leq j \leq k-2$	
$\mathbf{P}_0^{(j)} = \hat{x}(j-1) + \frac{1}{2} \cdot \left(\hat{x}(j) - \hat{x}(j-1) \right)$ $\mathbf{P}_1^{(j)} = \mathbf{P}_0^{(j)}$ $\mathbf{P}_2^{(j)} = \hat{x}(j)$	
The last but one curve	The last one curve
$\mathbf{P}_0^{(k-1)} = \hat{x}(k-2) + \frac{1}{2} \cdot \left(\hat{x}(k-1) - \hat{x}(k-2) \right)$ $\mathbf{P}_2^{(k-1)} = \hat{x}(k-1) + \frac{1}{3} \cdot \left(\hat{x}(k) - \hat{x}(k-1) \right)$ $\mathbf{P}_1^{(k-1)} = \mathbf{P}_0^{(k-1)} + \frac{1}{2} \cdot \left(\mathbf{P}_2^{(k-1)} - \mathbf{P}_0^{(k-1)} \right)$	$\mathbf{P}_0^{(k)} = \mathbf{P}_2^{(k-1)}$ $\mathbf{P}_1^{(k)} = \hat{x}(k-1) + \frac{2}{3} \cdot \left(\hat{x}(k) - \hat{x}(k-1) \right)$ $\mathbf{P}_2^{(k)} = \hat{x}(k)$

Where:

$\hat{x}$ is the predicted trajectory. It is a vector and its dimension is the same as the number of vectors to deform the curve.
$\mathbf{P}_i^{(j)}$ is the $i-$control point of the $j-$curve.

4. **The location of each vector in each Bézier curve.** As the control points are equally distributed, each vector is located in the middle of each Bezier curve, except the first one that is located at the initial point and the last one that is located at the end point.

   In Figure 3, the straight line is the predicted trajectory of the robot. In this line, the necessary control points are plotted with a red circle. The field of vectors are plotted on the positions of the predicted trajectory.

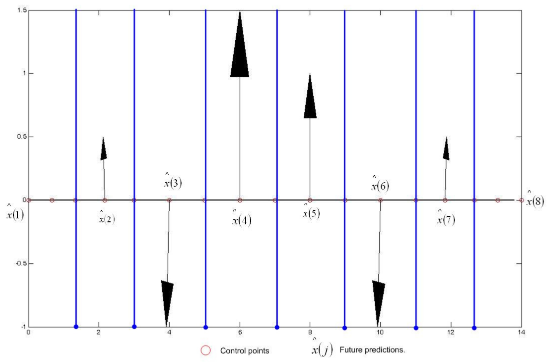

Figure 3
Control points and the Future predictions

5. **Translating the concatenated Bézier curves into a trajectory.** To transform the Bézier curve in a trajectory, that is, take into account the time at which the robot must be achieving each position. First of all, it is necessary to change the intrinsic parameter. The computation is explained in (3) and (4).

Using this notation, the time of the first Bezier curve is, $u_0 = 0$ sec, $u_f = 1.33$ sec In the second curve, $u_0 = 1.33$ sec, $u_f = 3$ sec. These data are described in the following
Table 1

CURVE	INICIAL TIME (seconds)	FINAL TIME (seconds)
First curve $k=1$	$u_0 = 0$	$u_f = 1.33$
Second curve $k=2$	$u_0 = 1.33$	$u_f = 3$
Third curve $k=3$	$u_0 = 3$	$u_f = 5$
Fourth curve $k=4$	$u_0 = 5$	$u_f = 7$
Fifth curve $k=5$	$u_0 = 7$	$u_f = 9$
Sixth curve $k=6$	$u_0 = 9$	$u_f = 11$
Seventh curve $k=7$	$u_0 = 11$	$u_f = 12.66$
Eight curve $k=8$	$u_0 = 12.66$	$u_f = 14$

Table 1
The initial and final time of every curve

6. **Resampling the deformed trajectory.** Using the hereinbefore definition, the concatenated Bézier curves are translated into a continuous trajectory. However, the robot and in particular the control loop needs discrete positions in each time step. In order to improve the resampling process a polyline is generated. The X axis of the polyline represents the cumulative time and in the Y axis is represented the cumulative intrinsic parameter. Using a polyline, the resampling process is quite easy. Taking values on the X axis homogeneously distributed (with a resampling period *T*) computing its image through this polyline a real number is obtained. The integer part of this real number corresponds to the number of the curve. The fractional part of this real number belongs to the intrinsic parameter of this curve. The polyline represented in Figure 4 is the polyline that corresponds to the example described below in Figure 5.

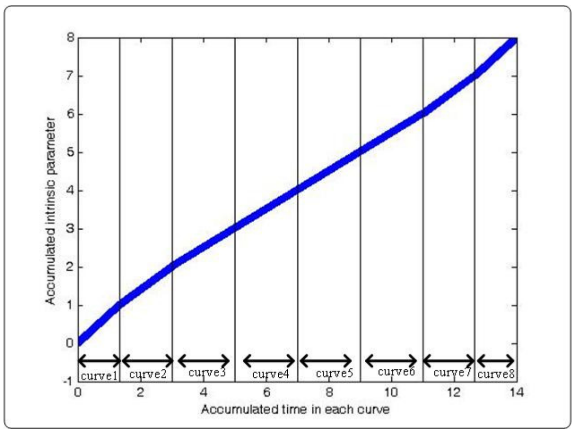

Figure 4
The resampling

The representation of the resampling Bézier curve is plotted in Figure 5. The blue cross represents the resampling of the intrinsic parameter in the predicted horizon. The blue square on the deformed Bézier curve is the resampling on the deformed Bézier. The red curve is

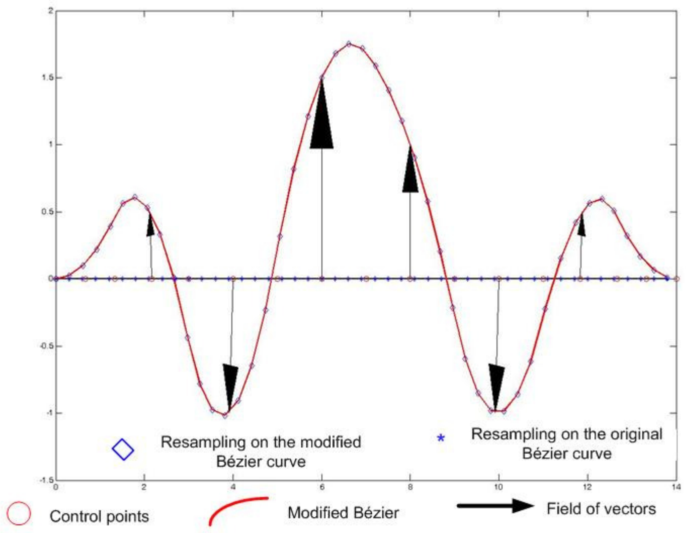

Figure 5
Deformed 8 Bézier curves concatenated

6.- SIMULATION RESULTS.

A simulation application has been implemented in Matlab to demonstrate the statements above. Figure 6 represents a 2D four-sided scenario with five mobile obstacles and a mobile robot. Initially, the robot and the obstacles follow linear trajectories (given by the kinematic model) going from side to side of the environment. When the obstacles come close to the robot, it starts a smooth avoiding maneuver, based on the Bezier Shape Deformation method, which modifies its initial trajectory and guides the robot to the goal without collision.

In Figure 6a) the robot is following a straight line trajectory when a future collision is detected. The initial trajectory is modified in real-time using the BSD method and the resulting trajectory successfully performs an avoiding maneuver. Similarly, the robot avoids another obstacle in Figure 6b) and in Figure 6c) and finally reaches the goal 1 in Figure 6d). Finally, Figure 6e) and Figure 6f) show the robot heading to a new goal (2). In this latter case, the robot is able to follow a quasi-straight trajectory.

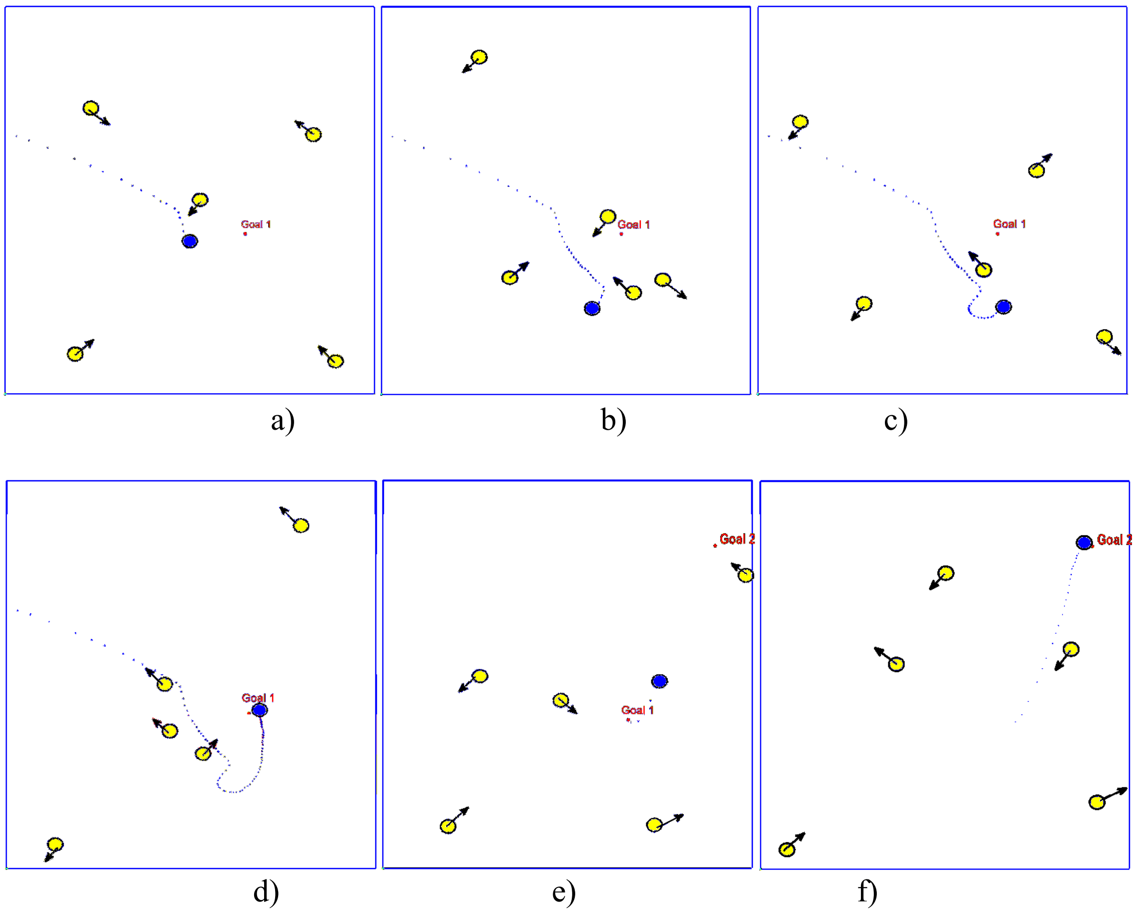

Figure 6
Simulation application - A robot heads to two consecutive goals avoiding the mobile obstacles using the BSD method.

7.- CONCLUSIONS AND FUTURE WORKS.

This paper presents a new technique, called Bézier Shape Deformation (BSD), for flexible path planning based on the deformation of a Bézier curve through a set of vectors. These parametric curves represent in a proper way an optimized path satisfying a set of constraints at the same time. The main advantage of this method is the solution obtained; it is an accurate solution of a linear system and the computational cost is low. It computes a new trajectory in real-time, avoiding the obstacles in the environment. The goal and the obstacles generate forces, these vectors are introduced in the BSD algorithm. The method used to solve it is the Lagrange theorem. The simulation results show the successful resulting trajectory.

The problem definition can include as many constraints as necessary. The matrix is defined with blocks, and it implies only to change or to include a new block in the matrix. It is important to work with other constrains like the curvature for non-holonomic cases because in a vehicle robot there is a maximum curvature that the vehicle can follow. Another constraint could be the path length because it affects the total travel time. Both functions could be included in the definition problem as a cost function or as a constraint.

REFERENCES.

[1] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design.* Morgan Kaufmann Publishers, Fifth Edition, 2002.

[2] Paluszny-Prautzch-Boehm, *Métodos de Bézier y B-Splines*. Springer Verlag Berlin Heidelberg 2002.

[3] David Salomon, *Curves and Surfaces for Computer Graphics,* Springer.

[4] Duncan Marsh, *Applied Geometry for Computer Graphics and CAD.* Springer, Second Edition.

[5] Piegl, L., *Modifying of the shape of rational B-Spline part 1: Curves.* Computer Aided Design, 21(8). Pp 509-518.

[6] Opfer, G.Oberle, H.J., *The derivation of cubic splines with obstacles by methods of optimization and optimal control*, Numer. Math.,52:17-31,1988.

[7] Qingbiao Wu, Shuyi Tao, *Shape modification of B-Spline curves via constrained optimization for multitarget points*, World Congress on Computer Science and Information Engineering, 2009.

[8] L.Piegl, *On NURBS: A Survey.*, IEEE Computer Graphics and Applications, 11(1):55-71, January 1991.

[9] Au, C.K., Yuen, M.M.F., *Unified approach to NURBS curve shape modification.* Computer Aided Design, Vol. 27(2).

[10] R.J.Sánchez, *A simple technique for NURBS shape modification,* IEEE Computer Graphics and Applications, Vol. 17(1), pp. 52-59. 1997.

[11] S.M. Hu, D.W. Zhou, J. G. Sun, *Shape modification of NURBS Curves via Constrained Optimization*. Proceedings of the CAD/Graphics, pp. 958-962. 1999.

[12] S.M.Hu, Y.F. Li, J.T. Chen, *Modifying the shape of NURBS surfaces with geometric constraints,* ComputerAided Design, Vol. 33(2), pp. 903-912. 2001.

[13] D.S.Meek, B.H.Ong, D.J.Walton, *Constrained interpolation with rational cubics*, Computer Aided Geometric Design. Vol.20, pp. 253-275, 2003.

[14] Fowler, B., Bartels, R.,*Constrained-based curve manipulation*, IEEE Computer Graphics and Application, Vol. 13(5); pp. 43-49. 1993.

[15] L.Xu, Y.J. Chen, N. Hu, *Shape modification of Bézier curves by constrained optimization*, Journal of software (China), Vol. 13(6), pp.1069-1074. 2002.

[16] O.B.Wu, F.H.Xia, *Shape modification of Bézier curves by constrained optimization*, Journal of Zhejiang University-Science A, Springer-Verlag GmbH, pp. 124-127, 2005.

[17] N. Montés, F.Sánchez, L.Hilario, A.Falcó, *Flow Numerical Computation through Bézier Shape Deformation for LCM process simulation methods*, International Journal of material forming, 2008, Vol.1 pp. 919-922 Lyon (France).

[18] N. Montés, N., Herráez, A., Armesto, L., Tornero, J., 2008. Real-time clothoid approximation by Rational Bézier curves. In *ICRA 2008, IEEE Int. Conf. on Robotics and Automation*, pp.2246 – 2251.

[19] N. Tatematsu and K. Ohnishi, *Tracking motion of mobile robot for moving target using NURBS curve*, In Int'l Conf. on Industrial Technology, pages 245-249, San Francisco, CA, December 2003.

[20] Jacopo Aleotti, Stefano Caselli, *Trajectory Clustering and Stochastic Approximation for Robot Programming by Demonstration*, 2005.

[21] Jacopo Aleotti, Stefano Caselli, *Trajectory Reconstruction with NURBS Curves for Robot Programming by Demonstration*, In IEEE Int'l Symp, on Computational Intelligence in Robotics and Automation, Helsinki, Finland, June 2005.

[22] Jacopo Aleotti, Stefano Caselli, *Grasp Recognition in Virtual Reality for Robot Pregrasp Planning by Demonstration*, in IEEE Intl. Conf. on Robotics and Automation, Orlando, FL, May 2006.

[23] M. Khatib, H. Jaouni, R. Chatila, J. P. Laumond, *Dynamic Path Modification for Car-Like Nonholonomic Mobile Robots*, Proceedings of the 1997 IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, April 1997.

[24] Birgit Graf, José Manuel Hostalet Wandosell, Christoph Schaeffer, *Flexible Path Planning for Nonholonomic Mobile Robots*, in Proceedings of the Fourth European workshop on advanced mobile robots (EUROBOT'01), September 19-21, 2001, Lund, Sweden, pp. 199-206.

[25] K. Nagatani, Y.Iwai, and Y.Tanaka, *Sensor Based Navigation for car-like mobile robots using Generalized Voronoi Graph,* Int. Conf. on Intelligent Robots and Systems, pp. 1017-1022, 2001.

[26] Jung-Hoon Hwang, Ronald C.Arkin, Dong-Soo Know, *Mobile Robots at your fingertip: Bézier Curve on-line trajectory generation for supervisory control*, in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System, 2003.

[27] Igor Skrjanc, Gregor Klancar, *Cooperative Collision Avoidance between Multiple Robots Based on Bézier Curves*, ITI 2007 29th (International Conference on Information Technology Interfaces), June 25-28, 2007, Cavtat, Croatia.

[28] M. Lizarraga, G.Elklaim, *Spatially Deconflicted Path Generation for Multiple UAVs in a Bounded Airspace*, ION/IEEE Position, Location and Navigation Symposium, ION/IEEE PLANS 2008, Monterey, CA, May 5-8, 2008.

[29] J. Choi, G.Elkaim, *Bézier Curves for Trajectory Guidance*, World Congress on Engineering and Computer Science, WCECS 2008, San Francisco, CA, Oct. 22-24, 2008.

[30] J. Choi, R.Curry, G.Elkaim, *Smooth Path Generation Based on Bézier Curves for Autonomous Vehicles*, WCECS (World Congress on Engineering and Computer Science 2009 Vol II), October 20-22, 2009, San Francisco, USA.

[31] J. Choi, R. Curr, G.Elkaim, *Path Planning based on Bézier Curve for Autonomous*

Ground Vehicles, in Proceedings of the Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science 2008 (WCECS 2008).

[32] K. Komoriya and K. Tanie, *Trajectory Design and Control of a Wheel-Type Mobile Robot Using B-Spline Curve*, Int. Workshop on Intelligence Robots and Systems, pp. 398-405, 1989.

[33] B. Vazquez, G.J. Humberto Sossa A. and Juan L. Diaz de Leon S, *Auto Guided Vehicle Control Using Expanded Time B-Spline*, Int. Conf. on Systems, Man, and Cybernetics, pp. 2786-2791, 1994.

[34] J. Zhang, J.Raczkowsky and A.Herp, *Emulation of Spline Curve and Its Applications in Robot Motion Control,* IEEE conf. on Fuzzy Systems, pp. 831-836, 1994.

[35] H. Eren, C. C. Fung and J.Evans, *Implementation of the Spline Method for Mobile Robot Path Control,* Instrumentation and Measurement Technology Conference, pp. 739-744, 1999.

[36] M. Yamamoto, M. Iwamura, and A. Mohri, *Quasi Time-Optimal Motion Planning of Mobile Platforms in the Presence of Obstacles,* Int. Conf. on Robotics and Automation, pp. 2958-2963, 1999.

[37] J. Connors, G.Elkain, *Manipulating B-Spline Based Paths for Obstacle Avoidance in Autonomous Ground Vehicles*, ION National Technical Meeting, ION NTM 2007, San Diego, CA, Jan. 22-24, 2007.

[38] J. Connors, G.Elkaim, *Analysis of a Spline Based, Obstacle Avoiding Path Planning Algorithm*, IEEE Vehicle Technology Conference, IEEE VTC 2007, Dublin, Ireland, Apr. 22-25, 2007.

[39] J. Connors, G. Elkaim, *Experimental Results for Spline Based Obstacle Avoidance of an Off-Road Ground Vehicle*, ION Global Navigation Satellite Systems Conference, ION GNSS 2007, Fort Worth, TX, Sept. 25-28, 2007.

[40] Mora, M.C., Pizá, R., Tornero, J., 2007. *Multirate obstacle tracking and path planning for intelligent vehicles*. In *IEEE Intelligent Vehicles Symposium*, pp. 172-177.

[41] Mora, M.C., Tornero, J., 2007. *Planificación de movimientos mediante la propagación de campos potenciales artificiales*. In *8º Congreso Iberoamericano de Ingeniería Mecánica*, e-book.

[42] Khatib, O., 1986. *Real-time obstacle avoidance for manipulators and mobile robots*. In *The International Journal of Robotics Research*, vol. 5, nº 1, pp. 90-98.

[43] Tornero, J., Salt, J., Albertos, P., 1999. *LQ Optimal Control for Multirate Sampled Data Systems*. In *14th IFAC World Congress*, pp. 211-216.

[44] Pizá, R., 2003. *Modelado del entorno y localización de robots móviles autónomos mediante técnicas de muestreo no convencional*, PhD Thesis, Universidad Politécnica de Valencia.

ACKNOWLEDGMENTS.

This research work is financially supported by the project GV/2010/054 of the Generalitat Valenciana.