ARCS-09

PROCEEDINGS OF THE 2009 INTERNATIONAL CONFERENCE ON AUTOMATION, ROBOTICS AND CONTROL SYSTEMS

Editors:
Ken Chen, Kamal A.F. Moustafa, Dimitrios A. Karras

Orlando, Florida, USA
July 13-16 2009

ISBN: 978-1-60651-008-7

Contents

Real-Time Spline Trajectory Creation and Optimization for Mobile Robots

Saso Koceski, Natasa Koceska, Pierluigi Beomonte Zobel, Francesco Durante
Department of Mechanical, Energy and Management Engineering, University of L'Aquila, 67040 Località Monteluco, Roio Poggio (AQ), Italy

Abstract

In the field of mobile robotics, calculating suitable paths, for point to point navigation, is computationally difficult. Maneuvering the vehicle safely around obstacles is essential, and the ability to generate safe paths in a real time environment is crucial for vehicle viability.
A method for developing feasible paths through complicated environments using a baseline smooth path based on Hermite cubic splines is presented. A method able to iteratively optimize the path is also presented. This algorithm has been experimentally evaluated with satisfactory results.
Keywords: path planning, Hermite cubic spline, obstacle avoidance.

1. Introduction

Path planning with motion modelling is an important and challenging task that has many applications in the fields of robotics, artificial intelligence (AI), virtual reality, autonomous agent simulation, etc. The basic task for the motion constraint path planning is to perform navigations from one place to another by coordination of planning, sensing and controlling whilst maintaining a smooth motion trajectory. For point to point navigation, calculating suitable paths is computationally difficult. Maneuvering the vehicle safely around obstacles is essential, and the ability to generate safe paths in a real time environment is crucial for vehicle viability.

Numerous motion planners consider the car-like vehicle as a three-dimensional system moving in the plane and subjected to constraints on the curvature in addition to the non-holonomic constraint of rolling without slipping. The pioneering work by Dubins [1] showed that the minimal length paths for a car-like vehicle consist of a finite sequence of two elementary geometrical components: arcs of circle and straight line segments. From then, almost all of the proposed motion planners compute collision-free paths constituted by such sequences [2]. As a result, the paths are piecewise C^2: they are C^2 along elementary components, but the curvature is discontinuous between two elementary components. To follow such paths, a real system has to stop at these discontinuity points in order to ensure the continuity of the linear and angular velocities. Continuous-curvature curve generation has become a key problem for on-going research in this area.

A few types of splines have been proposed to solve this problem. Gómez-Bravo et al [3] proposed a method for continuous curvature B-spline-based path planning for parking manoeuvres; Berlung et al [4] used the Bezier curve in path planning, having considered minimizing the square of the arc-length derivative of curvature along the curve. Shimizu et al [5] presented a method that uses clothoid curve for smooth path generation for mobile robot which is equipped with an omni-directional camera and a laser rangefinder. Scheuer [6] and Fraichard [7] also used clothoid curves in their vehicle control experiment. Unfortunately, clothoids do not have a closed form making the control of their shapes difficult and dangerous in the presence of obstacles. Other recent works [8], [9] adopted cubic splines in their trajectory generation algorithm. Later methods progressed to higher order polynomials [10]. However, previous work has mainly been focused on the static trajectory generation problem and on finding feasible solutions for 2D applications. All these solutions often require a great deal of computational power as they evaluate the entire path space [4], [11]. In a real time environment it is beneficial to directly compute feasible paths continuously to allow for variations in the environment, control error and unmodelled sensor error.

In this work a practical method for developing feasible paths, for nonholonomic car-like robot, through dynamic environments using a baseline smooth path based on Hermite cubic splines is presented. The developed method takes into consideration robot constraints. A method for adjusting and bending the spline to avoid obstacles is developed. This method is able to iteratively refine the path to more directly compute a feasible path and thus find an efficient, collision free path in real time through an unstructured environment. The efficiency of the proposed solution is evaluated in a custom, physics-based simulation environment provided by Open Dynamics Engine (ODE) [12]. In the simulation a simple car-like robot model equipped with 3D scanner and GPS has been developed. The generated motion path is smooth and has continuous curvature on the whole state space of the motion, thus satisfying the major requirements for the implementation of such strategies in real-time navigation.

2. Path objectives

One of the objectives in this work is to study and develop a path planning algorithm for autonomous robot navigation or exploration in dynamic environments. The task can be divided into three parts: to plan a main path according to the pre-information, to keep tracking the difference between the map and the real environment, and then locally to amend the pre-designed path. This strategy can efficiently use the available information and reduce the re-planning time. It is supposed that the vehicle accepts a sequence of GPS waypoints used to define the high level mission. The robot's task then is to traverse through each waypoint.

The robot uses the 3D scanner to detect the surrounding environment and obtain the local information. The sensor is fixed to the robot body, and obstacles are mapped relative to the robot position and heading. The robot then uses the local information to generate the path to the destination.

Therefore, it is naive to pre-plan the entire robot path from the outset. Thus, re-planed the path is from the current position and heading, and using the most current obstacle map and do so in a receding horizon fashion. The path is also limited to only look ahead past the next waypoint. In this way, the next turn will be feasible and once executed, a new path will be generated to maneuver the vehicle into a suitable position for the following turn. For this reason, only three consecutive waypoints are used at a time, the most recently passed, and the following two points. This approach can be also used in the cases of unknown environments, where no pre-information is available before the path planning algorithm has been executed.

3. Path planning using Hermite cubic splines

In this paper the implementation of Hermite cubic splines as a tool for path planning is adopted. The mathematics involved in creating splines (which are piecewise polynomial functions), allow easy construction of smooth paths through a given, finite set of control points. Given *N+1* control points, and knowing the robot starting and goal position and orientation, a series of *N* spline segments are generated, with the three-order polynomial functions of variable t ($t \in [0, 1]$), to traverse these points, as:

$$\begin{bmatrix} X(t) \\ Y(t) \end{bmatrix} = \begin{bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} \quad (1)$$

where *X(t), Y(t)* are the coordinates of any point on the cubic spline, a_x, b_x, c_x, d_x and a_y, b_y, c_y, d_y are the coefficients to determine.

As the first derivative of the path is proportional to the vehicle heading, a non-continuous derivative would result in an infeasible path for this type of vehicle, but the second derivative is proportional to the vehicle steering angle and any discontinuities would force the vehicle to stop at each control point to adjust its steering. By creating a path with continuous derivatives, a smooth vehicle control, to remain in motion throughout the vehicle path, is guaranteed.

For these *N* segments of cubic spline, the required number of equations is (*8N*) in order to solve out all the coefficients. The known conditions are:

- the initial and final position and the robot orientation in these points;
- the continuity of positions at (*N-1*) control points;
- the continuity of 1st derivatives at (*N-1*) control points;
- the continuity of 2nd derivatives at (*N-1*) control points;

The total number of know conditions is (*6N+2*) which remove (*6N+2*) degrees of freedom from the 8N ones. The number of remained degrees of freedom is 2(*N-1*). This number is exactly the same as the unknown *x*-, *y*-coordinates of (*N-1*) control points. One set of points determines one path. By searching for the suitable control points, a feasible cubic splines path can be determined.

Considering just the local parameterization of the i^{th} cubic spline sequence in only the *x* direction, we will have:

$$x_i(t) = a_i + b_i t + c_i t^2 + d_i t^3 \quad (2)$$

Any cubic equation can be used to construct a cubic spline by identifying the constants a_i, b_i, c_i and d_i; however, the natural Hermite cubic polynomial has a unique property where it satisfies all four of the following boundary conditions:

$$\begin{aligned} &x_i(0) = a_i, \\ &x_i(1) = a_i + b_i + c_i + d_i, \\ &x'_i(0) = D_i = b_i, \\ &x'_i(1) = D_{i+1} = b_i + 2c_i + 3d_i \end{aligned} \quad (3)$$

By re-arranging and writing $x_i(1) = x_{i+1}$, we will have:

$$\begin{aligned} a_i &= x_i \\ b_i &= D_i \\ c_i &= 3(x_{i+1} - x_i) - 2D_i - D_{i+1} \\ d_i &= -2(x_{i+1} - x_i) + D_i + D_{i+1} \end{aligned} \quad (4)$$

If equate the 2nd derivatives p=1 for the (i-1)th segment, p=0 for the ith segment we will have:

$$\begin{aligned} & 2c_{i-1} + 6d_{i-1} = 2c_i \\ & \Rightarrow 2\left[3(x_i - x_{i-1}) - 2D_{i-1} - D_i\right] \\ & + 6\left[-2(x_i - x_{i-1}) + D_{i-1} + D_i\right] = \\ & 2\left[3(x_{i+1} - x_i) - 2D_i - D_{i+1}\right] \\ & \Rightarrow D_{i-1} + 4D_i + D_{i+1} = 3(x_{i+1} - x_{i-1}) \end{aligned} \tag{5}$$

At the first point, $x''_0(0) = 0$. So $c_0=0$ and $3(y1-y0)-2D_0-D_1=0$, so:

$$2D_0 + D_1 = 3(x_1 - x_0) \tag{6}$$

Similarly for the end section $x''_N(1) = 0$. So $2c_m+6d_m=0$ from which

$$D_{N-1} + 2D_N = 3(x_N - x_{N-1}) \tag{7}$$

Gather all this together the solution of the spline path in the matrix form, will be:

$$\begin{bmatrix} 2 & 1 & & & & & & \\ 1 & 4 & 1 & & & & & \\ & 1 & 4 & 1 & & & & \\ & & & \ddots & & & & \\ & & & & 1 & 4 & 1 & \\ & & & & & 1 & 2 & \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ \vdots \\ D_{N-1} \\ D_N \end{bmatrix} = \begin{bmatrix} 3(x_1 - x_0) \\ 3(x_2 - x_0) \\ 3(x_3 - x_1) \\ \vdots \\ 3(x_N - x_{N-2}) \\ 3(x_N - x_{N-1}) \end{bmatrix} \tag{8}$$

As mentioned previously, obstacles are mapped from various sensors and stored into the map Fig. 1. It should be noted that each obstacle must be inflated by at least half the width of the vehicle to guarantee that a collision does not occur (in the future discussion, talking about obstacles, the inflated obstacles will be considered). Considering this, and knowing the start and goal positions and orientations, for each pair of segments, the initial path composed of Hermite cubic splines (depicted in blue in Figure 1) is constructed.

4. Calculation of the spline path length and curvature

The fundamental properties of the spline path defined as above are its length and curvature. The calculation of the path's length is done by integrating the path arcs length elements. Considering a path that each path segment is given in parametric form $x=f(t)$, $y=g(t)$. Then the length l of each path segment is given by:

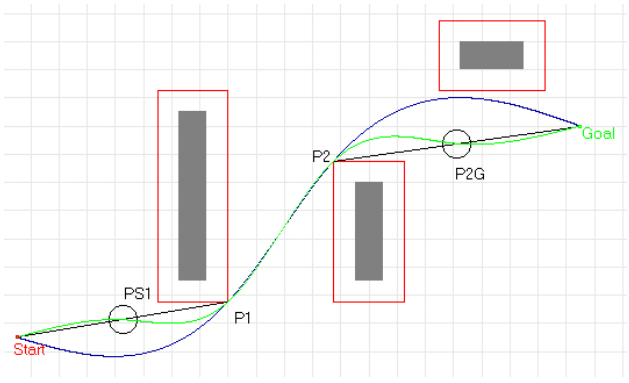

Figure 1. Spline path passing through the initial control points (blue). The introduction of additional control points can keep the initial path away from obstacles and decrease the path length (green).

$$l = \int_0^1 \sqrt{\left(f'(t)\right)^2 + \left(g'(t)\right)^2}\, dt \tag{9}$$

provided that f and g have continuous derivative in the interval [0, 1].

The curvature is a measurement of the rate of curving or simply how much the curve bends. Considering that the path is a curve that is traced at unit speed and is parameterized in term of the arc length s.

$$r(s) = \begin{bmatrix} x(s) \\ y(s) \end{bmatrix} \tag{10}$$

The unit tangent vector

$$t(s) = \frac{dr(s)}{ds} \tag{11}$$

is in direction of the curve for which s grows, as it can be seen in Figure 2. The angle of this vector in relation to the positive x-axis is denoted θ. The signed curvature is defined as:

$$k(s) = \frac{d\theta}{ds} \tag{12}$$

The normal vector $n(s)$ forms the counterclockwise oriented frame with $t(s)$. From this definition, the sign of curvature changes if the direction in which the curve is traced changes, but the curvature vector $k(s)n(s)$ does not depend on the direction. The figure also depicts the osculating circle in the point P.

This is the circle that best approximates the curve in this point. The radius of the osculating circle can be calculated as:

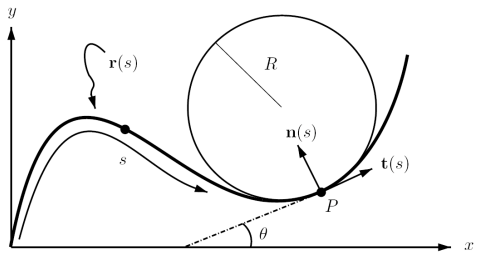

Figure 2. Definition of path curvature

$$R = \frac{1}{k(s)} \tag{13}$$

The osculating circle is not defined for points where k=0. So, in this case, where the path is constructed of Hermite cubic splines, the curvature is given by:

$$k = \frac{\left| \dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t) \right|}{\left[\dot{y}(t)^2 + \dot{x}(t)^2 \right]^{3/2}} \tag{14}$$

5. Path refinement and optimization

After the construction of the initial spline path it is useful to include additional points along the straight line path, connecting the initial control points.

The quantity and number of these points depend greatly on the relationship between the individual path lengths and the obstacles distribution. By including additional points, the initial path is kept closer to the nominal straight line path and it is therefore shorter. These points are only used for the initial path optimization and will not be rigid constraints in the final path. For demonstration, in the previous example depicted in Figure 1, we have included two additional points PS1 and P2G which are midpoints of the first and third path segments respectively. The path containing these points is depicted in green. As one may observe, addition of such points can iron the path and can be very useful in the environments where the robot has to traverse a corridor or a narrow passage.

In order to evaluate the quality of the paths, the following optimization function is introduced:

$$f = \frac{l}{l_m} + \left(\frac{\alpha}{d_{\min}} \right)^2 + k \cdot \frac{Rr_{\min}}{R_{\min}} \tag{15}$$

where the l_{min} is the Euclidian distance between Start and Goal, α is a weight constant of the distance from robot to obstacles, k is a weight constant of the minimum radius for robot driving, l is the path length, which is computed by Eq. 9, d_{min} is the minimal distance between any point on the path and the obstacles, given by:

$$d_{\min} = \min_{o \in O} \min_{P \in Path} \sqrt{(X_P - X_o)^2 + (Y_P - Y_o)^2} \tag{16}$$

where P denotes any point on the path, O is the set of all obstacles in the environment. R_{min} is the minimum radius along the whole path and Rr_{min} is the minimum turning radius, the robot can deal with. By minimizing this function, it is possible to shorten the path length, keep the robot as far as possible away from obstacles and smooth enough. Eventually the optimal path is a compromise of all the requirements. This optimization function strictly penalizes the trajectories that cause collision with the obstacles.

If it has been determined that the path collides with an obstacle, the spline has to be manipulated to avoid that obstacle. The proposed method is based on adding of an additional control point to the spline segments between the intersection points, to guide the path around the obstacle. As multiple collisions may have occurred, we will have a list of collision points. In order to minimize the computational time this method first calculates the convex hull of the obstacle. Then it calculates the pair of intersection points of the spline path and the convex hull of the obstacle. First point corresponds to the entry point of the spline into the convex hull and the second one, to the exit point, where the spline goes out of the hull area.

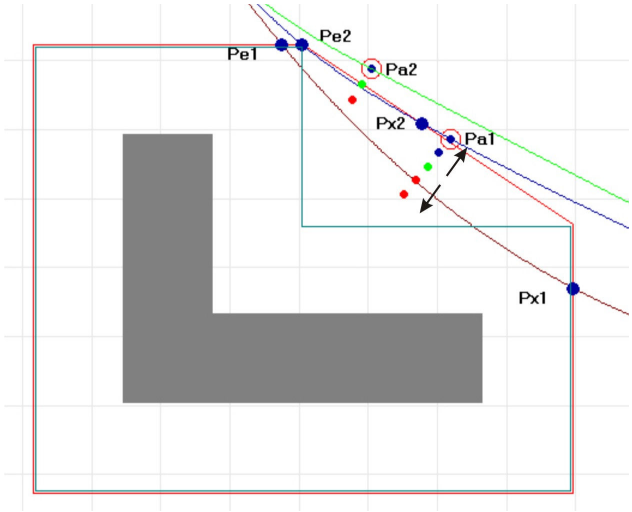

Figure 3. Initial spline path (brown) intersects the safety margin (dark green polygon) of the obstacle (grey). The computed convex hull (red polygon). Modified path after the first optimization is presented in blue, the final save path obtained after the second optimization is depicted in green.

The method then adds an additional point, P_a, on the segment, between the entry point (P_e) and the exit point (P_x)(Figure 3). Then this is moved point perpendicularly

to the segment $\overline{P_e P_x}$, by a predefined small distance (d_m) at each iteration, in both directions. The point P_a is continually updated, evaluated and checked against the obstacle's hull, until P_a is free of collision. This means that a path through P_a will no longer collide with the obstacle at that point. P_a is than added to the list of path control points. A new spline (depicted in blue in Figure 3) is computed through these points still having the desired characteristics but also passing through the new point. For each new spline the process is repeated. If the new spline still intersects the hull (P_{e2} ,P_{x2}), the collision will be shorter and closer to the edge of the hull. Thus the algorithm will continue to displace the spline around the remaining portion of the hull. Because the points are fit with cubic functions, a large number of control points within close proximity to each other can cause large deviations in the path and increasing of its curvature.

To avoid this, when adding a new point, any other control point within a given radius is removed. Therefore, in the example in Figure 3, the new added point P_{a2}, obtained in the second optimization step, replaces the point P_{a1}, calculated in the previous step.

6. Motion dynamics

Representing all the motion characteristics by analytical equations can be unpractical and out of the scope of this paper. A simplified motion model is considered in the current work. The developed moving robot model has two degrees of freedom and the dynamics of the model can be represented as a set of motion's equations in terms of mass, accelerations and steering angles as well as external force conditions, such as ground friction. The dynamics of a moving robot must follow the basic law of motion dynamics, which may be represented as a set of general ordinary differential equations in the form:

$$\frac{d^2X}{dt^2} = f\left(X, X', \delta\right) \tag{17}$$

where, $X, \dot{X} \in R$ is the motion state of the robot and its first derivative, and δ is the motion control input. We can recast the equation for our motion optimization problem in the form:

$$\begin{aligned} \frac{d^2X}{dt^2} &= \hat{f}\left(X, \dot{X}, \delta\right) + \Delta(\delta) \\ \Delta(\delta) &= f\left(X, \dot{X}, \delta\right) - \hat{f}\left(X, \dot{X}, \delta\left(\hat{a}, \hat{\theta}\right)\right) \end{aligned} \tag{18}$$

where, $\hat{\dot{X}}, \hat{a}, \hat{\theta}$ are approximate values of motion velocity, acceleration and direction of motion (i.e. a steering angle) respectively, and the motion control δ is a function of acceleration a and moving direction θ. A desired or predicted motion state of the moving object is pre-estimated by a set of approximate functions according to the state of moving object and the environment conditions related to the surrounding obstacle–space. The actual motion track is then computed. The difference between the predicted motion and the actual motion will be used for estimating the control input to mobile robot system.

7. Simulation experiments

In order to test the path planning algorithm a physics-based simulation environment provided by ODE has been developed. For the scope of this work a four wheels mobile robot which has a similar structure to the normal car is considered (i.e. two front steering wheels and two driven rear wheels). It is equipped with 3D scanner and GPS. All wheels have the same diameter and two rear wheels are conventional fixed wheels on the same axle and two front wheels are centered orientation wheels. The wheels are modeled using ODE's basic collision primitive, cylinder, and they are connected to the base body using motorized hinge joints with a horizontal rotational axis (vertical rotation plane). ODE also provides the possibility to set a desired velocity with a maximum force for each wheel.

The steering angle can be expressed as:

$$\phi = a\tan(L/R) \tag{19}$$

where L is the robot length and R is the distance from the middle point P_M of the rear wheels to the instantaneous center of curvature (ICC). The axes of all four wheels pass through the ICC during the driving.

Due to the fixed rear wheels, the robot is not permitted to change its orientation on the spot like the omni-directional robots. This special mechanical structure gives the constraint of the minimum radius or the maximum steering angle,

$$\left|\phi(t)\right| \le \left|\phi_{\max}\right| \quad or \quad R(t) \ge R_{\min} \tag{20}$$

The nonholonomic constraint for this kind of robot is expressed as:

$$\tan\theta = \dot{y}_{P_M} / \dot{x}_{P_M} \tag{21}$$

The angle θ stands for the orientation of the robot frame.

This constraint says that the direction of the translational velocity is the tangent direction of the path. The physical meaning of this constraint is that there is no possible motion in the axial direction. In other words, as long as the translational direction of the robot is coincident with the tangent direction of path, the nonholonomic constraint is naturally met. Testing environment cluttered with random positioned obstacles, was created, start and goal robot positions and orientations were set up (Figure 4).

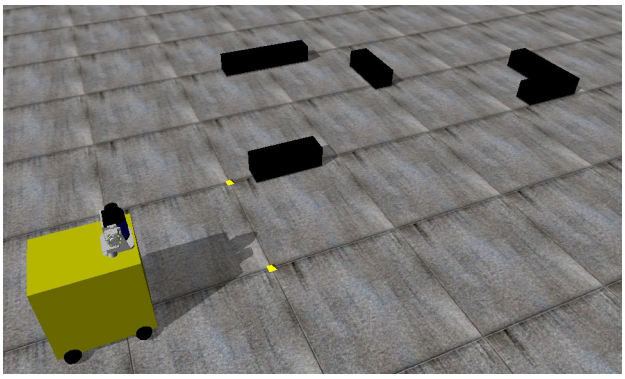

Figure 4. 3D simulation environment

The map with the obstacles and safety margins around, as well as the spline path computed in real-time using the algorithm presented in this work are depicted in Figure 5. The average end-position error along the whole path is less than 30 mm.

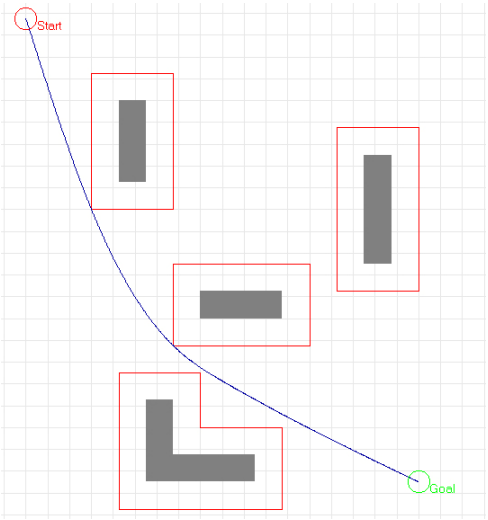

Figure 5. Calculated spline path after two optimization steps

8. Conclusion and future work

A novel motion constraint path planning approach for real-time navigation of mobile robots is proposed in this paper. The algorithm is able to produce a collision-free, time-optimal smooth motion trajectory. A 3D simulation has been conducted and the result is quite promising. The simulation result is quite satisfactory. The next step for our research is to refine the algorithm and look at path planning with more complex behaviors in simulated environments.

10. References

[1] L. E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. Amer. J. Math.,79:497-516, 1957.

[2] F. Lamiraux and J.-P. Laumond. Smooth Motion Planning for Car-Like Vehicles. IEEE Trans. of Robotics and Automation, 17(4):498-502, 2001.

[3] Gómez-Bravo, F., Cuesta, F., Ollero, A., and Viguria, A. 2008. Continuous curvature path generation based on ß-spline curves for parking manoeuvres. Robot. Auton. Syst. 56, 4 (Apr. 2008), 360-372.

[4] T. Berglund, H. Jonsson, and I. Soderkvist, "An obstacle-avoiding minimum variation b-spline problem," Proceedings of the 2003 International Conference on Geometric Modeling and Graphics, 2003.

[5] Shimizu, M. Kobayashi, K. Watanabe, K., Clothoidal Curve-based Path Generation for an Autonomous Mobile Robot, In Proc. of the 2006 International Joint Conference SICE-ICASE, 478-481, 2006.

[6] A. Scheuer and Th. Fraichard, "Planning Continuous-Curvature Paths for car-Like Vehicles," IEEE-RSJ Int. Conf. On Intelligent Robots and Systems, November 4-8, 1996. vol. 3, pp. 1304-1311.

[7] Th. Fraichard and J. M. Ahuactzin, "Smooth Path Planning for Cars," IEEE Int. Conf. On Robotics and Automation May 21-26, 2001.

[8] B. Nagy and A. Kelly, "Trajectory Generation for Car-Like Robots Using Cubic Curvature Polynomials," in Field and Service Robots 2001, Helsinki, Finland June 11, 2001.

[9] M. Saska, M. Macas, L. Preucil, L. Lhotska, Robot Path Planning using Particle Swarm Optimization of Ferguson Splines. In ETFA 2006 Proceedings [CD-ROM]. Piscataway: IEEE, 2006, ISBN 1-4244-0681-1

[10] S. Thompson and S. Kagami, "Continous curvature trajectory generation with obstacle avoidance for car-like robots," Proceedings of the 2005 International Conference on Computaional Intelligence for Modelling, Control and Automation and International Intelligent Agents, Web Technologies and Internet Commerce, 2005.

[11] Z. shiller and Y.-R. Gwo, "Dynamic motion planning of autonomous vehicles," IEEE Transactions on Robotics and Automation, vol. 7, no. 2, p. 241, 1991.

[12] Smith, R.: Open Dynamics Engine - ODE http://www.ode.org (2008)