Hybrid PSO-Cubic Spline for Autonomous Robots Optimal Trajectory Planning

Paulo Salgado*[1], Getúlio Igrejas**, Paulo Afonso***
* CITAB/ECT-Departamento de Engenharias, Universidade de Trás-os-Montes e Alto Douro, Vila Real, Portugal
** Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Bragança, Bragança, Portugal
*** IT-Instituto de Telecomunicações / ESTGA, Universidade de Aveiro, Portugal
psal@utad.pt, igrejas@ipb.pt, pafnaa@ua.pt

Abstract— **This paper presents a new version of the Particle Swarm Optimization algorithm where the particles are replaced by spline functions. The developed algorithm generates smooth motion trajectories with two times continuously differentiable curvature avoiding obstacles placed in the workspace. It can be used for autonomous robot path planning or transport problems. The spline based trajectory generation gives us continuous, smooth and optimized path trajectories. Simulation and experimental results demonstrate the effectiveness of the proposed method.**

I. INTRODUCTION

The scientific and industrial community have devoted much of their effort in creating optimization algorithms to solve global path planning problems. This is due to a growing and general interest of use of mobile robots, car driver, drones and industrial machines in different types of practical proposals [1–3]. Much of this work goes into the creation of intelligent, autonomous and mobile robots, particularly proposing new algorithms for motion trajectory generation, obstacle avoidance, position control, energy efficiency, location and navigation [4–5].

Intelligent robots should perform trajectory planning by optimally finding the best trajectory and avoiding static and dynamical obstacles in its workspace [6]. A good trajectory must satisfy a set of requirements as continuity, whereby the trajectory should be at least second order differentiable at every point, marginally safe to avoid obstacles, even when the robot deviates slightly from the planned trajectory, as well as to be as possible the short length or lower energy expenditure trajectory.

Global path planning is an important problem in many disciplines, including very-large-scale integration design, global positioning system applications, and autonomous robot navigation. Many trajectory-generation methods are proposed in the literature, but several problems persist. One of the fundamental functions is to generate motion trajectories for the robots considering several simultaneous criteria and constraints such as traveling distance, smoothness, security and feasibility [7]. These important criteria are normally unconsidered as a block in planning trajectory problem because it is almost an NP-hard problem. In these cases, there is no explicit solution or deterministic method to solve the problem in an optimal and global way, taking into account all the constraints. To overcome these difficulties the problem is usually simplified and the quality of the path solution is reduced, for example taking the trajectory as a piecewise linear path or even a sharp path [8]. Often these solutions impose abrupt and discontinuous modes of operation on the robot such as unexpected stop, rotate, and restart. This far-fetched mode frequently causes discontinuities, introduces delays, it is time and power consuming, and brings unnecessary deterioration wear on the robot parts.

This problem has been thoroughly studied, and the literature available in the area abounds specially for manipulators and two-dimensional mobile robots [9]. Other global methods, such as Cell Decomposition [10], [11], Skeleton, Potential Field [12], Visibility Graphs [13], Voronoi Diagrams [14], Vector Field [15,16], Probabilistic Roadmaps [17], Rapidly Exploring Random Trees [18] and many others have been proposed, each having their strengths and weaknesses. Therefore, a meritorious solution takes in account the weighed balance between computational expenses, trajectory flexibility and the minimum requirements of the problem.

In order to overcome the problems encountered solving the path planning problem by the classical motion planning techniques, such as high computational cost and time, the evolutionary techniques have been tested with some success. The main advantage is that it easy to deploy and the generation of an admissible solution is very fast - if there exists one. Particle Swarm Optimization, PSO, is included in this paradigm of solutions. It is a very simple, yet a very powerful heuristic optimization technique which has proved to be very effective in many complex optimization problems [19]. PSO offers many advantages when compared to Genetic Algorithm, GA, and other heuristic techniques, such as simplicity, fast convergence, and few tuning parameters [20]. Most of these mentioned path planning methods generate a path between a given set of points with specific constraints, consisting in a combination of straight lines and circular arcs. However, there is a curvature discontinuity at the straight line and circular arc joint. To overcome this problem, many researchers have modelled robot trajectories as piecewise quadratic or cubic Bézier curves [21].

These methods do not offer a continuous curvature or an arbitrarily setting of the second derivative at the starting point of a trajectory. The cubic or higher order

[1] orcid.org/0000-0003-0041-0256

978-1-4799-7678-2/17/$31.00 ©2017 IEEE

polynomials such as B-splines have been widely used as single curves to generate two times continuously differentiable curvature (trajectories), C^2 [22-24]. A numerically efficient path planning method using cubic splines and straight lines with wall collision avoidance constraint was discussed in [25].

For "point to point" navigation, computing suitable paths is difficult. Manoeuvring an autonomous vehicle safely around obstacles is essential, and the ability to generate safe paths in a real-time environment is crucial for vehicle viability.

In this paper, we propose a method for developing feasible paths through complicated environments using a baseline smooth path based on cubic splines founded by PSO algorithm. 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. This method is validated and its performance evaluated through a set simulations of hard and complex problems, with a great number of circular obstacles. The algorithm demonstrates a high success rate for all of the tested environments.

In the next section, we revised the conventional PSO algorithm in [19], specifically the numerical method to be used in the general optimization problem where is expected a discretized solution, generally designed by particles or points. We introduce the requirements for planning an optimal, continuous and security trajectory for autonomous robots, and finally, we present the adequacy of the cubic splines properties to be used in this work. Continuous-curvature path planning with obstacle avoidance is considered. The efficiency of the proposed algorithm is illustrated via simulation results.

The rest of this paper is organized as follows. In Section 2 we present the new continuous PSO algorithm for C^2 trajectories. The simulation and experimental results are presented in Section 3, followed by conclusions and future plans in Section 4.

A. Particle Swarm Optimization - PSO

The PSO method is one of the optimization methods developed for finding a global optimal of some nonlinear function [18]. It has been inspired by a social behaviour of birds and fish. The method applies the approach of problem solving in groups. Each solution consists of a set of parameters and represents a point in multidimensional space. The solution is called "particle" and the group of particles (population) is called "swarm". These particles are moved around in the search-space according to a few simple formulae, they are guided by their own best known position in the search-space as well as the entire swarm's best known position, iteratively trying to improve a candidate solution with regard to a given measure of quality.

Each particle i is represented as a D-dimensional position vector $\vec{x}_i$ and has a corresponding instantaneous velocity vector $\vec{v}_i$. Furthermore, it remembers its individual best value of fitness function and position $\vec{p}_i$ and the best position of the entire swarm $\vec{p}_G$ that represents the social knowledge. During each iteration k, the velocity update rule (1) is applied to each particle of the swarm.

$$\vec{v}_i(k+1) = \alpha\vec{v}_i(k) + c_1 r_1\left(\vec{p}_i - \vec{x}_i(k)\right) + c_2 r_2\left(\vec{p}_G - \vec{x}_i(k)\right). \quad (1)$$

where α is an inertia weight parameter, r´s are random numbers drawn from a uniform distribution between 0 and 1, c_1 and c_2 are weights also designed as 'cognitive acceleration coefficient', respectively for local or global best position. If any component of $\vec{v}_i$ is lower than $-v_{\max}$ or greater than $v_{\max}$, the corresponding value is replaced by $-v_{\max}$ or $v_{\max}$, respectively. The $v_{\max}$ is maximum velocity parameter.

Next, the position update rule (2) is applied

$$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k). \quad (2)$$

Generally, particles are initialized by giving them a random position and velocity in the swarm. Next, the update formulas (1) and (2) are applied during each iteration and the $\vec{p}_i$ and $\vec{p}_G$ values are updated simultaneously. The particles gradually reach the global best positions by communicating the personal best and the global best positions to each other. The algorithm stops if the maximum number of iterations is achieved or any other stopping criterion is satisfied.

In the context of this work, the PSO method does not use traditional particle entities. Instead, these were replaced by continuous functions, namely by cubic splines. The path is defined by Cubic Spline obeying certain C^2 continuity requirements throughout their domain and interpolating waypoints.

B. Trajectory planning problem

Consider a workspace $W \in \boldsymbol{R}^2$ with $(O_o, o = 1,\cdots,n)$ obstacles and X the trajectory waypoints $(X_j = (x_j, y_j), j = 1,\cdots,m)$, where m and n are the number of waypoints and obstacles, respectively. It is assumed that initial, X_0, and final waypoints, X_{m+1}, positions are given, but the intermediate points are unknown. Also, obstacles are static and their positions are known. So, the path is represented as a series of m +2 desired time-location pairs:

$$(t_0, X_0), (t_1, X_1), \cdots, (t_m, X_m), (t_{m+1}, X_{m+1}) \quad (3)$$

where $X_j \in \boldsymbol{R}^2$ denotes the desired location of the robot at time t_j, specified in task space. For simplicity, here we are taking $t_0 = 0$ and $t_{m+1} = 1$.

Given these waypoints, there is a unique piecewise-cubic trajectory, S, that passes through the points and satisfies a certain smoothness criterion.

Path planning in the aforementioned workspace W, considers the generation of smooth paths τ_j between two consecutive waypoints X_j and X_{j+1} in a local coordinate frame and stitch them together to generate the complete path $S = [\tau_1, \tau_2, \cdots, \tau_{m+1}]$, by aggregation of Spline interpolation. The path planning problem is to find a feasible path $\tau_j \in W$, by finding the best interpolating

spline connections, X^c such that the path avoids all the obstacles in between X_0 and X_{m+1} with a restriction that the curvature of complete path S is continuous and smooth.

In the context of path planning it is obvious that a path, S, needs to be continuous and, furthermore, when considering vehicles, which are unable to instantaneously change heading, we must also enforce the continuity of the first derivative, D^1S, in order to guarantee a feasible path. The additional restriction that the second derivative D^2S remain continuous allows the vehicle to remain moving throughout its path.

C. Cubic Splines as smooth trajectory

Specifically, we model the trajectory between times t_{j-1} and t_j as a cubic function:

$$x_j(t) = S_j(t) = a_j \Delta t_j^3 + b_j \Delta t_j^2 + c_j \Delta t_j + x_{j-1}(t_{j-1}) \quad (4)$$

where $x_j(t) \in C^2$ is the ordered pair and consists of two terms, the abscissa and the ordinate coordinate of the trajectory position in the time interval $t_{j-1} \le t \le t_j$, with $\Delta t_j = t - t_{j-1}$. The waypoints are $\{X_j = x_j(t_j)\}_{j=0,\cdots,m+1}$.

The path is represented by a spline $S(t) = \{S_j(t)\}_{j=1,\cdots,m+1}$, as a series of m+2 time-location waypoints positioned in W. Founded a set of sequential waypoints, there exists a unique set of coefficients $\{a_j, b_j, c_j\}_{j=1,\cdots,m} \in \boldsymbol{R}^3 \Leftrightarrow$ such that the resulting trajectory passes through the waypoints and has continuous C^2 profiles at each waypoint in the complete path.

The waypoints are the visible solution of the optimization process performed by the PSO algorithm. However, in their evolutionary strategy, it takes into account not directly these points, but the complete path trajectory curve (spline) by evaluating its performance in the generated path. Truly, the PSO-Cubic Spline algorithm, PSO-CS, is an optimization method in the space of continuous spline functions.

II. PSO IN SPLINES CONTINUOUS SPACE

A. PSO-CS

The PSO-CS use splines S instead of particles P. However, for simplicity of analysis, we are going to use here the same designation indistinctly. S_i represents a Spline with waypoints X_i. This spline belong to W space and has instantaneous velocity vector $\vec{u}_i(t)$, which is a Cubic Spline interpolation vector function. It has velocity waypoints at $\vec{U}_{ij} = \vec{u}_i(t_j)$ for the positional waypoint X_{ij} at instant t_j.

As other PSO algorithms, it remembers its individual best value of fitness function and position $\overline{S}_i$ as well as the global best of the whole swarm, stored in $\overline{S}_G$. During each iteration, the velocity update rule (5) is applied to each particle in the swarm.

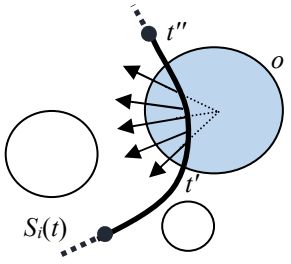

Figure 1. Repulsive force on Splice curve.

A 3rd term is used to represent a repulsive force for parts of Spline curve S_i that violates the space belonging to an obstacle.

$$\vec{u}_i^{(k+1)}(t) = \alpha \cdot \vec{u}_i^{(k)}(t) + C_1 \cdot r_1 \cdot \left(\overline{S}_i(t) - S_i(t)\right) + \\ + C_2 \cdot r_2 \cdot \left(\overline{S}_G(t) - S_i(t)\right) + C_3 \cdot r_3 \cdot \vec{R}_i(t) \quad (5)$$

The repulsion force, $\vec{R}_i(t)$ (6), acts on the Spline i, or in the portion that violates the space of the object, result of cumulative effect of all objects. It has a radial direction, from the center of the objects, CO, and equal intensity for length of line, as illustrated in Fig. 1 for object o. This force in null for the other parts of the spline that don´t violate obstacles (i.e. $u(t) = 0$), i.e.

$$\vec{R}_i(t) = \overline{S}_i(t) - CO_o, \text{ for } t' < t < t''; \text{ 0 otherwise.} \quad (6)$$

The particles' positions are again updated by Eq. (2). The solution candidates start their flight from random positions in a search area and in each iteration, the particles update their velocities according to (5) and positions by the following equation:

$$S_{i+1}^{(k+1)}(t) = S_i^{(k+1)}(t) + \vec{u}_i^{(k+1)}(t). \quad (7)$$

Flying is affected by a fitness function that assesses the quality of each solution. In this work, for the path planning problem proposed, the fitness of the spline is computed by the sum:

$$F_i = nv_i\left(1 + \beta \cdot VA_i\right) + LS_i + \delta \cdot SD_i. \quad (8)$$

where β and δ are weight parameters, nv is the number of object violated by the i^{th} spline, with the violating area VA, LS is the length of trajectory (Spline) and SD a function that measures the safety of trajectory, done by $SD_i = e^{-d_i^2/4}$, with d_i the shortest distance of spline i to the set of all obstacle objects, O.

This algorithm, which combines an appropriate cost function and continuous optimization techniques, guarantees the existence of a path with an arbitrary precision.

III. Simulation Experiments and Results

A. Methodology

We tested the PSO-CS algorithm for robot path planning in MATLAB platform. We assume that the start position of the robot is in $X_0 = [5,5]$ and the end position is in $X_{m+1} = [95,95]$. The number of waypoints is equal to one tenth of the number of obstacles. Three test environments were generated, with increasing number of circular obstacles, namely 50, 75 and 100 with one object bigger in center of the workspace. The size and location of other obstacles were generated randomly. For each environment, the algorithms PSO and PSO-CS were tested. The best Spline path is recorded. At end of the process, the best Spline trajectory is shown as well as the final population and the best performance evaluation in each iteration.

B. Results

PSO-CS method uses Spline curves as particles. This method was compared with the standard PSO, testing them with same examples and evaluating them with the same performance function. PSO was used to adjust the waypoints of Splines population, where the particles are the coordinates of these points, in a discrete optimization process. The results of both methods were compared. One hundred 'particles' have been used for simulations with 100 iterations. Three examples with 50, 75 and 100 objects were considered. The red line shown in Fig. 2-5 represents the optimum path generated by the algorithms and the filled circles represent obstacles.

The results for the first test were similar for both methods. In the second test, although both algorithms were able to find a solution the PSO method converged slower. However, for the third example with 100 objects, the PSO method fails (high cost function value of 1989), as shown in Fig. 2. It reveals the following weaknesses: does not perform a complete search in the problem space solution; does not avoid all obstacles or shows less ability to get around them; and has low diversity in the population (of particles), as shown in Fig. 3.

The proposed PSO-CS algorithm efficiently finds a collision-free path between the initial and destination points of the robot, a global solution with a cost function value of 270, as shown in Fig. 4. The diversity of the population is guaranteed as shown in Fig. 5 at the end of the iterative process.

IV. Conclusion

In this paper, the robot motion planning problem is treated as an optimization problem and a simple path planning based on particle swarm optimization is presented. We propose a new PSO-CS with its application to global path planning for autonomous robot navigation. In order to get smoother planned paths, the proposed algorithm uses a Cubic-spline smoothing technique.

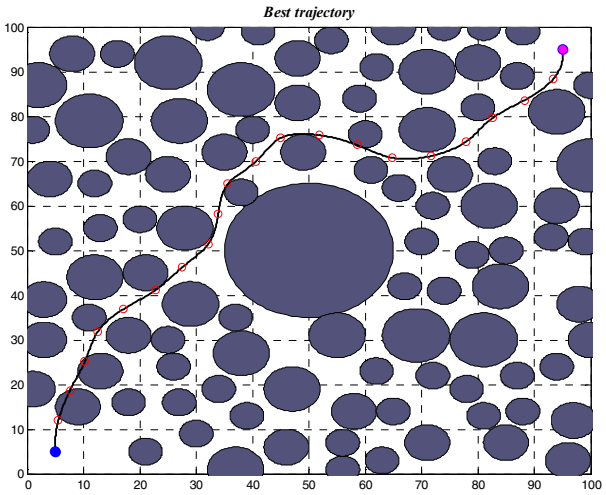

Figure 2. Solution of the PSO method.

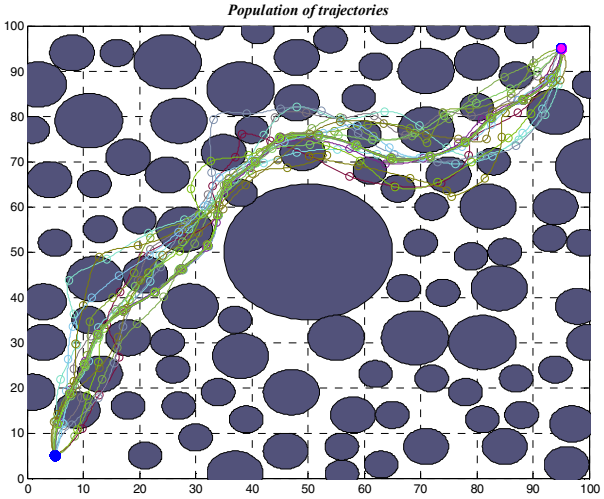

Figure 3. Population of particles at the end of the iteration process.

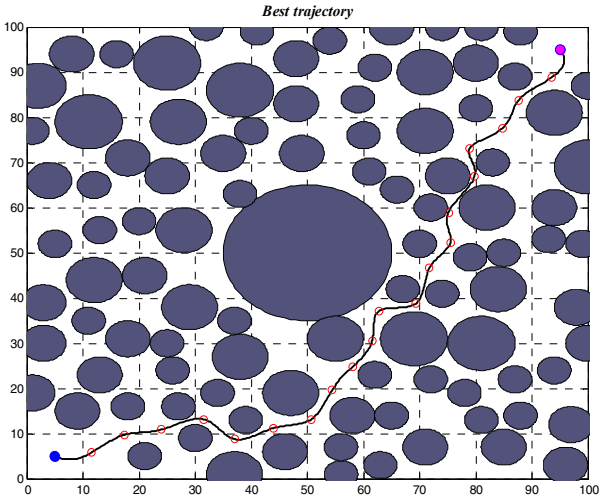

Figure 4. Solution of the PSO-CS method.

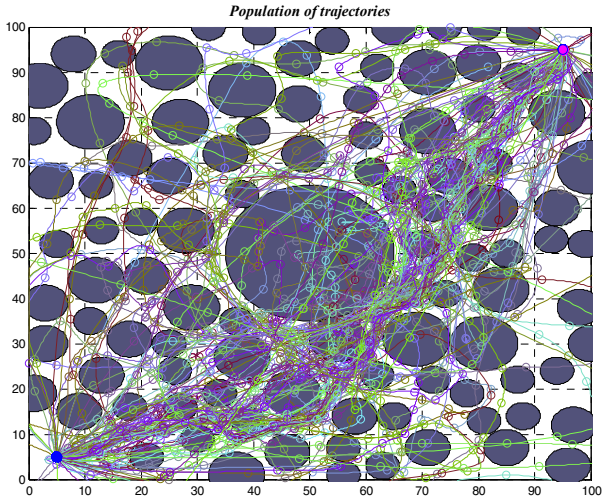

Figure 5. Population of particles at the end of the iteration process.

The proposed method places random obstacles on the workspace, generated between the robot start and goal positions, and finds the feasible Spline curve, with the appropriated waypoints. The result is a smoother planned path.

The method was successfully applied in global path planning for autonomous robot navigation and the simulations and experimental results shown that the proposed algorithm outperforms the conventional PSO algorithm, creating collision free trajectories. Moreover, the used strategy increases the search space and diversity of the population and, consequently, the possible solution domain

This method was performed without high computation demands and exhaustive search. Efficiency of the proposed algorithm is demonstrated via simulation results in different environments. This research study will be extended in the near future for dynamic obstacles.

REFERENCES

[1] Luigi Biagiotti and C. Melchiorri, "Trajectory Planning for Automatic Machines and Robots", Springer; 2008.

[2] Alaa Hassan, *Path Planning of Robot Manipulator Using Bezier Technique: Robotics and Automation*, LAP LAMBERT Academic Publishing, May 28, 2014.

[3] Steven M. LaValle, *Planning Algorithms*, Cambridge University, 2006.

[4] C. Goerzen Z. KongB. Mettle, "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance", Journal of Intelligent and Robotic Systems, pp. 57:65, Springer, Jan. 2010.

[5] Sgorbissa, A., & Zaccaria, R.. "Planning and obstacle avoidance in mobile robotics. Robotics and Autonomous Systems", vol. 60(4), pp. 628-638, 2012.

[6] Oroko, J., & Ikua, B., Obstacle Avoidance and Path Planning Schemes for Autonomous Navigation of a Mobile Robot: A Review. Sustainable Research and Innovation Proceedings, vol. 4, 2012.

[7] Prasenjit Chanak, and at.al., "Obstacle Avoidance Routing Scheme Through Optimal Sink Movement for Home Monitoring and Mobile Robotic Consumer Devices", IEEE Trans. on Consumer Electronics, Vol. 60, No. 4, pp. 596-604, Nov. 2014.

[8] S. K. Ghosh and at. al.,Online Algorithms with Discrete Visibility, IEEE Robotics & Automation Mag., Vol. 15 (2), pp. 67-76, 2008

[9] Steven M. LaValle, "Planning Algorithms", Cambridge University Press, 2006.

[10] A. Hourtash and M. Tarokh, "Manipulator path planning by decomposition: Algorithm and analysis," IEEE Trans. Robot. Autom., vol. 17, no. 6, pp. 842–856, Dec. 2001.

[11] C. Cai and S. Ferrari, "Information-driven sensor path planning by approximate cell decomposition," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 39, no. 3, pp. 672–689, Jun. 2009.

[12] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," IEEE Trans. Robot. Autom. , vol. 8, no. 5, pp. 501–518, Oct. 1992.

[13] H. Choset, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[14] L. E. Kavraki, P. Svestka, J-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," Robotics and Automation, IEEE Transactions on 12, no. 4, pp. 566-580, 1996.

[15] Gonçalves VM, Pimenta LCA, Maia CA, Dutra BCO, Pereira GAS, Vector fields for robot navigation along time-varying curves in n-dimensions. Trans Robotics, pp. 26:647–659, 2010

[16] Nelson DR, Blake D, Timothy B, Mclain W, Beard RW, Vector field path following for small unmanned air vehicles. IEEE Trans Control Syst Technol 48, pp. 5788–5794, 2006.

[17] S. M. LaValle, Rapidly-exploring random trees: Progress and prospects. S. M. LaValle and J. J. Kuffner. In Proceedings Workshop on the Algorithmic Foundations of Robotics, 2000.

[18] C. Hocaoglu and A. C. Sanderson, "Planning multiple paths with evoluionary speciation," IEEE Trans. Evol. Comput., vol. 5, no. 3, pp. 169–191, Jun. 2001.

[19] J. Kennedy, "Particle swarm optimization," In *Encyclopedia of Machine Learning*, pp. 760-766. Springer US, 2010.

[20] A. Abraham, and L. Jain, *Evolutionary multiobjective optimization*. Springer London, 2005.

[21] K. Yang and S. Sukkarieh, "An analytical continuous-curvature path smoothing algorithm," *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 561–568, Jun. 2010.

[22] T. Berglund, A. Brodnik, H. Jonsson, M. Staffanson, and I. Soderkvist, "Planning smooth and obstacle-avoiding B-spline paths for autonomous mining vehicles," IEEE Trans. Autom. Sci. Eng., vol. 7, no. 1, pp. 167–172, Jan. 2010.

[23] M. Elbanhawi, M. Simic, R.N. Jazar, Continuous path smoothing for car-like robots using b-spline curves, J. Intell. Robot. Syst., vol. 80 (1) pp. 23–56, 2015.

[24] R. Cowlagi and P. Tsiotras, "Curvature-bounded traversability analysis in motion planning for mobile robots," IEEE Trans. Robot. , vol. 30, no. 4, pp. 1011–1019, Aug. 2014.

[25] T. C. Liang, J. S. Liu, G. T. Hung, and Y. Z. Chang, "Practical and flexible path planning for car-like mobile robot using maximal curvature cubic spiral," Robot. Auton. Syst., vol. 52, no. 4, pp. 312–335, Sep. 2005.