# The use of the 3D Smoothed parametric curve Path planning for Autonomous mobile robots

O. Hachour

***Abstract*—**In this present work we present a three dimensional 3D path planning of autonomous mobile robots. The proposed method starts from an initial point to a target point establishing a control points for which connections are made to determine the form of the path without collisions. The robot moves within the unknown environment by sensing and avoiding the obstacles coming across its way towards the target. The navigation is done in 3D environment where the planar is considered as 3D smoothed cubic B-spline surface. The obtained path is the shortest path from all possible free trajectories (the smoothness of the trajectory is done around the control point). The start point and the target point must belong to the control points constructing the smoothed surface. To describe the geometric shape of the environment we have used the technique of cubic b-splines. Theses B-spline are used to represent surfaces. They combine a low degree polynomial or rational representation of maximal smoothness with a geometrically intuitive variation of the surface in terms of the coefficients: by connecting the coefficients one obtains a mesh that roughly outlines the surface. In this context, we have used Bezier surfaces which often fulfill the requirement of generating smooth geometry. The path connecting the start point to the target point must be interpolated with a spline curve to obtain a smooth curve which fits the surface perfectly. The proposed algorithm can deal with any shape obstacles even if it is the case of circular obstacles. This case is the hardest one in any navigation problem. The problem is solved by proposing some useful solutions for each situation. The robot succeeds to reach its target without collisions. The results are satisfactory to see the great number of environments treated. The results are promising for next developments and more design.

***Keywords*—**Autonomous Mobile robots**,** Navigation**,** 3D Path planning, Cubic B-splines.

## I. Introduction

A key prerequisite for a truly autonomous robot is that it can navigate safely within its environment . The problem of achieving this is one of the most active areas in mobile robotics research, which is stated as finding the answers to the three questions ''where am I?'', ''where do I go?'', and ''how do I get there?''. For an autonomous mobile robot these questions refer to the tasks of self-localization, map building, and path planning. In this paper three issues are addressed and new scientific results provided. The difficulty of this problem depends on the characteristics of the robot's environment, the characteristics of its sensors, and the map representation required by the application at the same time, the size of the environment may also affect the implementation method.

The theory and practice of Intelligent Autonomous Robot are currently among the most intensively studied and promising areas in computer science and engineering which will certainly play a primary goal role in future. These theories and applications provide a source linking all fields in which intelligent control plays a dominant role. Cognition, perception, action, and learning are essential components of such-systems and their use is tending extensively towards challenging applications (service robots, micro-robots, bio-robots, guard robots, warehousing robots).

The autonomous robot navigation problem has been studied thoroughly by the robotics research community over the last years. Contemporary methods for robot navigation. the basic feature of an autonomous mobile robot is its capability to operate independently in unknown or partially known environments. The autonomy implies that the robot is capable of reacting to static obstacles and unpredictable dynamic events that may impede the successful execution of a task . To achieve this level of robustness, methods need to be developed to provide solutions to localization, map building, planning and control. The development of such techniques for autonomous robot navigation is one of the major trends in current robotics research [4].

The robot has to find a collision-free trajectory between the starting configuration and the goal configuration in a static or dynamic environment containing some obstacles. To this end, the robot needs the capability to build a map of the environment, which is essentially a repetitive process of moving to a new position, sensing the environment, updating the map, and planning subsequent motion. Most of the difficulties in this process originate in the nature of the real world: unstructured environments and inherent large uncertainties. First, any prior knowledge about the environment is, in general, incomplete, uncertain, and approximate. For example, maps typically omit some details and temporary features; also, spatial relations between objects may have changed since the map was built. Second, perceptually acquired information is usually unreliable. Third, a real-world environment typically has complex and unpredictable dynamics: objects can move, other agents can modify the environment, and apparently stable features may change with time. Finally, the effects of control actions are not completely reliable, e.g. the wheels of a mobile robot may slip, resulting in accumulated zoometric errors.

Robot navigation can be defined as the combination of three basic activities:

• Map building: this is the process of constructing a map from sensor readings taken at different robot locations. The correct treatment of sensor data and the reliable localization of the robot are fundamental in the map-building process.

• Localization: this is the process of getting the actual robot's location from sensor readings and the most recent map. An accurate map and reliable sensors are crucial to achieving good localization.

• Path planning : This is the process of generating a feasible and safe trajectory from the current robot location to a goal based on the current map. In this case, it is also very important to have an accurate map and a reliable localization .

Recent research on intelligent autonomous robot has pointed out a promising direction for future research in mobile robotics where real-time, autonomy and intelligence have received considerably more weight then, for instance, optimality and completeness. Many navigation approaches have dropped the explicit knowledge representation for an implicit one based on acquisitions of intelligent behaviours that enable the robot to interact effectively with its environment, they have to orient themselves, explore their environments autonomously, recover from failure, and perform whole families of tasks in real-time [2, 3].

To perform all tasks in different environments, the vehicle must be characterized by more sever limits regarding mass volume, power consumption, autonomous reactions capabilities and design complexity. Particularly, for planetary operations sever constraints arise from available energy and data transmission capacities, e.g., the vehicles are usually designed as autonomous units with: data transfer via radio modems to rely stations ( satellite in orbit or fixed surface stations) and power from solar arrays, batteries or radio-isotope thermo electric generators (for larger vehicles). A common application of mobile robot is the object manipulation. Examples include pick and place operation on the factory floor, package sorting and distribution.

However, for navigation in dynamic environments or high-speeds, it is often desirable to provide a sensor-based collision avoidance scheme; it would be difficult for the (remote) operator to prevent the robot from colliding with obstacles. This is primarily due to: limited information from the robot's sensors, such as images within a restricted viewing angle without depth information, which is insufficient for the users full perception of the environment in which the robot moves, and significant delay in the communication channel between the operator and the robot.

The implementation of a collision avoidance scheme on-board the robot can cause conflict between the users actions and the movement of the robot. For example, consider a situation where the operator directly controls the movement of a mobile robot with a joystick and the robot is supposed to move forward when the user pushes the stick forward. Imagine that he robot is also programmed with a simple collision avoidance algorithm to avoid obstacle. If an obstacle exists in front of the robot, the robot may stop or turn in order to avoid collision, although he operator is clearly commanding it to, move ahead. In this example, the conflict may be not a problem if the user can easily see the obstacles. If however, the obstacles are invisible due a restricted viewing angle, the user might be confused since the robot does not move nor act according to the teleoperation commands. We hypothesize that the conflict can be naturally resolved by exploiting hap tic information, that is, by providing the operator with force feedback. Force-feedback has been used for precise remote control in teleoperation of manipulator.

Navigation is the science (or art) of directing the course of a mobile robot as the robot traverses the environment. Inherent in any navigation scheme is the desire to reach a destination without getting lost or crashing into any objects. The goal of the navigation system of mobile robots is to move the robot to a named place in a known, unknown, or partially known environment.

The goal of the navigation process of mobile robots is to move the robot to a named place in a known, unknown or partially known environment. In most practical situations, the mobile robot can not take the most direct path from the start to the goal point. So , path planning techniques must be used in this situation, and the simplified kinds of planning mission involve going from the start point to the goal point while minimizing some cost such as time spent, chance of detection, or fuel consumption.

Several approaches for path planning exist for mobile robots, whose suitability depends on a particular problem in an application. For example, behavior-based reactive methods are good choice for robust collision avoidance. Path planning in spatial representation often requires the integration of several approaches. This can provide efficient, accurate, and consistent navigation of a mobile robot. . It is sufficient for the robot to use a topological map that represents only the areas of navigation (free areas, occupied areas of obstacles). It is essential that the robot has the ability to build and uses models of its environment that enable it to understand the environment's structure. This is necessary to understand orders, plan and execute paths.

The major task for path-planning for single mobile robot is to search a collision –free path. The work in path planning has led into issues of map representation for a real world. Therefore, this problem considered as one of challenges in the field of mobile robots because of its direct effect for having a simple and computationally efficient path planning strategy. For path planning areas, it is sufficient for the robot to use a topological map that represents only the different areas without details such as office rooms. The possibility to use topological maps with different abstraction levels helps to save processing time. The static aspect of topological maps enables rather the creation of paths without information that is relevant at runtime. The created schedule, which is based on a topological map, holds nothing about objects which occupy the path. In that case it is not possible to perform the schedule [7,9]. To get further actual information, the schedule should be enriched by the use of more up-to date plans like egocentric maps.

Systems that control the navigation of a mobile robot are based on several paradigms. Biologically motivated applications, for example, adopt the assumed behavior of animals. Geometric representations use geometrical elements like rectangles, polygons, and cylinders for the modeling of an environment. Also, systems for mobile robot exist that do not use a representation of their environment. The behavior of the robot is determined by the sensor data actually taken. Further approaches were introduced which use icons to represent the environment. One of the specific characteristics of mobile robots is the complexity of their environment, therefore, one of the critical problem for the mobile robots is path planning.

To perform all tasks in different environments, the robot must be characterized by more sever limits regarding mass volume, power consumption, autonomous reactions capabilities and design complexity. Particularly, for planetary operations sever constraints arise from available energy and data transmission capacities, e.g., the vehicles are usually designed as autonomous units with: data transfer via radio modems to rely stations ( satellite in orbit or fixed surface stations) and power from solar arrays, batteries or radio-isotope thermo electric generators (for larger vehicles). A common application of mobile robot is the object manipulation. Examples include pick and place operation on the factory floor, package sorting and distribution. Some researchers are interesting in the simplest kind of object manipulation i.e. pushing. Pushing is the problem of changing the pose of an object by imparting a point contact force to it. For the simplicity, they constrain their self to the problem of changing the pose (in a horizontal plane).

The environment force prevents the robot from moving and turning towards obstacles by giving the user the distance information between the robot and the obstacle in a form of force. This force is similar to the traditional potential force field for path planning of mobile robot. However, the environment force is different from the potential force in some aspects. First there is no attention to a goal since we assume that the goal position is unknown. Secondly, only obstacles in the “relevant” area (according to the logical position of the interface) are consider, i.e. the obstacles that are far, or in the direction opposite to the movement of the robot are not relevant. In this context, a full range of advanced interfaces for vehicle control has been investigated by the researchers. These works demonstrates that obstacle detection and collision avoidance is improved with good results.

In this paper we deal with 3D path planning; we first supposed that the robot navigates in planar environment. In this case, the robot can avoid any obstacles shape even in congested environment. However, for the case of non-polygonal obstacles it is better to be surrounded by poly-solid objects, otherwise the algorithm will spend much time to return a result. And some complex cases the algorithm cannot achieve the target even a free path may exist. The planar is considered as 3D smoothed cubic B-spline surface.

In our previous work [1, 10, 11, 12, 13] concerned with a path planning problem of an autonomous robot operating in a 2-dimensional surface with obstacles. The robot was considered as a material point. In this paper we deal with 3-dimentional navigation. In the first part the obstacles are presented in 3-dimensions and the planar path connecting the start to the target point take into consideration the principal robot dimensions. In the second part, we treat the 3D Path Planning in natural cluttered Environment. In the last part, we discuss the typical sensors that can help us to implement our. This paper proposes a new methodology of work in 3D navigation where the problem of movement of an autonomous mobile robot is solved using Mesh creation and 3D surface smoothing.

## II. 2D Planar Path Planning

In this part we consider the planar navigation, we use the same algorithm used in case of 2-dimenssion navigation in our previous work [10,11,13] , where we have to take here into consideration 3D obstacles shape and the real robot dimensions. Furthermore, we use the NURB-spline for the smooth of the trajectory. In case of 3D the problem can be reduced to a problem of two dimensions by projecting the objects on the plan containing the initial point, the target point and the control point.

The use of the parametric curves for path planning was introduced by some researchers where a trajectory without collision is regarded as to be a series of curves connecting the initial points to the target in the workspace . The control points for which connection were made determines the form of the trajectory. For this case the problem of optimal trajectory calculation is posed.

The use of parametric curves for the path planning is very developed and largely used in computer graphics, in design and manufacture computer-assisted. A parametric curve has an inherent directional property which enormously reduces calculations. In this work, a linear parametric curve is used for path planning; the smooth of the trajectory around the control point is also taken in consideration.

The linear trajectory connecting the initial to the goal points is examined in the first step. If the trajectory collide an obstacle, a point of control is introduced between the initial and the goal point and a point of intermediate connection is created once again. The point of control can be structured in a coordinated way, and a checking of interference between the trajectory and the obstacles produces a map. For the problem in two dimensions, two parameters are necessary to define a control point, and four parameters are necessary to define two points of control. From where a certain number of control points define the dimension of the control point space CPS. An interference checking produces images in the CPS and this tracing is defined by a geometrical layout .

The control point space has a property in the Free Space (FS) which defines a trajectory without collision. Free space is a surface belonging to the CPS, but unoccupied by the image of the obstacles. The geometrical layout is a computing process. The construction of the CPS depends on the manner in which the control points are defined.

### *A. Collision detection and obstacles avoidance*

The linear parametric curve connecting the starting point, $S$, to the target point, $T$, is given by The directional property of the parametric curve enables us to express the intersection of two segments of a line in simple terms. To determine if the line $ST$ collides an object we must carry out a checking of the intersection of the line $ST$ with the polygonal contours of the obstacles. The line $ST$ has the tendency to intersect with the contour of the obstacle Obs2 in two points (1) and (2), such that $s_1 \le s \le s_2$. The calculation of the intersection between the line segment $ST$ and the obstacle contour $P_iP_j$ can be expressed as follows:

$$(T-S)\times s + S \ = \ (P_j - P_i)\times t + P_i$$

$t$: represents the curve parameter; In the case of 2D the vectorial equation is composed of two equations with two unknowns ($s$ and $t$ parameters). The parameter $s$ and $t$ can be solved by eliminating the others.

$$s = \frac{(P_{ix} - S_x)(P_{jy} - P_{iy}) - (P_{iy} - S_y)(P_{jx} - P_{ix})}{(T_x - S_x)(P_{jy} - P_{iy}) - (T_y - S_y)(P_{jx} - P_{ix})}$$

$$t = \frac{(T_x - S_x)(S_y - P_{iy}) - (T_y - S_y)(S_x - P_{ix})}{(T_x - S_x)(P_{jy} - P_{iy}) - (T_y - S_y)(P_{jx} - P_{ix})}$$

In the workspace, we can define the position of the obstacles with respect to the line $ST$, and there exist three cases: the obstacles which cut the line $ST$, which are above the line $ST$ and which are below the line $ST$ .for the case of 3D we have to take the width of the robot to consideration as it is shown in figure 1.

The path in 3D environment is considered as an object which has the width and the height of the robot. Our algorithm creates three selection sets: the first contains the virtual path, the second contains all the obstacles existing in the environment and the third one is initially empty; then the algorithm checks the interference between solids in the first selection set against those in the second selection set.

Once the algorithm starts checking for interferences, temporary interference objects are created and included to last created selection set (see figure 2). In order to avoid the obstacle, the virtual path must be rotated around the starting point, $S$, by a small angle $\Delta\theta$ given by the equation 1

$$\Delta\theta = \sin^{-1}\frac{w}{ST} \qquad (1)$$

Where: $ST$ represents the distance between the starting point and the target, and $w$ is expected robot width. After this rotation, the temporary interference objects that are created during interference checking are deleted (see figure .3). Thus the interference selection set becomes empty. The algorithm rotates again by $\Delta\theta$ and checks for collision by testing the interference selection set, the linear path is in collision with obstacles unless the selection set is not empty. It keeps rotating until the interference selection becomes empty (see figure 4). When the selection set is not empty, the created interference objects must be removed before the next iteration. The iteration is limited for obstacles which interfere with the linear curve

### *B. Control point insertion*

When the linear path collide an obstacle, a control point must be inserted between the starting point and the obstacle. The control point $Q$, belong to the characteristic curve network describing the trajectory without collision projected in 2D workspace. In the figure 5, the polar coordinates are used to define a control point with the parameters $\theta$ and $\rho$. The arbitrary value of $\theta$ and $\rho$ is fixed between 0°< $\theta$ < 360° ($\theta \neq$ 0, 180°) and 0 < $\rho \le 1$. To obtain the control point $Q$ all the obstacles must be projected into 2D workspace, the distance '$d$' between C and the extreme limit of AWS is expressed by :

$$d \ = \ \sqrt{(W_x - C_x)^2 + (W_y - C_y)^2}$$

As the center C of CWS is in the middle of $ST$ : The values of $Q_{\max}$ correspondent to the maximum values of $D$ are given by :

$$\begin{cases} Q_{\max x} = \dfrac{1}{D}\{(T_x - S_x)(d_{\max}\cos\theta + C_x) - (T_y - S_y)(d_{\max}\sin\theta + C_y)\} + \dfrac{w}{2} \\ Q_{\max y} = \dfrac{1}{D}\{(T_y - S_y)(d_{\max}\cos\theta + C_x) - (T_x - S_x)(d_{\max}\sin\theta + C_y)\} + \dfrac{w}{2} \end{cases}$$

(2)

Where $D$ represents always the distance $ST$, i.e. $D = |T - S|$. W represents the width of the robot .We have added $\frac{w}{2}$ because for navigation we conceder the center of gravity of the robot. For the case of non-polygonal obstacles (such as cylinders, cones etc.) must be circumscribed by poly-solids with the maximum edges, as shown in the figure 5 (Obs4).

The checking of interference with the obstacles must be carried out before the tracing of $QT$.The checking for interference must be carried out for the obstacles above and below the line $ST$, hens the rotation is done to the left and the right. As each obstacle produced well defined $\rho$ ranges, the union of these ranges gives us as a result a well defined $\rho$ ranges for a given value of $\theta$. The checking of interference must be made for all the obstacles according to the above procedure. The Figures 6 and 7 show a typical case of geometrical planning. The shading surface on figure 6 indicates that all control points belonging to this surface intersect with the line segment $SQ$ or $QT$. The surface defined by $A_1$, $A_2$, $A_3$ and $A_4$ of CPS in the figure 7 shows the trajectories without collision. The obstacles in the Euclidian space are represented by a complex form that show only the surface for the control point $Q$. The obstacles are represented as 3D objects, where the path is drawn in the plan, but the height of the robot is also taken to count. Consequently, it may exist a free path throw objects (for example, in case of bridge, the feasible optimal path may be under that bridge). As the process end up when a feasible path is found, the construction of the whole map is not necessary, unless there are other constraints which must be taken on consideration. According to figure 7, it is clear that the trajectory taken from $A_3$ and $A_4$ cross behind $Ob1$ and $Ob3$ respectively. The circle $C_1$ and $C_2$ are the inscribed circle with an important diameter to be more suitable for surfaces $A_3$ and $A_4$ respectively, and $C_1$ is largest than $C_2$. The path gotten from $C_1$ is far from obstacles comparing to any other path can be regarded as the most certain trajectory. It is one of the useful properties of this geometric planning.

In the workspace, we can define the position of the obstacles with respect to the line *ST*, and there exist three cases: the obstacles which cut the line *ST*, which are above the line *ST* and which are below the line *ST*. In the case where there is no intersection between the obstacle and the line *ST*, an additional calculation is necessary in order to locate its position with respect to *ST*. This does not require a precise value but a relative value with the aim of making a comparison. A method which does not use the trigometrical functions consists in calculating the determinant of the equation :

$$A = \begin{vmatrix} S_x & T_x & P_{ix} \\ S_y & T_y & P_{iy} \\ 1 & 1 & 1 \end{vmatrix}$$

This equation gives the double of the area of the triangle *STP₁* (see the figure 4), where the order (orientation) determines of the points constituting the vertices of the obstacle. The equation (6) indicates if a vertex $P_1$ of an obstacle is on top, below or on the line *ST*. Three cases arise:

1. $|STP_1|>0$, in this case S,T and $P_1$ are ordered in the clockwise direction, i.e. the obstacle is bellow the line ST.
2. $|STP_1|<0$¶, in this case S,T and $P_1$ are ordered in the counter clockwise direction, i.e. the obstacle is above the line ST.
3. $|STP_1|=0$¶, in this case S,T and $P_1$ are in the same line.

In order to reduce calculations this checking is limited to the cases where obstacles are in collision with the line *ST*.

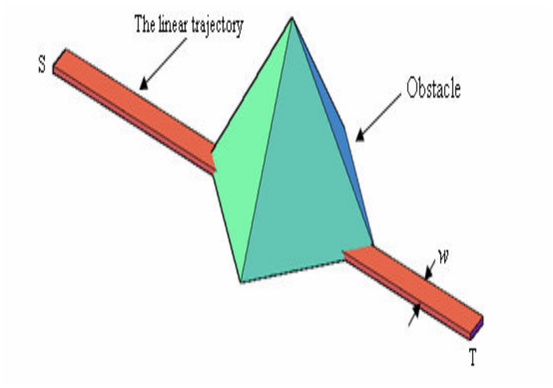


Fig. 1 The interference between the linear trajectory and the obstacle in 3D environment (isometric view)

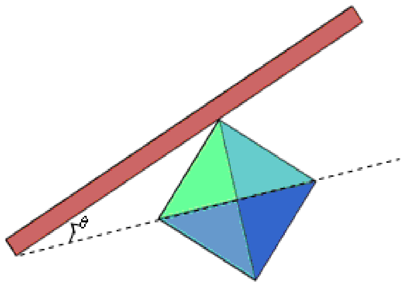


Fig. 4 the obstacle avoidance (Top view)

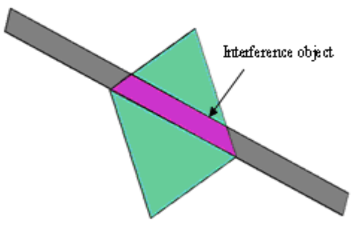


Fig. 2 the temporary interference object creation (bottom view)

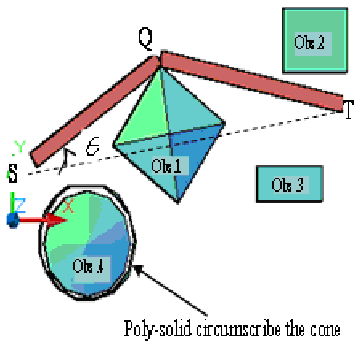


Fig. 5 The insertion of the control point (top view).

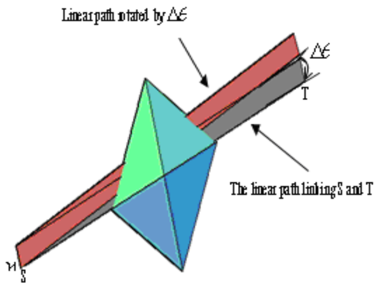


Fig. 3 Linear path rotation (Top view).

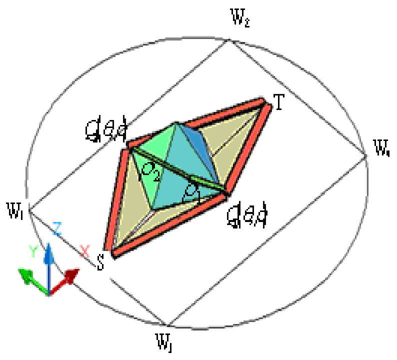


Fig.6 The interference computation to define the range of $\rho$ (Isometric view).

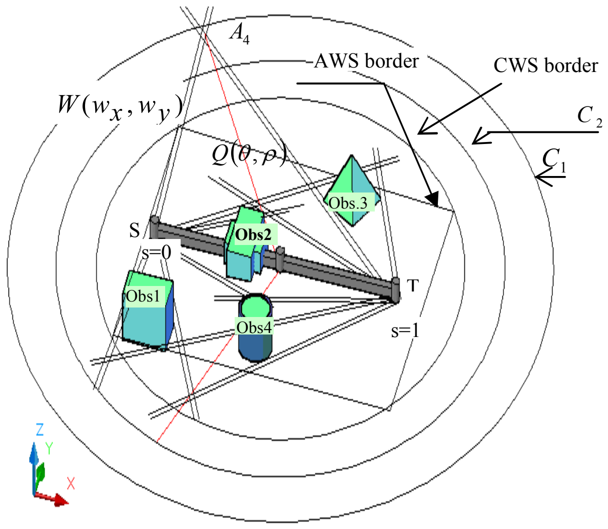


Fig.7 Euclidian space of the obstacles

*C. Connection smoothing using NURBs*

Previously we have got a path in form of segments which are discontinuous in the control point. To remedy this problem we used in the third chapter the quadratic and cubic parametric curves which generate Bezier curves .

In our algorithm, initially we add only one control point; then we test for collision between $QT$ and the obstacle. In case of collision, we have to insert another control point, and so forth. Before, we had mentioned that the Bezier curves do not ensure the local control. To overcome this drawback we use the NURBs, which are characterised by the following points:

- A NURBS curve produces a smooth curve between control points.
- We create splines by specifying coordinate points which are the starting point, the control points and the end point.
- We can change the spline-fitting tolerance. Fit tolerance refers to how closely the spline fits the set of fit points we specify.
- The lower the tolerance, the more closely the spline fits the points.
- At zero tolerance, the spline passes through the points.
- We can add a fit point (other control point) and refits the spline through the new set of points.

Here are a few things to be respected.

- The order of the curve must be equal to or higher than 2. Order 2 gives us a polyline effect.
- The control points are represented in homogeneous form, meaning that we have to divide the x, y, and z components by the w component to find the point's actual position in three-dimensional space.
- The w component of each control point must be positive.
- The number of control points must be equal to or greater than the order.
- The number of knots must be equal to the number of control points plus the order of the curve.
- The knots must be specified in non-decreasing order

We start with a NURB curve defined by starting point, one ontrol point and target point which are denoted by $Q_i$. The URB curve is given by the formula (3):

$$Q(t)=\sum_{i=0}^{t-1} Q_i N_{i,k}(t) \qquad (3)$$

with a knot vector $\{x_0, x_1, \ldots, x_{n+k-1}\}$. To add a new knot $x_{new}$, where $x_i < x_{new} \le x_{i+1}$. The new curve will be defined by the equation 4.

$$\hat{Q}(t)=\sum_{i=0}^{t-1} \hat{Q}_i N_{i,k}(t) \qquad (4)$$

Now we have to figure out not only where the new control point is located and where it goes in the ordered vector of control points, but also how to adjust some of the existing control points to keep the shape of the curve unchanged; this process yields the new control point vector, $\hat{Q}$. It turns out that the relationship between the old and new control points is given by the relation (5):

$$\hat{Q}_j = (1-\alpha_j)Q_{j-1} + \alpha_j Q_j \qquad (5)$$

Where $\alpha$ is defined by the equation (IV.6)

$$\alpha_j = \begin{cases} 1 & j \le i-k+1 \\ \dfrac{x_{new}-x_j}{x_{j+k-1}-x_j} & i-k+2 \le j \le i \\ 0 & j \ge i+1 \end{cases} \qquad (6)$$

In the figure 8, the black spline shows the smoothed path to avoid the first obstacle (obs.1), and in order to avoid the second obstacle (obs.2) a control point is added to this spline. The refitted spline is shown on red. It is clearly shown that the adding of a control point does not affect the global spline.

### *D. Simulation results*

The path given in the figure 8 shows the path as a 2D modelling NURB spline [5,8]. To consider the width of the robot we should create a region from a set of entities. This method will create a region out of every closed loop formed by the input array of curves. The first curve is the spline created from the control points, the second curve is a spline shifted away from the obstacles by a distance equal to the width of the virtual robot, and the two ends of the two splines are linked by lines to create a closed region.

The previous region presents our free path as 2D planar environment. To take the height of the robot into consideration we have to extrude the region by the height of the robot. This suction concerned with the simulation of our algorithm in different 3D environment.

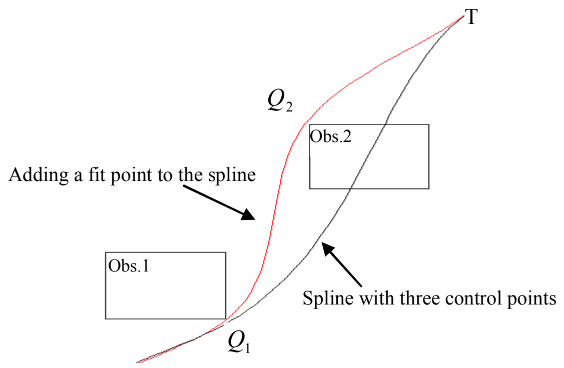
In the figure 9 the obstacle is avoided by inserting one control point. To get the smoothed path we used the slip of the first portion of the linear path as the start tangent of the NURB spline, and the slip of the second portion as end tangent of the spline. Whereas, in the case of the figure 10 the two obstacles are avoided by inserting two control points. Furthermore, the start tangent of the spline is defined by the slip of the first linear path, and the end tangent is defined by the slip of the third linear path.

To go around a corner, more than one control point must be added when necessary. The start tangent of the spline for each case is defined by slip of the first portion of the linear path; however the end tangent is defined by the slip of the last portion of the linear path. The more is the number of control points the great is the computation time. The figure 17 and 18 are an illustration example.

In the figure 10, it is clearly shown that the number of control points and the degree of the NURBs are independent, i.e. the number of control points does not affect the shape of the smoothed curve. A free smoothed path realization the insertion of one control point Smoothed path (SE Isomertic view) is shown in the figure11. The figure12 shows another example of two control points where we the main idea of work is done. The SE Isomertic view –linear trajectory-of this environment is shown in the figure13. Finally the smoothed path of this environment is shown in the figure 14. another example where a path planning in congested environment is proposed in the figure 15, we can see the movement how is done. The smoothed path of this environment is shown in the figure16.

The figure 17 and the figure 18 illustrate that the optimal free path can be by passing under a bridge. Where the figure 19 and the figure 20 is an example of navigation in complex environment, the obstacles are non-polygonal objects created by extruded spline surfaces.

Until now, although the obstacles are presented as 3D objects, we have dealt only with path planning in 2D planar environment. To deal with no planar path planning we have to start by defining the mesh, then we smooth this mesh to fit the


T
$Q_2$
Obs.2
Adding a fit point to the spline
Obs.1
Spline with three control points
$Q_1$

B-spline surface. We avoid the existing obstacles on this surface by using the previous method.

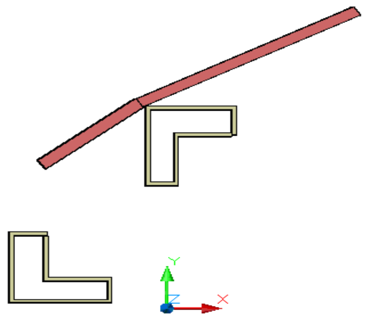


Fig.9 Free smoothed path realization the insertion of one control point-Linear path (Top view)-

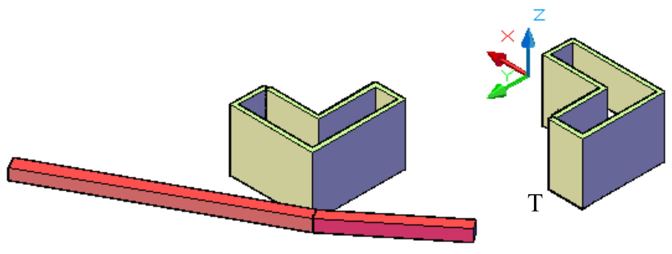


Fig.10 Free smoothed path realization the insertion of one control point Linear path (NW Isomertic view)

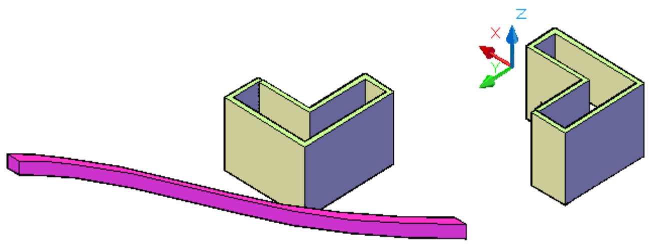


Fig.11 Free smoothed path realization the insertion of one control point
Smoothed path (SE Isomertic view)

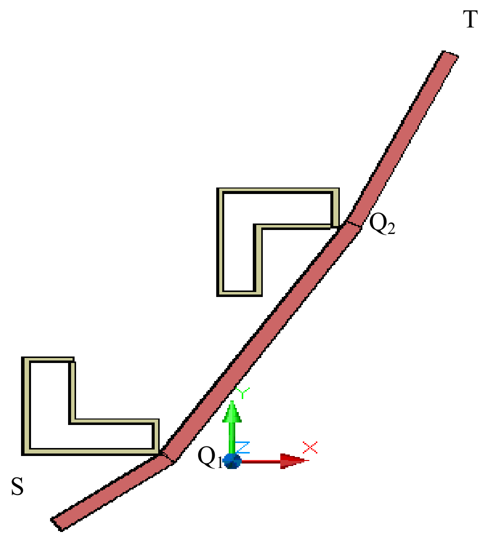


Fig . 12 Free smoothed path realization the insertion of two control
Linear trajectory (Top view)

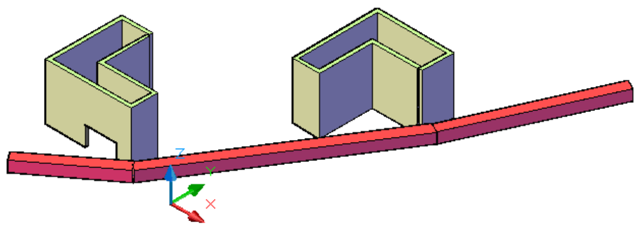


Fig. 13 Free smoothed path realization the insertion of two control
points : linear trajectory (SE Isomertic view)

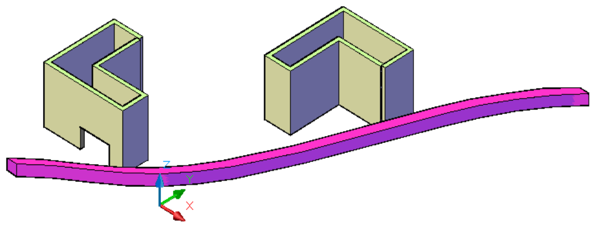


Fig. 14 Free smoothed path realization the insertion of two
control points : smoothed path (SE Isomertic view)

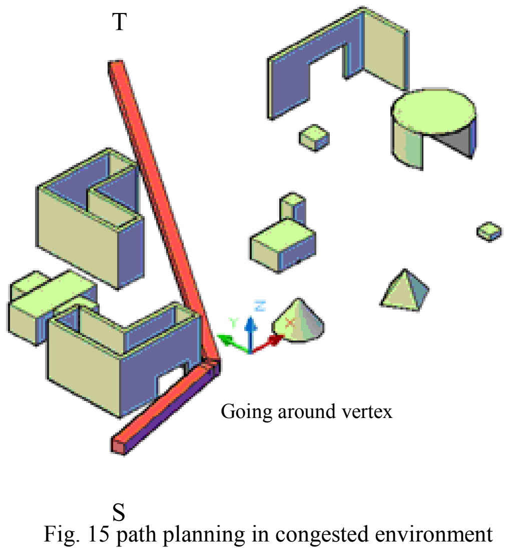


Fig. 15 path planning in congested environment
Linear trajectory (SE Isomertic view)

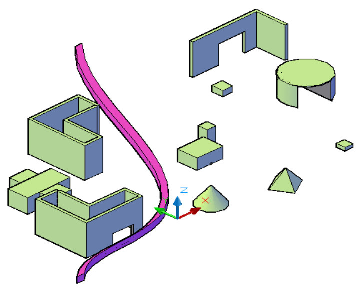
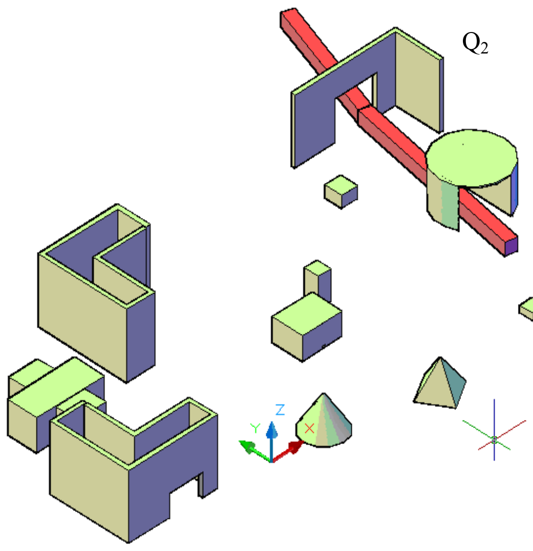
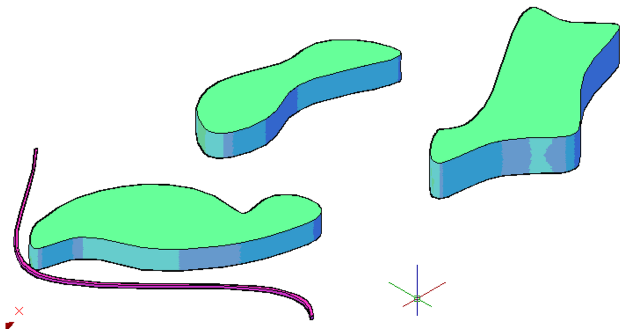


Fig. 16 path planning in congested environment
smoothed path (SE Isomertic view)

Fig. 17 path planning in complex environment
Passing behind obstacles





## III. 3D Planar Path Planning

B-spline are widely used to represent surfaces. They combine a low degree polynomial or rational representation of maximal smoothness with a geometrically intuitive variation of the surface in terms of the coefficients: by connecting the coefficients one obtains a mesh that roughly outlines the surface. Repeated refinement of this mesh by knot insertion results in a sequence of meshes whose points are averages of the preceding and whose limit is the surface itself. In addition to an elegant algebraic definition this yields an alternative geometric, procedural characterization of the splines useful for establishing many shape proprieties of spline surfaces. Each point in the interior of the B-spline mesh must be regular, that is surrounded by exactly four quadrilateral mesh cells.

A Polygon-Mesh object is an $M$ x $N$ mesh where $M$ represents the number of vertices in a row of the mesh and $N$ represents the number of vertices in a column of the mesh.

A mesh can be open or closed in either or both the $M$ and $N$ directions. A mesh that is closed in a given direction is considered to be continuous from the last row or column on to the first row or column. Vertices may be any distance from each other.

A Polygon-Mesh is always created as a simple mesh. A mesh can be smoothed after creation by using even: a quadratic B-spline surface fit, a cubic B-spline surface fit or a Bezier surface fit. The figures (21, 22, 23, 4) show the different cases of smoothing a 4x4 mesh using quadratic and cubic B-spline as well as the Bezier surface concept [6].

In our simulation the technique of cubic b-splines is used in order to describe the geometric shape of the environment. Bezier surfaces fulfill often the requirement of generating smooth geometry. In complex environments approximation based on a large number of scattered data, Bezier surfaces show the disadvantageous property of global modeling possibility. Therefore the concept of Bezier surfaces is generalized to the concept of segmented surfaces, which leads to the surface representation with b-spline technique. Furthermore, the cubic B-spline preserves the $C^2$ continuity.

There are no restrictions on the number of cells meeting at a mesh point or the number of edges to a mesh cell. Mesh cells need not be planar .The surface must not have an abrupt change in its form; consequently we have to smooth the created mesh . The start point and the target point must belong to the control points constructing the surface. The path connecting the start point to the target point must be interpolated with a spline curve to obtain a smooth curve which fits the surface perfectly. The figure 25 shows the smoothed curve fitting the surface.

The figure 25-left is an illustration example of the smoothed spline curve creation, it is shown as a 3D wireframe. Whereas the figure 25-right shows the 3D path generation with consideration of the robot width. As there are no abrupt changes on the surface, the width of the robot is considered by offsetting the first smoothed curve by a distance equal to the width of the robot. The figures 26 and 27 illustrate the use of our algorithm to avoid the obstacles existing on this surface.

The robot is supposed to be very tiny, so that it is interpreted as a point. Initially, the robot can estimate its position $S(S_x, S_y)$ but does not know its orientation $\theta$. This leads to a position sensor defined as $Y = R^2$ with

and $y_2 = S_y$. A compass (Compact Outdoor Multi

Pose Assessment System) or orientation sensor can lik

made by observing only the robot orientation. In th

$y = \theta$

The position and orientation sensors generalize nic

3D world. In this case the robot position is presented

coordinates $(x, y, z)$ which can be measured with a

sensor; whereas an orientation sensor measures th

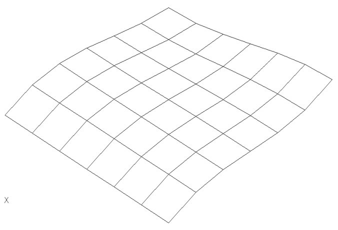
orientatio

is often called a gyroscope. These are usually based

principle of precession, which means that they co

spinning disc that is reluctant to change its orientatio

angular momentum.



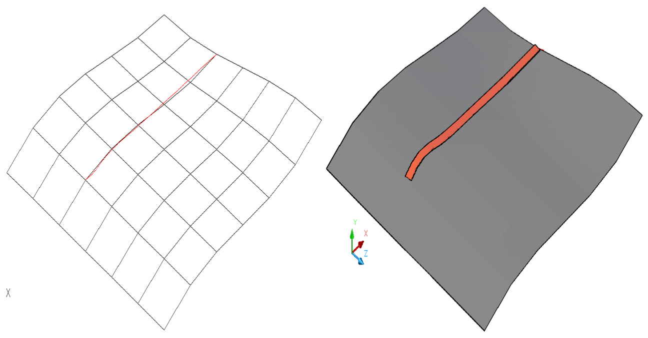


Fig. 25 the smoothed 3D path generation The smoothed spline curve

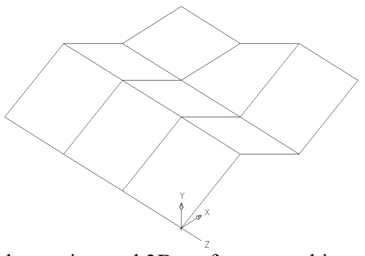


Fig. 21 Mesh creation and 3D surface smoothing: case of 4x4 simple mesh

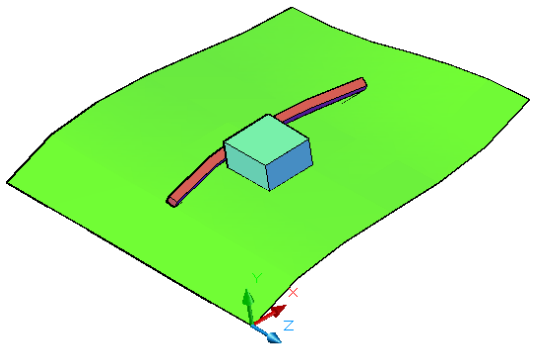


Fig. 26 3D path planning with obstacles avoidance: environment 1

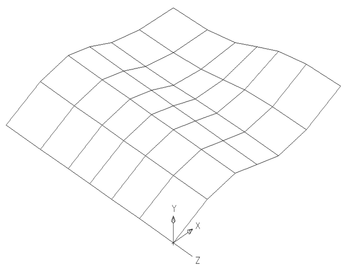


Fig. 22 Mesh creation and 3D surface smoothing: case of Quadra B-spline surface

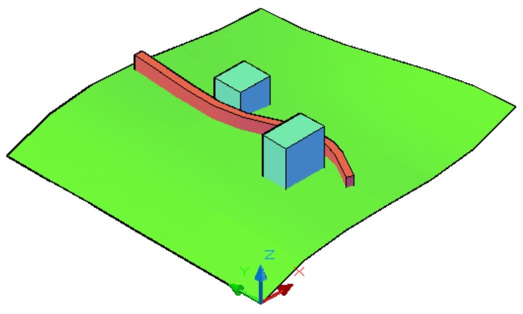


Fig. 27 3D path planning with obstacles avoidance: environment 2

## IV. Conclusion

In this paper we studied the path planning problem of an autonomous robot operating in a 3-dimentional surface with obstacles. A complete path planning algorithm guarantees that the robot can reach the target if possible, or returns a message that indicates that there is no free path when the target cannot be reached. We first supposed that the robot navigates in planar environment. In this case, the robot can avoid any obstacles shape even in congested environment. However, for the case of non-polygonal obstacles it is better to be surrounded by poly-solid objects, otherwise the algorithm will spend much time to return a result. And some complex cases the algorithm cannot achieve the target even a free path may exist**.** The robot moves within the unknown environment by sensing and avoiding the obstacles coming across its way towards the target.

The navigation is done in 3D environment where the planar is considered as 3D smoothed cubic B-spline surface. The obtained path is the shortest path from all possible free trajectories (the smoothness of the trajectory is done around the control point). In this case, the start point and the target point must belong to the control points constructing the smoothed surface. And the obstacles are avoided in the same manner as in the case of planar navigation. The proposed algorithm has the advantage of being generic and can be changed at the user demand. The obstacles can take any shape since the algorithm is general for any obstacle detection. This approach works perfectly even if an environment is unknown. We have run our simulation in several environments where the robot succeeds to reach its target in each situation and avoids the obstacles capturing the behaviour of intelligent expert system. For the main idea we propose to use simple projection sensors to measure the robot position and orientation. Our autonomous mobile robot is able to achieve these tasks: avoid obstacles, taking a decision, perception, and recognition and to attend the target which are the main factors to be realized of autonomy requirements. However in the future, it is necessary to use a robot in hostile environment and space exploration or other applications by using advanced micro-product control systems that can be dealt in 3D dimensions.

## References

[1] O. Hachour AND N. Mastorakis Behaviour of intelligent autonomous ROBOTIC IAR", *IASME transaction*, issue1, vol 1, ISSN 1790-031x WSEAS, January 2004,pp 76-86.
[2] Ihn Namgung and Joseph Duffy, "Two dimensional collision-free path planning using linear parametric curve", *Journal of Robotic Systems*, 1998.
[3] S.Florczyk, *Robot Vision Video-based Indoor Exploration with Autonomous and Mobile Robots*, WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim, 2005.
[4] O. Hachour and N. Mastorakis, IAV : A VHDL methodology for FPGA implementation, *WSEAS transaction on circuits and systems,* Issue5, Vol 3,ISSN 1109-2734, 2004,pp.1091-1096.
[5] D. F. Rogers and J. A. Adams, « Mathematical Elements for Computer Graphics », 2nd ed., *McGraw-Hill*, New York, 1990.
[6] Bezier, P., 1972, "Numerical Control-Mathematics and Application, A. R. Forrest and A. F. Pankhurst, Trans., John Wiley & Sons, New York, 1972.
[7] V. J. Lumelsky, "Effect of kinematics on motion planning for planar robots arms moving amidst unknown obstacles", *IEEE J. of Robotics and Automation*, RA-3(3), 1987,pp207-223,.
[8] I. D. Faux et M. J. Pratt, *« Computational Geometry for Design and Manufacture"*, Jhon Wiley & Sons, New York, 1979.
[9] S. M. LaValle, "*Planning Algorithms*" , Published by Cambridge University Press, 2006.
[10] O.Hachour, "The Proposed Genetic FPGA Implementation For Path Planning of Autonomous Mobile Robot", *International Journal of Circuits , Systems and Signal Processing,* Issue 2, vol2 ,2008,pp151-167.
[11] O. Hachour AND N. Mastorakis, Avoiding obstacles using FPGA –a new solution and application ,*5th WSEAS international conference on automation & information (ICAI 2004) , WSEAS transaction on systems* , issue9 ,vol 3 , Venice , Italy , , ISSN 1109-2777, November 2004, pp2827-2834.
[12] O. Hachour AND N. Mastorakis, "Behaviour of intelligent autonomous ROBOTIC IAR", *IASME transaction*, issue1, vol.1, ISSN 1790-031x WSEAS, January 2004,pp 76-86.
[13] O. Hachour, ,"path planning of Autonomous Mobile Robot", *International Journal of Systems Applications, Engineering & Development*, Issue4, vol.2, 2008, pp178-190.