NURBS Based Multi-objective Path Planning

Sawssen Jalel[1,2](✉), Philippe Marthon[2], and Atef Hamouda[1]

[1] LIPAH Research Laboratory, Faculty of Sciences of Tunis, Tunis El Manar University, 2092 Tunis, Tunisia
atef_hammouda@yahoo.fr

[2] Site ENSEEIHT de l'Institut de Recherche en Informatique de Toulouse (IRIT), University of Toulouse, 2 rue Charles Camichel, BP 7122, Toulouse, France
{sawssen.jalel,philippe.marthon}@enseeiht.fr

Abstract. Path planning presents a key question for an autonomous robot to evolve in its environment. Hence, it has been largely dealt in recent years. Actually, finding feasible paths and optimizing them for different objectives is computationally difficult. In this context, this paper introduces a new mobile robot path planning algorithm by introducing an optimized NURBS (Non Uniform Rational B-Spline) curve modelling using Genetic Algorithm to represent the generated path from the specified start location to the desired goal. Thus, given an a priori knowledge of the environment, an accurate fitness function is used to compute a curvature-constrained and obstacles-avoiding smooth path, with minimum length and low variations of curvature. The performance of the proposed algorithm is demonstrated through extensive MATLAB simulation studies.

Keywords: Robot path planning optimization · Genetic algorithm · NURBS curves parameterization · Weight parameter

1 Introduction

The autonomous mobile robotics aims, more specifically, to develop systems able to move independently. Direct applications are particularly in the fields of automotive, planetary exploration and service robotics. Which is why the problem of motion planning with obstacle avoidance has been extensively studied over the last decade since one of the crucial tasks for an autonomous robot is to navigate intelligently from a starting node to a target node.

The path planning environment can be categorized into two major classes which are static and dynamic. In the static environment, the whole solution must be found before starting execution. However, for dynamic or partially observable environments, replannings are required frequently and more update time is needed. Depending on environment type, path planning algorithms are divided into two categories, which are local and global methods. They might also be divided into traditional and intelligent methods. Eminent traditional path planning methods include potential field [1], visibility graph [2] and cell decomposition approaches [3]. As to intelligent methods, they include particle swarm optimization [4], neural networks [5], ant clony algorithms [6], fuzzy logic [7] and genetic algorithms. Obviously, each one of them has its own strengths and weaknesses which encourage researchers to search alternative and more efficient methods.

This paper deals with a novel path planning algorithm that consists of two steps. The first agrees to calculate the shortest, obstacles-avoiding polyline path from a start to a goal position. While the second undertakes to use both the wealth of genetic algorithms and the flexibility of NURBS curves to meet the path planner constraints.

The remainder of this paper is organized as follows: In Sect. 2, we present the theoretical background describing the theory of NURBS curves. The approach proposed is outlined in Sect. 3. Section 4 shows the simulation studies made to verify and validate the effectiveness of the presented method. Conclusions are finally presented in Sect. 5.

2 Theoretical Background

A p^{th} degree NURBS curve is a vector valued piecewise rational polynomial function of the form:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \tag{1}$$

where $\{P_i\}$ are the n control points, $\{w_i\}$ are the corresponding weights and the $\{N_{i,p}(u)\}$ are the p^{th} B-spline basis functions defined on the non-uniform knot vector U, by DeBoor-Cox Calculation as follows :

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{2}$$

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{else} \end{cases}$$

The degree, number of knots, and number of control points are related by $m = n + p + 1$. The knot vector is defined by $U = \{a, ..., a, u_{p+1}, ..., u_{mp-1}, b, ..., b\}$ with a multiplicity of a and b which is equal to the order of the curve. This constraint ensures the interpolation of the two endpoints. Throughout this paper, we assume that the parameter lies in the range $u \in [0, 1]$. For more details we refer the reader to [9].

NURBS have various useful properties which are ideal for the design of complicated geometry, making them a standard tool in the CAD/CAM and graphics community. In fact, they offer a common mathematical form for representing and designing both standard analytic shapes and free-form curves and surfaces. Furthermore, by manipulating the control points as well as the weights or the knot vector, NURBS provide the flexibility to design a large variety of shapes. Another useful property is the local modification, for example, if a control point P_i is moved or a weight w_i is changed, it will affect the curve only in $[u_i, u_{i+p+1})$.

Also not to overlook the fact that evaluation is reasonably and computationally stable. As well they have clear geometric interpretations and are invariant under scaling, rotation, translation and shear as well as parallel and perspective projection.

A considerable amount of research has been carried out in the parametrization domain. Indeed, finding a correct parameterization and the weight of the control points when calculating the curve have been the main issues in curve fitting techniques. Many evolutionary optimization techniques have been successfully applied [10,11]. Through this paper, we aim to exploit the effect of the weight parameter. For this, we recall that the weight of a point P_k determines its influence on the associated curve. Therefore, increasing (decreasing) w_k pulls (pushes) the curve toward (away from) this point (Fig. 1). It should also be noted that a control point with zero weight removes the contribution of this point on the generated curve.

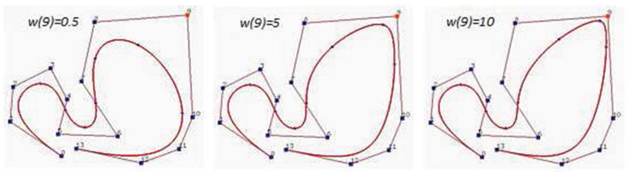

Fig. 1. NURBS Curve with w(9) Varying

3 Proposed Path Planning Optimization Algorithm

In this section, an optimized NURBS curve modelling using Genetic Algorithm for mobile robot navigation is presented. Thus, given a map with a priori knowledge of the locations of obstacles and positions of departure and arrival, the objective is to compute an optimal trajectory. In other words, a curvature-constrained smooth path avoiding collisions, with minimum length, and having low variations of curvature. For this, we assume that the robot is represented by a circle of diameter d (taking into account the robot's dimensions) and minimum radius of curvature ρ_{min}. As for the workspace Ω, it is considered as the union of the subspace of abstacles-free configuration Ω_{free} and the subspace of abstacles configuration Ω_{obst}.

Algorithm 1 describes the procedure of shortest polyline path computation. This algorithm starts by extracting the skeleton of the obstacle-free space, to which, it connects the start and target point giving, accordingly, the extended skeleton X upon which the robot may move (Fig. 2(a)). Then, a morphological process is applied to classify the pixels of X into End Points (EP), Junction Points (JP), Critical Curve Points (CCP) and Curve Points (CP). Afterward, a weighted graph is constructed by associating $EP(X)$, $JP(X)$ and $CCP(X)$ with the vertices, and the set of $CP(X)$ with the edges. The weights of edges are generated by considering the Euclidean distances between vertices. Before applying Bellman Ford Algorithm to calculate the shortest path between S and T, the algorithm applies a graph reduction step to remove the edges that represent inaccessible corridors by the robot [8].

Algorithm 1. Shortest polyline path computation

```
Input: $\Omega$, $S$, $T$, $d$
Output: Shortest polyline path $S_{opt}$
1  begin
2      $X = SkeletonExtraction(\Omega_{free}, S, T)$;
3      $EP(X) = [\bigcup_i \epsilon_{\theta_i(\overline{A})}(\overline{X})] \cap X$;
4      $JP(X) = [\bigcup_i \epsilon_{\theta_i(B)}(X)] \cup [\bigcup_i \epsilon_{\theta_i(C)}(X)]$;
5      $CCP(X) = CurveCriticalPoint(X)$;
6      $CP(X) = X \backslash (EP(X) \cup JP(X) \cup CCP(X))$;
7      $G = GraphConstruction(EP(X), JP(X), CCP(X), CP(X))$;
8      $G' = GraphReduction(G, d, \Omega)$;
9      $S_{opt} = BellmanFordAlgorithm(G', S, T)$;
10     return $(S_{opt})$;
```

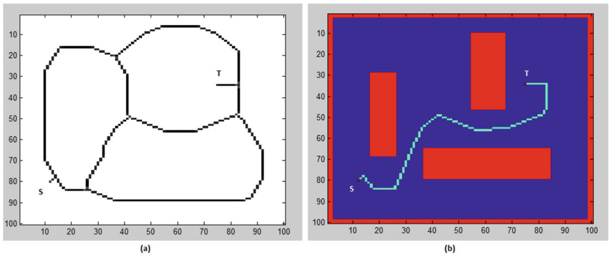

Fig. 2. Shortest path finding process. (a) Extended skeleton. (b) Shortest path between S and T.

As shown Fig. 2(b), the shortest path S_{opt} is modeled by a set of connected edges. This trajectory is not smooth and don't respect the curvature constraint. Then, it needs a mathematical model to refine it. Hence, the importance of using NURBS curves given their properties previously cited. Thus, a good parameterisation is needed to obtain the desired result. In fact, generating a NURBS curve requires the specification of the set of control points as well as their associated weights. In our algorithm, the control points are extracted from the selected polyline path S_{opt}:

$$P = \{P_i\}_{i=1..n} = JP(S_{opt}) \cup EP(S_{opt}) \cup CCP(S_{opt}) \quad (3)$$

where $JP(S_{opt})$ are the junction points of S_{opt}, $EP(S_{opt})$ are the end points of S_{opt} and $CCP(S_{opt})$ are the critical curve points corresponding to variations of directions of S_{opt}. Regarding the weight factor, we propose a novel parameterization method based on genetic algorithm.

3.1 Genetic Representation

The chromosome structure must have sufficient information about the entire path from the start to the end point in order to be able to represent it. In our context, an individual is a NURBS curve representing a path candidate. Such a curve is defined by a set of n weighted points for which P_1 and P_n are always the starting and destination configurations. Each gene represents the location of a control point (x and y coordinate) and its associated weight. It is noteworthy that location of this point is fixed while its associated weight is varying.

3.2 Fitness Function

During each generation, the population of paths is evaluated to quantify their degree of elitism which depends on how suitable the solution (path) is according to the problem. Consequently, an individual's fitness value should be proportional to its survival ability. Most GA-based path planning existing methods consider the path length as fitness function to minimize and neglect feasibility. Consequently, the resulted path may not be followed by the robot if it does not respect the curvature limit related to its minimum radius of curvature. In addition, having low variation of curvatures along the path is of considerable importance.

In this study, we propose an accurate evaluation function that depends on four parameters: the safety, the feasibility, the length and the curvatures's standard deviation of the path. The objective function for a path C is then defined as:

$$f(C) = \alpha.P_{safety}(C) + \beta.P_{feasibility}(C) + \gamma.P_{length}(C) + \delta.P_{\sigma}(C) \tag{4}$$

where $P_{safety}(C)$ is the path safety constraint and which denotes the number of collisions:

$$P_{safety}(C) = |P_{col}| \tag{5}$$

where:

$$P_{col} = \{p_i \in C, i = 1..k1\} = C \cap \Omega_{obst} \tag{6}$$

$P_{feasibility}(C)$ is the path feasibility related to the robot's minimum radius of curvature and determined as the number of curve points that exceeds the absolute value of the curvature limit:

$$P_{feasibility}(C) = |P_{cur}| \tag{7}$$

where:

$$P_{cur} = \{p_j \in C, j = 1..k2 | |curvature(p_j)| > \frac{1}{\rho_{min}}\} \tag{8}$$

$P_{length}(C)$ is the path length, determined as:

$$P_{length}(C) = \int_0^1 \|C^{'}(u)\| \,\mathrm{d}u \tag{9}$$

And finally $P_\sigma(C)$ denotes the standard deviation of curvatures along the path which measures the dispersion of the of the curvatures's values around their arithmetical mean c_{mean}:

$$P_\sigma(C) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(c_i - c_{mean})^2} \tag{10}$$

α, β, γ and δ are weighting factors. Their values will be fixed through experimentation. It is abvious, in our method, that the best individual will have the minimum fitness value.

3.3 Genetic Operators

Classical GA algorithms primarily use three genetic operators: selection, crossover and mutation that simulate the process of natural selection and gives them their powerful search ability.

Selection Method. There are many selection strategies such as rank-based selection, roulette wheel selection, elitist selection and tournament selection. In this study, the rank-based fitness assignment is employed.

Crossover Operator. The fundamental role of crossover is to allow the recombination of information in the genetic heritage of the population by combining the features of two parents to form two offsprings. Conventional crossover methods include single-point crossover and two-point crossover which is proven more effective. To enhance the population diversity, an n-point crossover is used where n is the number of crossing points. The number and sites of the crossover are randomly generated in each iteration. Therefore, the two offsprings are obtained by copying in the first offspring the genes of $Parent1$ up to the first crosspoint then supplementing with the genes of $Parent2$ up to the second crosspoint and so on until the n^{th} crosspoint as shown in Fig. 3, where n equals 3 and the crossover sites are 2, 5 and 7.

Mutation Operator. For the purpose of increasing the population's diversity, mutaion operation is required in genetic algorithms despite its low probability. In this paper, the classical mutation operator is used which takes as input a selected individual for mutation and returns a mutant individual obtained by local transformation of one of its genes. In fact, a gene is replaced with a new gene not included in the individual. Thus, the mutation operator modifies randomly the characteristics of a solution, which allows to introduce and maintain the diversity within our population.

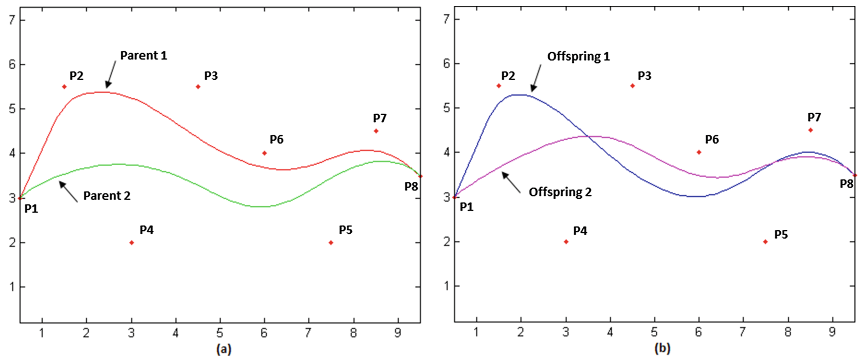

Fig. 3. (a) Two chromosomes (NURBS curves approximation of 7 control points) before the crossover: weight vector of Parent 1(0.1, 3.4, 4.5, 1.2, 2.3, 3, 4, 2), weight vector of Parent 2(5.2, 0.4, 3.2, 3.6, 5.1, 2, 2, 2) (b) Generated offsprings after the crossover: weight vector of offspring 1(0.1, 3.4, 3.2, 3.6, 5.1, 3, 4, 2), weight vector of offspring 2(5.2, 0.4, 4.5, 1.2, 2.3, 2, 2, 2).

4 Experimental Results

To investigate the feasibility and effectiveness of the proposed algorithm, three simulation experiments were conducted. Various input parameters related to the environment and the robot are summarized in Table 1. Likewise, different values of the GA parameters are considered as listed in Table 2. We notice that Blue denotes an accessible area while Red denotes an impenetrable area. S and T represent the starting point and the goal point. All generated NURBS curves are of degree 4.

Table 1. Input parameters of environment and robot

Parameters	Map I	Map II	Map III
Space size	50*60	100*100	80*80
Start position	(28,46)	(5,80)	(33,48)
Target position	(21,7)	(67,12)	(47,8)
Robot diameter	2	1	2
Minimum radius of curvature	0.4	0.33	2

For Map I, after calculating the shortest polyline path from S to T, a set of 12 control points (red points in Fig. 4(a)) is determined. Initial population is created by generating randomly values of weights between 0.1 and 5. This population is evolved generation by generation to calculate the optimum path with respect to the system requirements. We note that with a crossover probability equal to 0.6 and a mutation probability of 0.2, the algorithm succeeded to provide, within 5

Table 2. Input parameters of GA

Parameters	Map I	Map II	Map III
Chromosome length	12	16	23
Population size	20	100	100
Crossover probability	0.6	0.07	0.7
Mutation probability	0.2	0.19	0.4
Max generations	50	100	40

iterations (Fig. 4(b)), the best solution. In fact, the best individual of the first generation has a fitness value of 22.25, a length equal to 73.75 and a standard deviation of curvatures equal to 0.19. As seen in Table 3, these values have been optimized to provide an optimal path of length equal to 69.44, with 20.95 fitness value and 0.18 as final measure of curvatures dispertion. The weight vector of the computed path is $W = (0.92, 2.56, 2.77, 1.09, 3.21, 0.23, 2.09, 4.53, 0.94, 0.99,$ $1.31,\ 3.85)$.

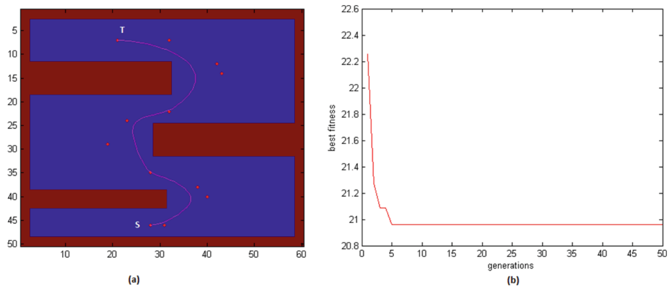

Fig. 4. Simulation results under Map I. (a) Generated path. (b) Evolution procedure of the proposed GA based mobile robot path planning algorithm (Color figure online).

As the robot's minimum radius of curvature of the first experiment was set to 0.4, the generated path must have 2.5 as the maximum allowable value of curvature. This constraint was ensured and the maximum value of curvature is 0.54.

Figure 5 shows the simulation results of the proposed algorithm under Map II. The obtained path is of length 105.95 computed from the approximation of 16 control points and it has the value 0.28 as standard deviation of curvatures. The corresponding weight vector is $W = (3.03, 2.48, 3.18, 0.50, 3.03, 4.92, 0.30,$ $1.24,\ 3.45, 1.10, 0.17, 3.91, 4.45, 1.13, 0.57, 1.03)$. For Map III, the planned path is modeled by a NURBS curve which approximates a set of 23 control points (Fig. 6). The length of this curve is 143.39 and the standard deviation of curvatures equals 0.24. The corresponding weight vector is $W =$(1.75, 2.20, 2.50, 4.64, 2.96, 2.22, 0.27, 4.85, 3.95, 1.80, 4.55, 0.20, 0.15, 3.35, 4.18, 0.51, 0.15, 1.92, 2.16, 2.18, 2.41, 2.43, 2.84). Experimental results are summarized in Table 3.

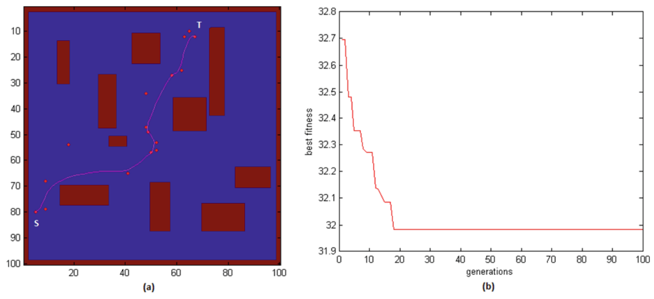

Fig. 5. Simulation results under Map II. (a) Generated path. (b) Evolution procedure of the proposed GA based mobile robot path planning algorithm.

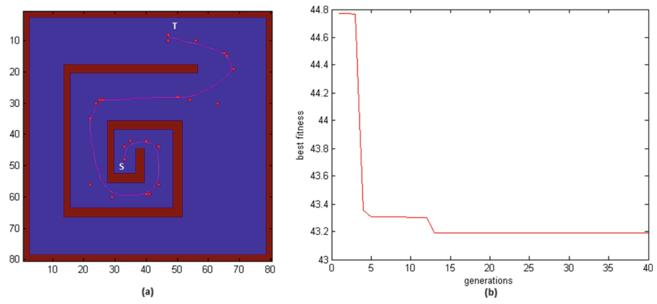

Fig. 6. Simulation results under Map III. (a) Generated path. (b) Evolution procedure of the proposed GA based mobile robot path planning algorithm.

Table 3. Output results

Parameters	Map I	Map II	Map III
Best fitness	20.95	31.98	43.19
Standard deviation of curvatures	0.18	0.28	0.24
Path length	69.44	105.95	143.39
|Max curvature|	0.54	1.49	0.48

As shown, the presented algorithm can be applied to mobile robot global path planning under different environments and is able to guide a mobile robot while achieving an optimal or near-optimal collision-free path thanks to the proposed multi-objective fitness function used in the parameterization step of the NURBS curve modelling the generated solution.

5 Conclusion

This paper has presented a novel approach to the path planning problem based on an optimized NURBS curve using genetic algorithm. It has introduced an accurate fitness function showing that the proposed algorithm is capable of efficiently generating a smooth and obstacle avoiding path with minimal length and minimal variations of curvature. Future studies include investigating the path planning task with different optimization methods. Furthermore, it will be interesting to adopt our algorithms into real world applications. Also, it involves to research the problem of path planning in an unknown dynamic environments.

References

1. Khatib, O.: Real time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. **5**, 90–98 (1986)
2. Nilsson, N.J.: A Mobile automaton: an application of artificial intelligence techniques. In: Proceedings of the 1st International Joint Conference on Artificial Intelligence, Washington, pp. 509–520 (1969)
3. Lingelbach, F.: Path planning using probabilistic cell decomposition. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 467–472 (2004)
4. Zhang, Y., Gong, D.W., Zhang, J.: Robot path planning in uncertain environment using multi-objective particle swarm optimization. Neurocomputing **103**, 172–185 (2013)
5. Bin, N., Xiong, C., Liming, Z., Wendong, X.: Recurrent neural network for robot path planning. In: Liew, K.-M., Shen, H., See, S., Cai, W. (eds.) PDCAT 2004. LNCS, vol. 3320, pp. 188–191. Springer, Heidelberg (2004)
6. Cong, Y.Z., Ponnambalam, S.G.: Mobile robot path planning using ant colony optimization. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pp. 851–856 (2009)
7. Hassanzadeh, I., Sadigh, S.M.: Path planning for a mobile robot using fuzzy logic controller tuned by GA. In: 6th International Symposium on Mechatronics and its Applications (ISMA), pp. 1–5 (2009)
8. Jalel, S., Marthon, P., Hamouda, A.: Optimum path planning for mobile robots in static environments using graph modelling and nurbs curves. In: 12th WSEAS International Conference on Signal Processing, Robotics and Automation(ISPRA), pp. 216–221 (2013)
9. Piegl, L., Tiller, W.: The NURBS book, 2nd edn. Springer, Heidelberg (1997)
10. Adi, D.I.S., Shamsuddin, S.M., Ali, A.: Particle swarm optimization for NURBS curve fitting. In: Proceedings of the Sixth International Conference on Computer Graphics, Imaging and Visualization: New Advances and Trends(CGIV), pp. 259–263 (2009)
11. Jing, Z., Shaowei, F., Hanguo, C.: Optimized NURBS curve and surface modelling using simulated evolution algorithm. In: Second International Workshop on Computer Science and Engineering (WCSE), pp. 435–439 (2009)