SMOOTH TRAJECTORY PLANNING METHOD FOR MOBILE ROBOTS.

V. F. Muñoz† and A. Ollero ‡.
†Dpto. de Ingeniería de Sistemas y Automática. Universidad de Málaga.Plaza el Ejido s/n, 29013 Málaga (Spain).
Phone:(+34) 5 213-14-12; Fax: (+34) 5 2131413; E-mail:victor@tecma1.ctima.uma.es

‡Dpto. de Ingeniería de Sistemas y Automática.Universidad de Sevilla. Avenida Reina Mercedes s/n, 41012 Sevilla (Spain).
Phone: (+34) 5 455-68-71; Fax: (+34) 5 455 68 49; E-mail: aollero@esi.us.es

ABSTRACT.

This paper presents a new procedure for planning mobile robot trajectories by considering kinematic and dynamic constraints on the vehicle motion. This approach combines an original kinematic visibility graph planning method, an efficient path generation algorithm based on Beta-Spline curves and a cubic Spline speed profile definition technique. The maximum value of the curvature can be assured to be smaller than the value given by the constraints. Furthermore, speeds along the path are planned subject to the kinematic and dynamic constraints. The resulting trajectories provide ideal conditions for high precision path tracking and positioning. In the paper we present the application of the proposed methods to RAM-1, a mobile robot designed and built for indoor and outdoor industrial environment.

1. INTRODUCTION.

The path planning goal is the computation of a obstacle free route from an initial position to a final position while minimizing a certain criterion. Usually, the smallest distance measure is used to calculate the route. Many different approaches can be used to solve this problem by modelling the robot's environment and applying a heuristic search. Configuration space method [7], visibility graph representation, Voronoi diagram, and cell decomposition [4] are methods for environment modelling. None of these methods take into account the physical limitations of the vehicle to follow the planned route. However, these limitations have a significant influence for the ability of the path tracking to execute the route in the vehicle.

A continuous curvature curve can be fitted to the computed route for solving this problem. The path is obtained by sampling the fitted curve into a stream of robot postures. This process is named path generation and uses a planned route as entry. The emphasis of path generation process is in the geometric properties of the path to avoid discontinuities in the vehicle's steering. Continuous-curvature paths provide good conditions to be followed by autonomous vehicles. There are several methods which provide continuous curvature but have the disadvantage of lacking a closed-form expression to generate the curve coordinates. Thus, the computer requirements may preclude its application in real time. β-Spline path generation methods have proven efficiency to generate smooth continuous curvature paths with low computational cost [8].

Most of path planning methods assume that the robot executes the path at a low constant velocity. This assumption means that the vehicle's dynamic features do not affect the precision of the path tracking algorithm when the vehicle is following the path. However, in many applications a speed profile definition along the path is necessary (i.e. mobile obstacle avoidance [3]). So, the combination between a speed profile and a planned path is called trajectory. In this way, a trajectory is the composition of a spatial plan (path) with a time plan (speed profile).

In this paper a three-step trajectory planning method is described: i) route planning, ii) path generation and iii) speed profile definition. The planned route must keep a set of features for fitting a curve which provides a path with good kinematic conditions. These properties are a function of the fitting algorithm and the planning approach. In this way, the path generation method is presented in section 2. Section 3 relates the integration of the previous path generation method with a visibility graph planning approach. This integration is made in two steps: expanded environment computation (subsection 3.1) and kinematic visibility graph construction (subsection 3.2.). Section 4 deals with velocity planning. The velocities are set along the path to define a trajectory which satisfy the kinematic and dynamic constraints of the robot. Section 5 presents the implementation in the RAM-1 [10] autonomous mobile robot and real experiments with this vehicle. The final sections are devoted to conclusions and references.

2. BETA-SPLINE PATH GENERATION METHOD.

Let $\omega=\{\omega_0,\omega_1,...,\omega_k\}$ be a route as a set of sparse points computed by the path planner. Furthermore let (θ_0,κ_0) and (θ_k,κ_k) be the starting and ending heading-curvature pair. The path generator builds a curve $\tau(s)$ that starts at ω_0 with heading θ_0 and curvature κ_0, passes though the points $\omega_1,...,\omega_{k-1}$ and ends at ω_k with a heading of θ_k and curvature of κ_k. The path generator process samples this curve into a set of postures $Q=\{q_1,...,q_m\}$, which are used by the path tracking algorithm for executing the path. A posture q_i is composed of four basic elements: x_i, y_i, θ_i and κ_i. First two elements are position components, third is the heading with respect to a global work frame, and the last one is the curvature component. Moreover, all postures of Q are evenly spaced.

The path generation method described is based on cubic β-Spline curves due to their properties for fast computation [2]. Then, the problem is to compute a set of control points of the β-Spline, in such a way that the curve satisfies a set of requirements for providing goods conditions for mobile robot path tracking: i) continuity in position, heading and curvature; ii) the maximum curvature along the path is limited by the robot kinematic model; and iii) smoothly curvature variation.

The method is based on two properties of the β-Spline curves [8]:

Property 1: Let $\{V_{-2},V_{-1},V_0,V_1\}$ be a set of control points that defines a β-Spline curve. If this set lies over a circle with radius ρ, and the points are separated a distance $\delta \le 0.5\pi\rho$ over the perimeter of the circle, then the β-Spline curve defined by these points is contained in a concentric circle with radius ρ' defined by:

$$\rho' = \rho - s \qquad \text{where} \qquad s = \frac{\delta^2 \sqrt{\rho^2 - \frac{\delta^2}{4}}}{3\rho^2} \tag{1}$$

Property 2: It exists one set of values for the control points $\{V_{-2}, V_{-1}, V_0\}$ such that β-Spline curve f(λ) proves f(0)=P, f'(0)=P' and f''(0)=P'', where P, P' and P'' are the desired point, first and second derivatives of the curve at the beginning.

$$V_{-2} = P - P' + \frac{1}{3}P'' \qquad V_{-1} = P - \frac{1}{6}P'' \qquad V_0 = P + P' + \frac{1}{3}P'' \tag{2}$$

In the same way, there is one set of values of the control points $\{V_{-1},V_0,V_1\}$ such that f(1)=P, f'(1)=P' and f''(1)=P''.

$$V_{-1} = P - P' + \frac{1}{3}P'' \qquad V_0 = P - \frac{1}{6}P'' \qquad V_1 = P + P' + \frac{1}{3}P'' \tag{3}$$

However, the known parameters are the initial and final heading-curvature pair. So the relationship between (θ,κ) and (P',P'') must be defined. The following expressions provide this relation:

$$P'_y = \frac{\delta}{\sqrt{(\tan\theta)^2 + 1}} \qquad P'_x = P'_y \tan\theta$$
$$P''_x = \frac{\|P\|^3 \kappa}{P'_x \tan\theta - P'_y} \qquad P''_y = P''_x \tan\theta \tag{4}$$

The path generation method consists of the following steps:

1.- Building of the reference path (see figure 1).

a) Build a set of circles over the set ω. These circles have an initial radius ρ_i and their centers are in the line D_i that bisects the minor angle between two consecutive segments defined by (ω_{i-1},ω_i) and (ω_i,ω_{i+1}). The radius ρ_i should be greater than the minimum turning radius, provided by the kinematic and dynamic constraints.

b) Translate each circle C_i from ω_i a distance *s* (given by (1)) along D_i, such that the control points over the circle C_i are computed forcing the β-Spline curve passes through ω_i.

c) Determine the tangent segment T_i between the circles.

d) These set of actions builds a reference path composed by arcs and straight lines where the set of control points of the β-Spline curve will be computed. In figure 1 the reference path is shown with bold trace.

2.- Fitting the β-Spline curve.

a) The control points of the β-Spline curve are computed over the arcs of each C_i and over the tangent segment T_i. These control points are evenly spaced by a distance δ, in such a way that δ verifies the *property 1.*

b) Compute the phantom points [2] of the β-Spline using *property 2* to impose the starting and the ending conditions of the path.

c) Build the β-Spline curve f(λ) with the computed set of control points.

d) Sample the β-Spline curve into a stream Q of postures $q_i=(x_i,y_i,\theta_i,\kappa_i)$ evenly spaced.

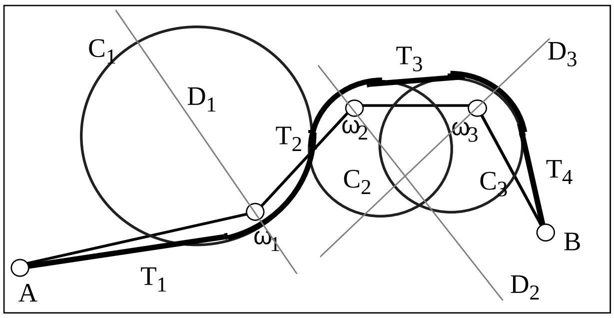

Fig. 1. Building the circles over the points of set ω.

3. KINEMATIC VISIBILITY GRAPH PLANNING.

Tangent graphs [5] are an extension of visibility graphs concept, which supply a geometric point of view of the route planning problem. The planning algorithms, which works with this approach, uses a polyhedral two dimensional environment model and the visibility relationship for graph building [4]. However, the graph is composed of circular arcs and tangent lines between arcs. Others versions are based on Radius-Independence graphs [6] for finding the shortest route for any robot modelled as a circle. None of these methods consider the vehicle's kinematic and dynamic features or minimize the control actions for path tracking. The proposed algorithm modifies the original tangent graph method in order to keep the above issues, and integrates the planning approach with the path generation method presented in the previous section. The planning action is made in two basic steps which will be developed in the next two subsections.

3.1. Expanded environment model computation.

The route planner algorithm finds the optimal obstacle-free route by using a visibility graph in an environment modelled by a set of polygons. This first step expands all the obstacles inside the environment in order to build a free obstacle path for avoiding the collision with the corners of the environment obstacles. The expansion factor is a function of the vehicle size, the minimum turning radius, and a safety distance related to the robot uncertainty model [9][13]. In figure 2, dotted lines show the expanded version of the original environment.

This action provides a safety planning with the non-point mobile robot. Expanded obstacles which share any area of the environment are considered as a single obstacle.

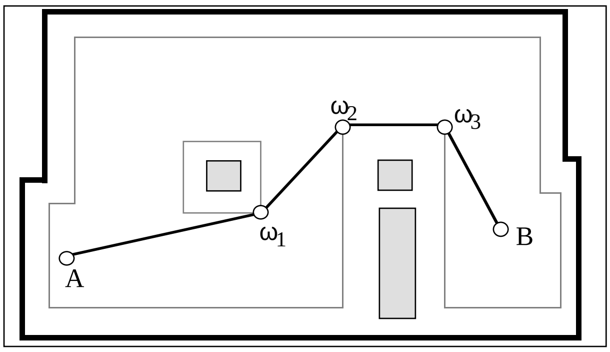

Fig. 2. Expanded environment model for route planning.

3.2. Kinematic visibility graph

The path generator needs a planned route with specific properties for making a path with good kinematic conditions. However, a planned method based on visibility graph approaches does not support this feature. Therefore, it is necessary to introduce kinematic information which provides a guide for planning the shortest route with the best path generation properties.

A visibility graph is defined by GV=(N,γ), where N is the graph nodes set $\{n_1,...,n_m\}$, and γ models the visibility relationship between two graph nodes of N. Thus, $\gamma(n_i,n_j)$=TRUE if the nodes n_i and n_j are visible, otherwise the function returns FALSE. A kinematic visibility graph is determined by (B_a,χ,γ). B_a is the admissible route segments subset, and it is defined by using the route segment graph set B. A segment b_i of B is composed of the nodes ${}^in_1$, ${}^in_2$, ${}^in_3$ of N in such a way that confirms the following expression:

$$b_i = ({}^in_1, {}^in_2, {}^in_3) / {}^in_j \in N$$
$$\gamma({}^in_1, {}^in_2) \wedge \gamma({}^in_2, {}^in_3) = \mathrm{TRUE} \qquad (5)$$

So, B is the collection of all possible route segment b_i of GV. The function $\chi(b_i)$ assigns a real positive number to each b_i which defines the C_i turning radius circle attached to the current segment. A route segment b_i belongs to the admissible route segments subset B_a if it verifies the following conditions:

- The result of the function $\chi(b_i)$ is equal or greater than the minimum turning radius ρ_{min} defined by the vehicle kinematic constraint. Moreover, the turning circle defined by $\chi(b_i)$ does not contain the left and right components of the current route segment and the circle arc used for reference path definition touches none of the environment obstacles.
- The obstacle, which touches the middle route segment node ${}^in_2$ of b_i, lies on the convex hull area defined by b_i (see figure 3).

All the route segments of the planned route must belong to the admissible route segment subset for the best path generation properties. Therefore, the heuristic graph search only works with the B_a components.

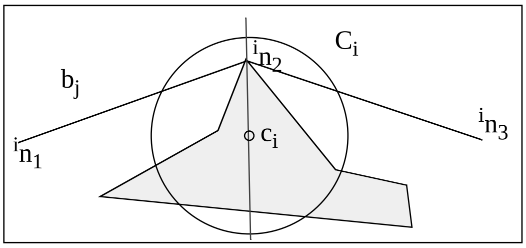

Fig. 3. Second admissibility condition.

The algorithm V* for kinematic visibility graphs is based on A* graph exploration. The route segment expansion is made by using the γ' function as an extension of the visibility function γ, and it is defined in the $B \times N$ space:

$$\gamma'(b_i, n) = \mathrm{TRUE} \Leftrightarrow \gamma({}^in_3, n) = \mathrm{TRUE} \qquad (6)$$

The expansion of b_i means that V^* selects a GV node **n** which verifies the following condition:

$$\gamma'(b_i, n) = \mathrm{TRUE} \wedge ({}^in_2, {}^in_3, n) \in B_a \qquad (7)$$

Furthermore, V^* builds the reference path between b_i and the new branch defined above. This action is made by determining the tangent segment between the circles defined by the χ function of b_i and $({}^in_2,{}^in_3,n)$. The route segment expansion is shown in figure 4.

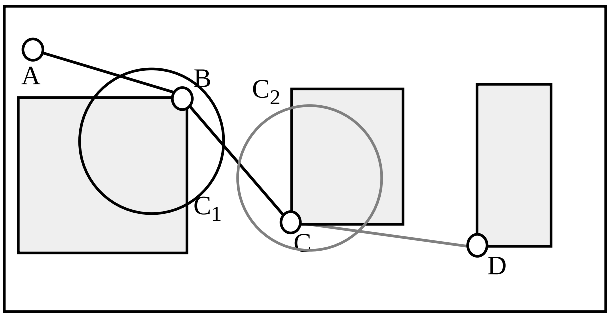

Fig. 4. Route segment expansion in V* algorithm.

The above figure presents the route segment expansion of (A,B,C) with the node D. The route segment has attached the turning circle C_1 provided by the function χ. A successful expansion with the node D must prove the following conditions:

- Route segment (A,B,C) and node D comply expressions (6) and (7).
- Tangent segment line between the turning circles C_1 and C_2 touches none of the obstacles.

In this case, the route segment (B,C,D) is considered as a component of the current planned route.

Thus, the V^* search algorithm, which provides the integration of the path planner and generator, is defined in the next scheme:

Procedure $V^*(n_s,n_f)$
    CLOSED={};
    OPEN=$\{b_i / b_i=(n_s,{}^in_2,{}^in_3)$ *belong to* $B_a\}$
    if $n_s==n_f$ *then route found*
    if $n_f \in b_i$ *belong to OPEN then route found*
    While OPEN is not empty and route not found do
        Choose the b_i *of OPEN with minimum* $F(b_i)$.
        OPEN=OPEN-$\{b_i\}$; *CLOSED=CLOSED* $\cup \{b_i\}$.
        if $n_f \in b_i$ *then route found*
        SUC= $\{b_j \in B_a / b_j=\varphi(b_i, n);\ \exists n \in N \wedge \gamma'(b_i, n)\}$.
        For each member b_j *of SUC do*
            if b_j *belong to OPEN or CLOSED sets then*
                Change the search tree from b_j *toward* b_i.
            else
                Build the associated reference path
                if reference path is admissible then
                    Add the route segment to the current route
                    Validate the built reference path.
                    OPEN=OPEN $\cup \{b_j\}$
                end if
            end if
        end for
    end while
    if OPEN is empty then route not found.
End procedure

The algorithm uses the A^* template for the graph exploration and tries to find a route from the start node n_s to the final node n_f. Thus, OPEN and CLOSED sets have the classical definition. The function F() is the A^* heuristic monotonic function form. Finally, function $\varphi(b_i,n)$ builds a branch as shown at expression (7).

Further improvement of the kinematic visibility graph is to translate the original visibility graph nodes at the environment concavities (figure 5.a) because it increases the B_a set module. This translation is made in a such way that allows building a turning circle which does not touch the obstacle over the translate node as shown in figure 5.b. So, in this figure, the original route segment (n_1,n_2,n_3) does not belong to B_a because it does not verify the second admissibility condition shown in figure 3. However, the new route segment (n_1,n_2',n_3) confirms this condition.

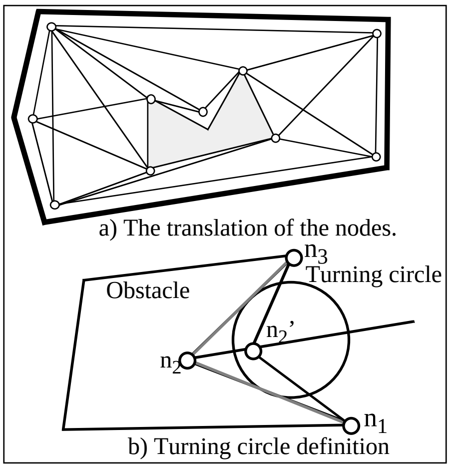

Fig. 5. Translation of the nodes at the concavities.

4. TRAJECTORY PLANNING AND GENERATION.

In order to convert a path Q into a trajectory $\tilde{Q}$ it is necessary to append a speed component to each posture of the path. In other words, the trajectory conversion process must turn each $q_i=(x_i,y_i,\theta_i,\kappa_i)$ into $\tilde{q}_i=(x_i,y_i,\theta_i,\kappa_i,v_i)$, where v_i is the posture speed component. This transformation is made by the definition of a parametric arc length speed function V(s). Such a curve is defined in the space-speed plane [12], in which the upper speed limit for each posture q_i of the path Q is represented. These constraints are obtained by taking into account kinematic limitations and dynamic effects on the vehicle [11]. So, the V(s) specification is made in such a way that it preserves all the posture speed limits, in order to obtain a trajectory with good kinematic and dynamic tracking conditions.

Let $F_s(t)$ be a space-time function which defines the path travelled per time unit. So, the time-speed function V(t) is defined as follows:

$$V(t) = \frac{d}{dt}F_s(t) \tag{8}$$

In this way, if the vehicle needs T time units to travel S_t space units along the path, the relationship between these parameters is given by the expression:

$$F_s(t) = \int_0^T V(t)dt = S_t \tag{9}$$

And the definition of the parametric arc length speed function V(s) in the space-speed plane is:

$$V(s) = V(F^{-1}{}_s(s)) \tag{10}$$

where $F^{-1}(s)$ is the inverse space-time function, which yields the time spent to travel a distance s along the path.

The computation of a space-speed function which fits all the speed limits at each posture q_i of Q is avoided, in order to reduce the complexity of $F_s(t)$. Hence, the path is divided into a set $S=\{S_1,..., S_p\}$ of path segments whose curvature can be approximated by either a linear variation law or a constant value. An ending speed limit v_{i+1} and the top acceleration ${}^iA_t$ are computed for the path segment S_i by using its curvature variation law and the kinematic and dynamic constraints of the vehicle. This operation sets up a speed control set $V=\{v_1,...,v_{p+1}\}$ for the path Q, where v_1 (always null) is the starting speed for S_1 and the remaining components v_i are speed border conditions between segments S_{i-1} and S_i.

A cubic space-time function $\sigma_i(t)$ is assigned to each S_i. Therefore, the function $F_s(t)$ is determined by the composition of these cubic functions.

The space-time function $\sigma_i(t)$, which belong to the i^{th} path segment S_i, is defined by the following expression:

$$\sigma_i(t) = {}^i\alpha_0 t^3 + {}^i\alpha_1 t^2 + {}^i\alpha_2 t + {}^i\alpha_3 \tag{11}$$

where $\sigma_i'(t)$ and $\sigma_i''(t)$ are the speed and acceleration functions respectively. If the vehicle navigates a distance s_i along this segment in a time t_i, then the following matrix expression arises:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ t_i^3 & t_i^2 & t_i & 1 \\ 0 & 0 & 1 & 0 \\ 3t_i^2 & 2t_i^2 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} {}^i\alpha_0 \\ {}^i\alpha_1 \\ {}^i\alpha_2 \\ {}^i\alpha_3 \end{bmatrix} = \begin{bmatrix} 0 \\ s_i \\ v_i \\ v_{i+1} \end{bmatrix} \tag{12}$$

The only free parameter is t_i. Then, the problem is to choose t_i in order to build a safety speed profile $\sigma_i'(t)$ for the i^{th} path segment, in such a way that it keeps the following constraints:

- Starting acceleration $\sigma_i''(0)$ inside the interval $[-{}^iA_t,{}^iA_t]$.
- Ending acceleration $\sigma_i''(t_i)$ inside the interval $[-{}^iA_t,{}^iA_t]$.
- Function $\sigma'_i(t)$ inside the range $[v_i,v_{i+1}]$.
- Continuity in speed and acceleration between two consecutive cubic space-time functions.

The computation of suitable parameter values based in the above constraint requires the application of numerical methods [8]. The proposed method evaluates a safe $\sigma_i(t)$ by using a closed form.

Two new cubic functions ${}^1\sigma_i(t)$ and ${}^2\sigma_i(t)$ can be obtained from $\sigma_i(t)$. Function ${}^1\sigma_i(t)$ covers the first half of the segment and ${}^2\sigma_i(t)$ the second. Let ${}^it_m$ be $t_i/2$ and $t \in [0,{}^it_m]$. If the acceleration is null at the beginning and end of the path segment S_i and the second derivatives of both functions ${}^1\sigma_i(t)$ and ${}^2\sigma_i(t)$are continuous at their joints, then, ${}^1\sigma_i''(0)=0$, ${}^2\sigma_i''({}^it_m)=0$ and ${}^1\sigma_i''({}^it_m)={}^2\sigma_i''(0)$.

The first function is defined by the set $\{v_i,{}^iv_m,{}^1s_i,{}^it_m\}$ and the second one by $\{{}^iv_m,v_{i+1},{}^2s_i,{}^it_m\}$ where ${}^iv_m$ is the average between v_i and v_{i+1}. The values for the elements of these sets are:

$$
{}^{i}t_m = \frac{s_i}{v_i + v_{i+1}} \qquad {}^{i}v_m = \frac{v_i + v_{i+1}}{2}
$$
$$
{}^{1}s_i = \frac{s_i(5v_i + v_{i+1})}{6(v_i + v_{i+1})} \qquad {}^{2}s_i = s_i - {}^{1}s_i \tag{13}
$$

Thus, the relationship between the starting and ending speeds for keeping the i^{th} path segment top acceleration constraint is defined by the following expression:

$$
\text{if}\,(v_i < v_{i+1})\,\text{then}
$$
$$
\left(v_{i+1} \le \sqrt{v_i^2 + {}^{i}A_t s_i}\right) \qquad \text{else}\left(v_{i+1} \ge \sqrt{v_i^2 - {}^{i}A_t s_i}\right) \tag{14}
$$

The algorithm which verifies the correctness of assigned speeds by taking into account the acceleration constraints is detailed bellow:

Procedure SpeedVerification(S)
    For each S_i belong to S so
        ${}^{i}A_t$=Acceleration constraint for S_i
        if v_i and v_{i+i} not verifies expression (14) then
            *if $v_i<v_{i+1}$ then v_{i+1}=sqrt(sqr(v_i)+${}^{i}A_t$*s_i)*
            *else v_{i+1}=sqrt(sqr(v_i-${}^{i}A_t$*s_i))*
        end if
    End loop
End procedure

The algorithm, which turns a path into a trajectory, is the following:

Procedure Path2Trajectory(Q)
    $\tilde{Q}$ ={}.
    {S}=Divide_path _into_path_segments(Q).
    {V}=Compute_Speed_Control_Set(S).
    Assign_top_acceleration(S).
    SpeedVefification(S)

    For each S_i belong to S do
        Compute ${}^{1}\sigma_i(t)$ and ${}^{2}\sigma_i(t)$ by using expression (13).
        $\sigma_i(t)$=Composition(${}^{1}\sigma_i(t)$, ${}^{2}\sigma_i(t)$).
        $V_i(s)=\sigma'_i(\sigma_i^{-1}(s))$. / Speed profile for S_i */*
        $q_{ini}=q_j$ $\quad S_i/S_i = \{q_j, q_{j+1},...,q_{j+p}\}$.
        For each posture q_k belong to S_i do
            d=Arc length from q_{ini} to q_k.
            w=V_i(d).
            $\tilde{q}_k$ =(q_k,w)
            $\tilde{Q} = \tilde{Q} \quad \tilde{q}_k$
        End loop.
    End loop.
    Return($\tilde{Q}$).
End procedure.

The performance of this algorithm can be improved by minimizing the size of the control speed set V. The v_i component of set V is the speed border condition between S_{i-1} and S_i path segments. If v_i is removed from V, these path segments can be consider as a single one. Therefore, the elimination of one control speed element reduces in one unit the size of the path segment set S. This fact decreases the number of iteration at the main loop in *Path2Trajectory* algorithm. The question is which components can be eliminated from V without affect the features of the space-speed function V(s).

The set V can be divided into a collection of speed subsets whose elements follow either increasing, decreasing or constant speed law. Let $W_i=\{v_j,v_{j+1},...,v_{j+n}\}$ be one of these subsets. Therefore, its components comply with the relationship either $v_k<v_{k+1}$, $v_k>v_{k+1}$ or $v_k=v_{k+1}$; for k=j to j+n.

If all components of W_i verify expression (14), then this relationship is also proved with the first element v_j and the last one v_{j+n}. This means that the space-time function defined by using all W_i elements has the same features than the curve obtained by using only the first and last components of W_i. In this way,only the components v_i of V which verify the expression (15) are considered.

$$
V_R = \{v_i\} / (\text{sgn}(v_{i+1} - v_i) \neq \text{sgn}(v_{i-1} - v_i)) \tag{15}
$$

Where *sgn(x)* is the signum function and V_R the reduced control speed set. The use of this set in *Path2Trajectory* algorithm reduces the S set size and therefore the time needed to plan the trajectory. Moreover, the speed profile generated by using V_R is smoother than the generated with V.

5. EXPERIMENTS.

The system has been integrated in the intelligent control architecture of RAM-1[10]. This mobile robot has four wheels located in the vertices of a rhomb. The front and rear wheels in the longitudinal axis are steered at the same time by a DC motor with a rigid link. The two parallel wheels are driven independently by DC motors. For path tracking the front and rear wheels are steered, and the parallel wheels are used with differential drive.

Figures 6, 7 and 8 show actual results of the application of the proposed integrated trajectory planning-generation method in RAM-1.

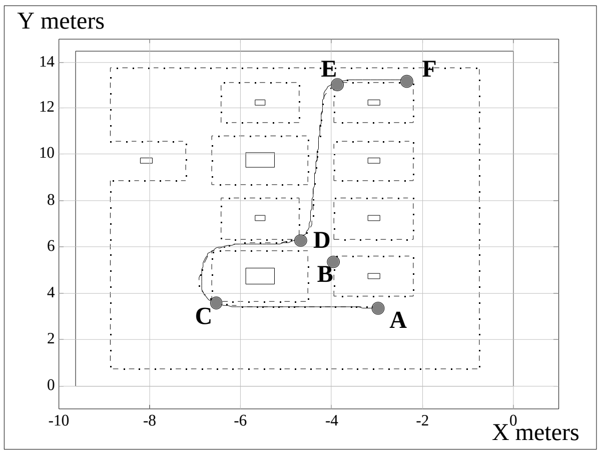

Fig. 6. Planned and generated trajectory and its real execution.

Figure 6 display the shortest path with best curvature properties in solid line. The solution (A,C,D,E) is not the minimum length route for travelling from point **A** to point **F**. However, this route keeps the required kinematics features for fitting a β-Spline curve with the path generation method proposed at section 2. The V* algorithm return a route with the appropriate properties for path generation.

Because of the closeness of the obstacle corner labelled B form start point A, the route segment (A,B,E) does not belong to B_a set because does not prove the first admissibility condition (χ((A,B,E))<ρ_{min}). Therefore, the route (A,B,E,F) is not accepted for path generation. Moreover, this figure shows the real execution with RAM-1 mobile robot of the planned-generated trajectory in dotted line.

Figure 7 shows the curvature produced by the path generation algorithm with the planned route (solid line). Furthermore, the curvature is without exception lower than 2 m^{-1} (ρ_{min}=0.5 m), which is the value determined from the kinematic, dynamic and practical experimentation with RAM-1 for the top speed. Besides the figure presents the real execution in dotted line.

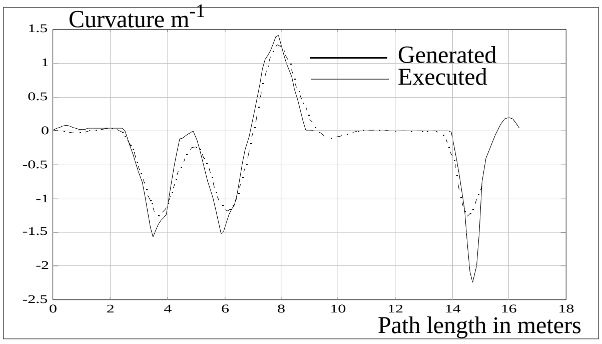

Fig. 7. Generated trajectory curvature and its real execution.

Figure 8 shows the smooth speed profile assigned to the generated path. This is continuous in acceleration, and its maximum values never exceed the limits imposed by the kinematic and dynamic constraints. The anticipative propierties of the control action, shown in dotted line, are due to the path tracking algorithm used in this experiment; a pure pursuit technique [1].

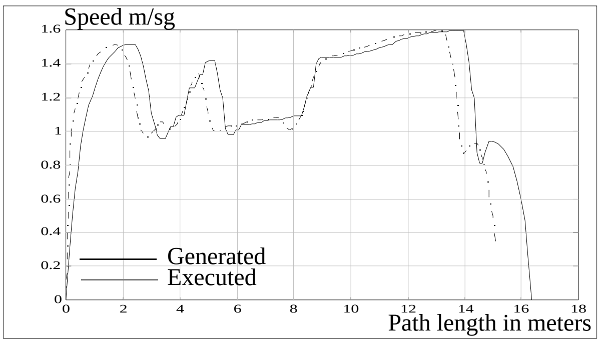

Fig. 8. Generated trajectory speed profile and its real execution.

6. CONCLUSIONS.

Planning optimal trajectories between obstacles, with good properties to be efficiently tracked by real autonomous vehicles and mobile robots is a complex problem that normally requires too much computations to be implemented in real time. This paper proposes a solution based on the combination of a kinematic visibility graph path planning algorithm, an efficient continuous curvature path generation method based on β-Spline curves and a speed profile generator based on cubic Splines.

The path generator builds a path with good kinematic conditions because of its continuity in position, heading and curvature. Moreover the curvature varies lineally and never exceeds the limits imposed by the kinematics of the vehicle. Speeds along the path are planned according to the kinematic and dynamic constraints of the vehicle and the path's features. Since the speed profile is made with a cubic spline curve, it has continuous acceleration and requires little computational time for its evaluation.

The kinematic visibility graph building is a time-expensive action (from a few secs needed for planning the example shown at Fig. 6, to several minutes on complex environment), but it is made off-line and the graph reconstruction is not necessary as long as the environment remains static. However, the β-Spline path generation method and the speed profile generator are able to produce a 25 m. path in about 34 msecs in a SPARC 2 workstation.

7. REFERENCES.

[1] Amidi O. (1990) "Integrated Mobile Robot Control". Carnegie Mellon University. Robotics Institute. Techinical Report CMU-RI-TR-90-1.

[2] Barsky B. A. (1987) "Computer graphics and Geometric Modelling Using Beta-Splines". Springer-Verlag.

[3] Kant K., Zucker S. (1.986). Toward Efficient Trajectory Planning: The Path-Velocity Decomposition. The International Journal of Robotics Research, Vol 5, No. 3.

[4] Latombe J.C. (1991) "Robot Motion Planning". Kluwer academic publishers. ISBN 0-7923-9129-2.

[5] Liu Y. H., Arimoto S. (1991) " Proposal of Tangent Graphs and extendend tangent graph for ptah planning of mobile robots. Proc. of 1991 IEEE International Conference on Robotics and Automation. pp 312-317.

[6] Liu Y. H., Arimoto S. (1995) " Finding the Shortest Path of a Disc Among Polygonal Obstacles Using Radius-Independence Graphs. IEEE Transactions on Robotics and Automation. October 1995, Vo. l 1, No. 5, pp 682-691.

[7] Lozano-Perez T. (1983) "Spatial Planning: A Configuration Approach" IEEE Transactions on Computers. Vol. 32, pp 108-120.

[8] Muñoz V, Ollero A., Prado M., Simon A. (1994) "Mobile Robot Trajectory Planning with Dynamic and Kinematic Constraints". Proc. of 1994 IEEE International Conference on Robotics and Automation. Vol 4, pp 2802-2807.

[9] Muñoz V., Martínez J.L., Ollero A. (1994)."Navigation with Uncertainty Position Estimation in the RAM-1 Mobile Robot" IFAC Symposium on Artificial Intelligence in Real Time Control. Valencia (Spain).

[10] Ollero A., A. Simón, F. García, and V. E. Torres (1993). "Integrated Mechanical Design of a New Mobile Robot".Proc. SICICA´92. Pergamon Press.

[11] Prado M., Simon A., Muñoz V., Ollero A. (1994). "Autonomous Mobile Robot Dynamic Constraints due to Wheel-Ground Interaction. Proc. of 1994 European Robotics and Intelligent Systems Conference. Vol 1, pp 347-360. Malaga (Spain)

[12] Shiller Z., Gwo Y. (1991). Dynamic Planning of Autonomous Vehicles. IEEE Transactions on Robotics and Automation, Vol 7 No. 2.

[13] Smith R. C. and P. Cheeseman (1986). "On the Representation and Estimation of Spatial Uncertainty". The International Journal of Robotics Research, Vol. 5, No. 4.