International Journal of Advanced Robotic Systems

OPEN ACCESS ARTICLE

A Collision-Free G^2 Continuous Path-Smoothing Algorithm Using Quadratic Polynomial Interpolation

Regular Paper

Seong-Ryong Chang[1] and Uk-Youl Huh[1]*

1 Electrical Engineering Department, Inha University, In-cheon, Republic of Korea
*Corresponding author(s) E-mail: uyhuh@inha.ac.kr

Received 28 April 2014; Accepted 20 September 2014

DOI: 10.5772/59463

Abstract

Most path-planning algorithms are used to obtain a collision-free path without considering continuity. On the other hand, a continuous path is needed for stable movement. In this paper, the searched path was converted into a G^2 continuous path using the modified quadratic polynomial and membership function interpolation algorithm. It is simple, unique and provides a good geometric interpretation. In addition, a collision-checking and improvement algorithm is proposed. The collision-checking algorithm can check the collisions of a smoothed path. If collisions are detected, the collision improvement algorithm modifies the collision path to a collision-free path. The collision improvement algorithm uses a geometric method. This method uses the perpendicular line between a collision position and the collision piecewise linear path. The sub-waypoint is added, and the QPMI algorithm is applied again. As a result, the collision-smoothed path is converted into a collision-free smooth path without changing the continuity.

Keywords Continuous path, Function approximation, Interpolation, Path planning, Path smoothing, Robot motion, Smoothing algorithm, Smooth path, Vehicle navigation

1. Introduction

The goals of path planning are to avoid obstacles and to find a path. The Probabilistic Roadmaps (PRM) [1] and the Rapidly exploring Random Trees (RRT) [2] algorithms are widely used in sample-based planning algorithms. These algorithms generate points and a collision-free linear piecewise path. The points are regarded as the waypoints of the mobile robot's movements. In addition, the collision-free linear piecewise path is considered as a collision-free G^0 continuous path because this path consists only of straight lines. On the other hand, a high continuous path requires curves.

The G^2 continuous path means a continuous velocity and a continuous acceleration of the robot's movements. If the velocity and acceleration are not continuous, slippage and over-actuation can occur, which can affect the robot movements in a real environment. Moreover, if a planned path has a vertex, the robot cannot follow the path while maintaining the velocity at the vertex. Therefore, the low continuous path cannot be an optimal path as regards time and dynamics. As a result, the path must consist of curves.

The continuity is defined in the geometry [3]. A G^0 continuous path is simply connected for all sections.

Sample-based searching algorithms (the PRM [1] and the RRT [4]) construct a G^0 continuous path. A G^1 continuous path matches the first-order differential values at each point. This path shares a common tangent direction and indicates that the robot and vehicle can have a continuous velocity. The G^2 continuous path has the same second-order differential values at each point. This path also shares a common centre of curvature, which means that the robot and the vehicle can move with continuous acceleration. Accordingly, the G^2 continuous path is called the continuous-curvature path because the curvature can be obtained using the first-order differential values and second-order differential values. A G^n continuous path indicates the equality up to the n^{th} differential values at each point.

To apply to a robot or a vehicle, Villagra et al. reported a smooth path and speed planning for smooth autonomous navigation [5]. Yang et al. proposed a continuous-curvature path-smoothing algorithm using cubic Bézier curves with reduced nodes [6]. Komoriya et al. suggested the trajectory design and control of a wheel-type mobile robot using a B-spline [7]. These reports are focused only on creating a smooth path. Therefore, the result of a path-smoothing algorithm can be a collision. The following studies evaluated a path-smoothing algorithm without collision. Laumond described finding a collision-free smooth trajectory [8]. Scheuer and Fraichard reported collision-free and continuous curvature path planning for car-like robots [9]. Ho and Liu suggested collision-free curvature-bounded smooth path planning using composite Bézier curves based on a Voronoi diagram [10]. These studies sought to obtain a collision-free and a smooth path simultaneously. Pan et al. also reported collision-free and smooth trajectory computation in cluttered environments using B-spline curves [11]. They constructed a smooth path from a linear piecewise path. In addition, they provided an example of a collision path from a path-smoothing algorithm, and improved the collision path to create a collision-free path using the proposed algorithm.

The aims of this paper can be divided into three categories. The first was to create a smooth path including the entire waypoint. Huh and Chang reported a path-smoothing algorithm using modified quadratic polynomial and membership function interpolation (QPMI) [12]. This algorithm can generate a path including the entire waypoint with simple calculations. This paper uses the QPMI algorithm to construct a curvature-continuous smooth path. The second aim was to check the collisions of the generated path. Pan et al. described a collision detection algorithm [11]. This paper use Pan's algorithm to the detection of collisions. The third was to improve the collision path to create the collision-free path. This paper proposes a new collision improvement algorithm for the QPMI algorithm. The proposed algorithm can avoid collisions by adding a sub-waypoint. The added waypoints modify the collision path to create a collision-free path.

In the simulation, the linear piecewise path from the PRM algorithm [1] was improved to create the G^2 continuous path using the QPMI algorithm [12]. In addition, the first-order and second-order differential values at each waypoint are shown on the differential value's graphs. These graphs indicate that the robot and the vehicle can follow a smoothed path with a continuous velocity and acceleration. To verify the collision improvement algorithm, a collision path was made from the planned smooth path. The collision path is improved to create a collision-free path using the collision detection and improvement algorithm.

This paper is organized as follows: Section 2 reports the path-smoothing algorithms using the interpolation method and the requirements of the path-smoothing algorithms. Section 3 explains the characteristic of the QPMI algorithm. Section 4 proposes the collision detection and improvement algorithm. Section 5 reports the simulation results. Section 6 presents the conclusions.

2. Path-smoothing algorithm using interpolation

2.1 Collision-free smooth path

An interpolation is a mathematical field of numerical analysis. This method is used to construct new data points between a series of known data points. Many researchers have applied this method to prepare a path for moving a robot or a vehicle.

In path planning, the path must visit the waypoints. If the searching algorithm creates the waypoints, the smoothing algorithm should not alter the waypoints to prevent the mobile robots or vehicle from losing the waypoints. This is the difference between computer graphics and path planning. Many interpolation-based path-smoothing studies have used the method of computer graphics such as B-splines and Bézier curves.

B-spline and Bézier curves require control points to decide the curvature of the curves. If these methods are applied to smooth path planning, some waypoints must be used as a control point or else a new control point will be needed to decide the curvature. The smoothed path does not include the control points.

A sample-based path-searching algorithm produces the waypoints and the robot must visit the waypoints. On the other hand, the robot cannot visit those waypoints used as control points to decide the curvature. In Figure 1, the squares are the searched waypoints and the circles are the control points. The lines are the linear piecewise path, and the dotted lines are the continuous path using the B-spline method. The dotted lines only contact the control points. The control points are variable and the curves can be modified using the position of the control points. Therefore, the smoothed path is not unique, and the B-spline planned path might not visit the entire waypoint. For the movements of a robot or a vehicle, the planned waypoints and

the searched linear piecewise paths using the searching algorithms guarantee a collision-free path. Therefore, if the smoothed path is close to the linear piecewise path, it is less dangerous than the path that does not visit the waypoints. To obtain a collision-free path, the path smoothing algorithm should follow the waypoints and the searched linear piecewise path faithfully.

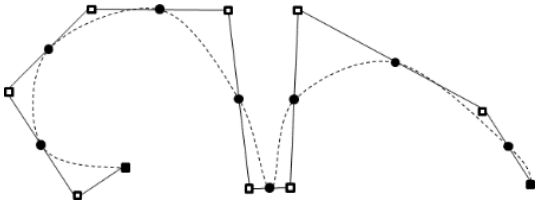


Figure 1. Liner piecewise path using the searching algorithm (line) and the smooth path using a B-spline (dotted line)

The path in Figure 1 needs to be modified. The smoothed path with every waypoint is as follows:

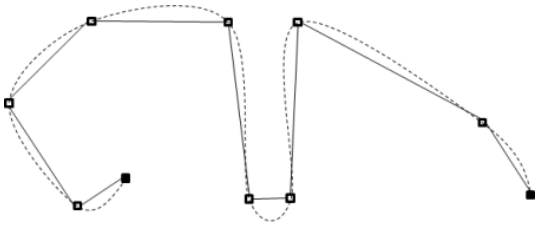


Figure 2. Liner piecewise path using the searching algorithm (line) with the smooth path contacting each waypoint (dotted line)

In Figure 2, the dotted path visits every waypoint. This path is closer to the linear piecewise path than the B-spline-planned path (Figure 1). Therefore, the smooth path, by visiting every waypoint, is closer to the collision-free path.

If the paths in Figures 1 and 2 are placed on narrow passages, the path of Figure 1 can occur as the collision path as follows:

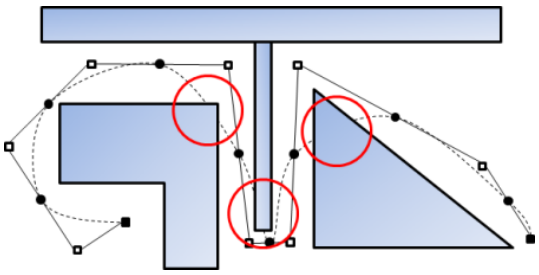


Figure 3. Collisions are created using control points

On the other hand, the path of Figure 2 does not give rise to a collision because this path follows the searching algorithm-planned linear piecewise path.

If the control points are moved, the smoothed path can become the collision-free path (Figure 3). In this case, another algorithm is needed to decide the position of the control point. Figure 4 shows the smooth path in the same environment. The path serves as a smooth path by following the guaranteed collision-free linear piecewise path. The smoothed path should approach the collision-free linear piecewise path as much as possible in order to decrease the probability of collision.

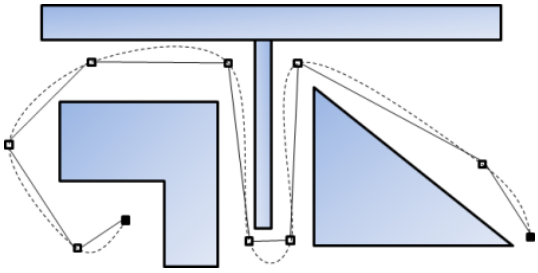


Figure 4. If the smoothed path is close to the searched linear piecewise path, the probability of collision decreases

2.2 Requirements of the path-smoothing algorithm

This paper has the following purposes.

1. The smoothed path must contain the entire waypoint.
2. The smoothed path should be closed to the linear piecewise path with continuity.
3. The smoothed path should be able to check the continuity.
4. If the smoothed path has a collision, the collision is detectable and can be improved.
5. Simple calculations and unique solution.
6. Simple geometry interpretation.

Items **1.** and **2.** mark the differences between other studies (e.g., B-splines and Bézier curves) and the proposed method. Item **3.** is the necessary condition of the path-smoothing algorithm. Item **4.** marks the main issue of this paper. This paper proposes a collision detection and improvement algorithm. Items **5.** and **6.** are important for implementing the proposed algorithm for a real system.

3. Modified quadratic polynomial and membership function interpolation

The QPMI algorithm [12] is a simple path-smoothing algorithm. This algorithm was developed to avoid Runge's phenomenon [3] and the weakness of spline interpolation. The QPMI algorithm can construct a G^2 continuous path using just the quadratic polynomials and membership functions. Furthermore, the continuity of the planned path can be checked.

The quadratic polynomial can construct the shortest path for three waypoints with G^2 continuity because it is the minimum-order polynomial that can connect three points for G^2 continuity. In addition, it is a unique solution for three points. The QPMI-planned path consists of quadratic

polynomials. Therefore, the planned path is the shortest G^2 continuous and unique path with the given waypoints. Moreover, every waypoint is included in the planned path, unlike other algorithms.

The QPMI algorithm does not require the trigonometric functions or a high-order function to create a G^2 continuous path. Therefore, it has a simple calculation. Additionally, the proposed algorithm can provide differential values, the curvature and the heading angle of the planned smooth path. These data can be used in designing the control algorithm.

Huh and Chang [12], however, did not prove the following two lemmas: the first is that the QPMI algorithm has a unique real number solution; and the second is that the continuity of the QPMI algorithm-planned path is decided by the continuity of each axis. This section will prove these two lemmas.

3.1 Unique real number solution of the QPMI algorithm

Lemma 1: The QPMI algorithm has a unique solution in the real number field.

Proof 1: The QPMI algorithm-planned smooth path $\mathbf{P}:(x(u),\ y(u))$ is defined as follows:

$$x(u) = a_x u^2 + b_x u + c_x \tag{1}$$

$$y(u) = a_y u^2 + b_y u + c_y \tag{2}$$

$x(u)$ and $y(u)$ express the variations in the x and y axes. The parameter u was defined as:

$$\begin{aligned} u_1 &= 0 \\ u_n &= \sum_{n=2}^{m} \sqrt{(x_n - x_{n-1})^2 + (y_n - y_{n-1})^2} \end{aligned} \tag{3}$$

The parameter n is the visiting order of the waypoints. Equations (1) and (2) can be obtained using equations (4) and (5):

$$\begin{pmatrix} a_{x_{n-1}} \\ b_{x_n} \\ c_{x_{n+1}} \end{pmatrix} = \begin{pmatrix} u_{n-1}^2 & u_{n-1} & 1 \\ u_n^2 & u_n & 1 \\ u_{n+1}^2 & u_{n+1} & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} x_n(u_{n-1}) \\ x_n(u_n) \\ x_n(u_{n+1}) \end{pmatrix} \tag{4}$$

$$\begin{pmatrix} a_{y_{n-1}} \\ b_{y_n} \\ c_{y_{n+1}} \end{pmatrix} = \begin{pmatrix} u_{n-1}^2 & u_{n-1} & 1 \\ u_n^2 & u_n & 1 \\ u_{n+1}^2 & u_{n+1} & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} y_n(u_{n-1}) \\ y_n(u_n) \\ y_n(u_{n+1}) \end{pmatrix} \tag{5}$$

$(2 \le n \le m - 1)$

The parameter u is defined as follows:

$$u_{n-1} < u_n < u_{n+1} \text{ and } u_{n-1},\ u_n,\ u_{n+1} \ge 0 \tag{6}$$

$$U_{n-1}^{n+1} = \begin{pmatrix} u_{n-1}^2 & u_{n-1} & 1 \\ u_n^2 & u_n & 1 \\ u_{n+1}^2 & u_{n+1} & 1 \end{pmatrix} \tag{7}$$

The parameter u is a cumulative value, and is an increasing function. To obtain the inverse matrix of the parameter u, the determinant value should not be zero. The determinant of the parameter u can be obtained as follows:

$$\begin{aligned} \det\left(U_{n-1}^{n+1}\right) = &\left(u_{n-1}^2 \cdot u_n + u_{n+1}^2 \cdot u_{n-1} + u_n^2 \cdot u_{n+1}\right) \\ &- \left(u_{n+1}^2 \cdot u_n + u_n^2 \cdot u_{n-1} + u_{n-1}^2 \cdot u_{n+1}\right) \end{aligned} \tag{8}$$

$$\det\left(U_{n-1}^{n+1}\right) \ne 0 \tag{9}$$

Therefore, equation (9) can be solved as follows:

$$u_{n-1}^2 \cdot (u_n - u_{n+1}) + u_{n+1}^2 \cdot (u_{n-1} - u_n) + u_n^2 \cdot (u_{n+1} - u_{n-1}) \ne 0 \tag{10}$$

if $u_{n-1} = 0$. In this case, equation (10) is changed to:

$$u_{n+1}^2 \cdot (-u_n) + u_n^2 \cdot (u_{n+1}) \ne 0 \tag{11}$$

If $u_{n-1} = 0$, then u_n and u_{n+1} are not zero according to (6). Therefore, equation (10) cannot be zero.

The second case is the case of $u_{n-1} > 0$. If equation (10) is zero, equations (12) or (13) is satisfied.

$$(u_n - u_{n+1}) = (u_{n-1} - u_n) = (u_{n+1} - u_{n-1}) = 0 \tag{12}$$

$$u_{n-1} = u_n = u_{n+1} = 0 \tag{13}$$

On the other hand, in equation (6), u_{n-1}, u_n and u_{n+1} are not equal. Therefore, equation (10) cannot be zero. In addition, equation (13) is not satisfied because u_n is not zero. Therefore, the determinant value is not zero in any case. As a result, Lemma 1 is proven. ■

3.2 Continuity of the QPMI algorithm

The continuity of the path from the QPMI algorithm is determined by the continuity of each axis. The QPMI algorithm uses the parametric method. This method separates each axis using the parameter u. To check the continuity of the planned path, the differential values of $x(u)$ and $y(u)$ are continuous.

Lemma 2: If $x(u)$ and $y(u)$ are continuous, $\mathbf{P}:(x(u),\ y(u))$ are continuous. In addition, the continuities of $x(u)$, $y(u)$ and the path are equal.

Proof 2: The continuity is determined by the matching of the differential values at each waypoint. If $x(u)$ is G^2 continuous, $dx(u)$ and $d^2x(u)$ have connected graphs in

the entire section. If $y(u)$ is G^2 continuous, $dy(u)$ and $d^2y(u)$ have connected graphs. The first-order and second-order differential values of $\mathbf{P}:(x(u),\ y(u))$ can be expressed as $\mathbf{P}:(dx(u),\ dy(u))$ and $\mathbf{P}:(d^2x(u),\ d^2y(u))$. Therefore, the differential graphs of $\mathbf{P}$ consist of the differential values of $x(u)$ and $y(u)$. As a result, the continuity of $\mathbf{P}$ is equal to $x(u)$ and $y(u)$. ■

4. Collision detection and improvement algorithms for the smooth path

The piecewise linear path is a collision-free path using the path-searching algorithm. Generally, this path does not require a collision-check. On the other hand, the smoothed path requires a collision-checking process because collisions can occur while constructing a smooth path using the path-smoothing algorithm. Figure 5 presents the case of a collision of the smooth path.

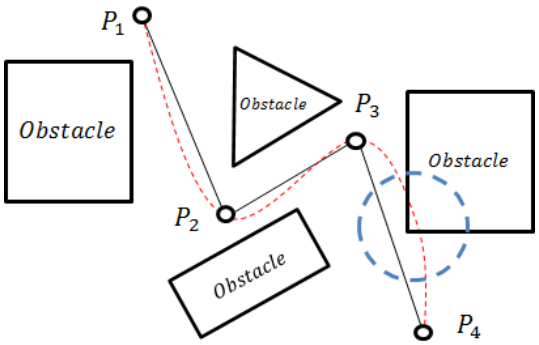


Figure 5. Collision-free linear piecewise path (line) and smoothed path (dotted line). A collision occurred at the dashed circle.

In Figure 5, the line is the linear piecewise path from the path-searching algorithm. The dotted line is the smoothed path using the QPMI algorithm. A collision occurs between P_3 and P_4. The collision-checking algorithm must detect the collision of the smoothed path. In addition, the path should be improved to create a collision-free path.

4.1 Collision-checking algorithm for the QPMI algorithm

The simplest collision-checking algorithm checks the collision of the path by discrete samples. The idea of an 'efficient spline collision detection algorithm' [11] is used in this paper.

Given the smoothed path $\mathbf{P}$: $(X(u),\ Y(u))$, fixed obstacles are represented as B. Equation (14) is the collision-free condition, u_1 is the start point and u_m is the goal point. Parameter m is the number of waypoints, including the start point and the goal point:

$$P(X(u),\ Y(u)) \cap B = \emptyset \ \ for\ every\ u \in \{u_1 : u_m\} \tag{14}$$

If a collision is detected, equation (14) is changed to equation (15):

$$P(X(u),\ Y(u)) \cap B \neq \emptyset \ \ for\ any\ u \in \{u_1 : u_m\} \tag{15}$$

The proposed collision-checking algorithm is as follows: let $d(P(u),\ B)$ be the distance between $P(u)$ and B. φ is a boundary of the robot. The bound of the robot can be defined as the size of the robot or the sensing area of the robot. If a collision is detected, $\rho < 1$. Equation (16) is a collision-checking equation:

$$\rho(u) = \frac{d(P(u),\ B)}{\varphi} \tag{16}$$

In equation (16), the bound of the robot φ should be smaller than the distance between $P(u)$ and B. Algorithm 1 is described as a collision-checking algorithm.

Algorithm 1: Collision-checking algorithm

Input : Smooth path ***P*** and obstacles ***B***.
Output: Decision of Collision-free or Collision position
begin
  for $u = u_1$ *to* u_m : Δu **do**
    Compute current distance $d(P(u), B)$
    Estimate the motion bound φ
    Compute $\rho(u)$ for collision checking
    if $(\rho(u) \leq 1)$ **then** // collision is detected
      return $u_c = u$ // u_c is collision position
  return collision-free

This study used Algorithm 1 to check the collision; u_c is the collision position. If the path is not collision-free, the collision position u_c is sent to Algorithm 2 for improving the collision of the path. If this path is a collision-free path, Algorithm 1 returns the collision-free path.

4.2 Path improvement for the collision-free path

If a collision occurs, a path improvement algorithm is needed. In Figure 6, a collision has occurred in a P_{34} section. The aim of the collision improvement algorithm is to correct the smoothed path closer to the linear piecewise path in a collision section, because the linear piecewise path already guarantees the collision-free path using the path-searching algorithm. Therefore, the smoothed path can be a collision-free path which is as close as the linear piecewise path. Figure 6 is an improvement on the smoothed path using the smoothed path moved to the linear piecewise path.

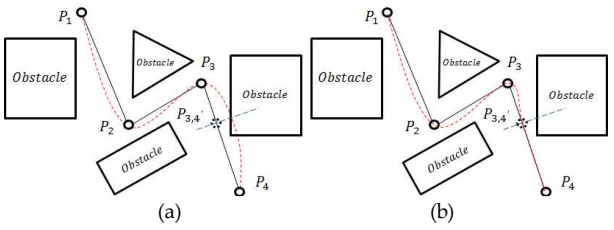


Figure 6. (a) Collision of the smoothed path; (b) improved collision path

In Figure 6, the first step was to finding a perpendicular line between the collision position and the linear piecewise path. The blue dashed line is the perpendicular line. The second step is to create a sub-waypoint on a crossing position of the linear piecewise path and the perpendicular line. The crossing position is defined as sub-waypoint $P_{3,4}'$. The third step is to reconstruct the smooth path including the sub-waypoint using the QPMI algorithm. This process is described as Algorithm 2.

Algorithm 2: Collision improvement algorithm

Input : Smooth path $\boldsymbol{P}$ and Collision position u_c
Output: Collision Improved path $\boldsymbol{P_{new}}$
begin
  $P_c(x_c, y_c) = P(X(u_c), Y(u_c))$ // collision position
  find collision section $P_{n,n+1}$
  create perpendicular line //collision position to the linear path
  define sub-waypoint $P_{n,n+1}'$ on the linear piecewise path
  build collision improved smooth path include sub-waypoint $P_{n,n+1}'$ using QPMI algorithm
  return $\boldsymbol{P_{new}}$
end

Algorithm 3 is the collision-checking and improvement algorithm using Algorithms 1 and 2.

Algorithm 3: Collision-free path-smoothing algorithm

Input : Linear piecewise path
Output: Collision-free path or Decision of collision-free
begin
  build smooth path using QPMI algorithm
  do Algorithm 1 // collision-checking
  if(collision-free) **then**
    return collision-free path $\boldsymbol{P_s}$
  else
    while checking_count ≤ checking_count_max
      do Algorithm 2 // collision improvement
      do Algorithm 1 // collision-checking
      if(collision-free) **then**
        return collision-free path $\boldsymbol{P_s}$
      else
        ++checking_count
    if(checking_count > checking_count_max) **then**
      return decision (path has collision)
end

Algorithm 3 can be used for collision checking and improving the smooth path. Algorithm 1 checks the collision and Algorithm 2 improves the collision path to create the collision-free path. These two algorithms are combined as Algorithm 3. In this paper, Algorithm 3 is called the 'Collision-free Checking and Improvement' (CCI) algorithm.

The maximum checking count value is the number of collision checks. In the case, where it is impossible to find a collision-free path using the proposed algorithm, Algorithm 3 can be an infinite loop. To avoid an infinite loop, the checking count's maximum value needs to be checked.

5. Simulations

In this section, the linear piecewise path is converted to the G^2 continuous path using the QPMI algorithm. In addition, the proposed CCI algorithm is applied to this smooth path.

A simulation map has a narrow passage that makes it collide with the obstacle. The PRM algorithm is used to obtain the linear piecewise path. This algorithm is implemented using the MATLAB toolbox of [13]. Figure 7 presents the simulation map, and Figure 8 shows the result of the PRM algorithm.

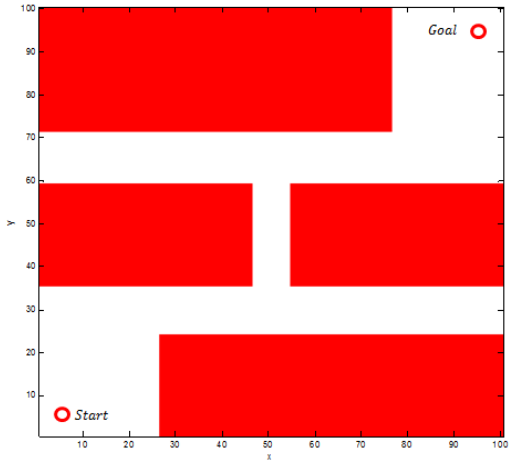


Figure 7. Simulation map with the start point and goal point. The red blocks are obstacles.

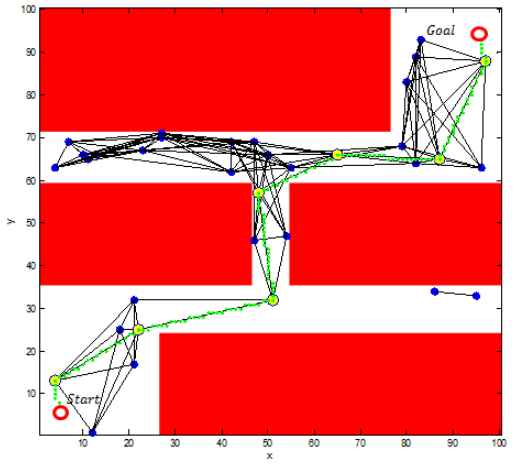


Figure 8. Result of the PRM algorithm. The green line is the searched linear piecewise path using the PRM algorithm.

The searched collision-free waypoints are as follows:

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
x	5	4	22	51	48	65	87	97	95
y	5	13	25	32	57	66	65	88	95

Table 1. Searched position using the PRM algorithm. P_1 is the start point and P_9 is the goal point.

5.1 Path smoothing and analysis using the QPMI algorithm

The QPMI algorithm requires a distance parameter u. The set of u is as follows using equation (3).

u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9
0	8.06	29.7	59.53	84.71	103.94	125.97	151.05	158.33

Table 2. Set of parameter u

Equations (4) and (5) construct the quadratic polynomials. These polynomials are shown in Figure 9 (a). In addition, Figure 9 (b) presents the result of the QPMI that combined the quadratic polynomial and membership function.

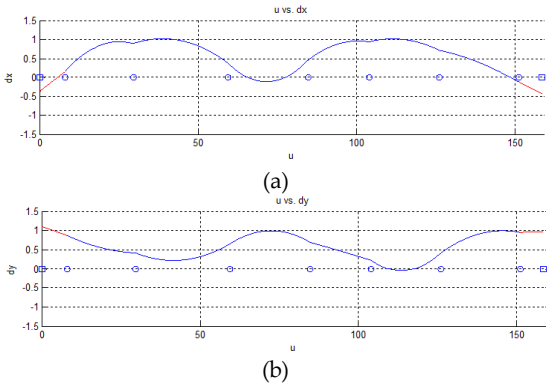


Figure 9. Graph of the parametric quadratic polynomials (a), and a merged graph (red line) using the membership function (b)

Figure 10 presents the final result.

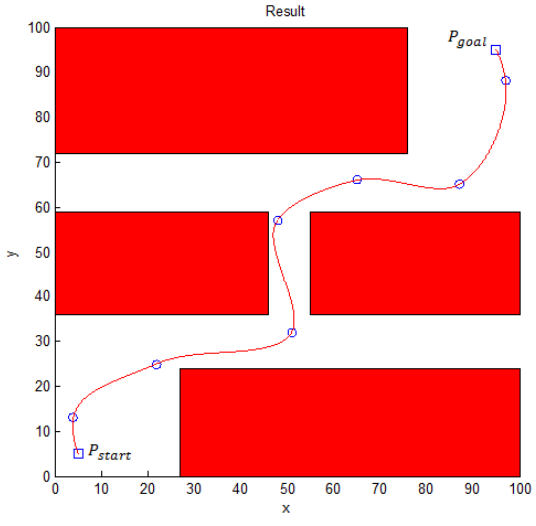


Figure 10. G^2 continuous path using the QPMI algorithm

The QPMI algorithm proffers variations of x and y and the differential values of them. In addition, the curvature and the heading angle can be obtained.

Figure 11 shows the graph of x and y.

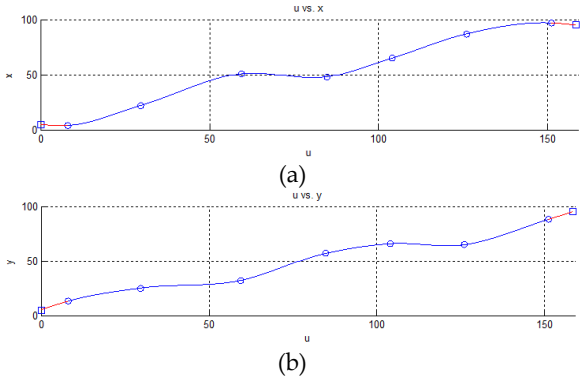


Figure 11. (a) graph of u vs. x; (b) graph of u vs. y

Figure 12 presents the first-order differential values of x and y.

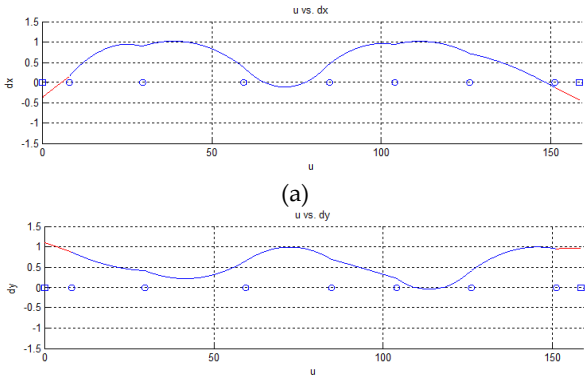


Figure 12. (a) graph of u vs. dx; (b) graph of u vs. dy

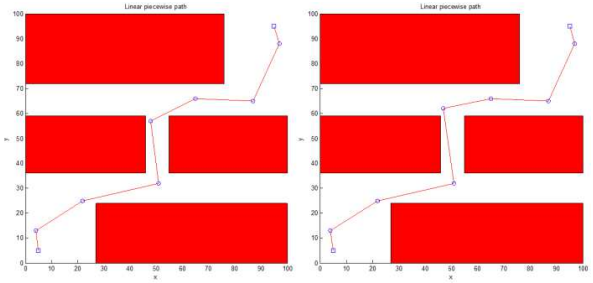
In Figure 12, the graph connects the entire section. This graph indicates that the smoothed path is the G^1 continuous path.

The condition of the G^2 continuous path is that the second-order differential values should be contacted at each waypoint. To check the G^2 continuous path, the graph of the second-order differential values was obtained. Figure 13 presents a graph of the second-order differential values.

Figure 13 show that the smoothed path is the G^2 continuous path.

The curvature graph can be obtained, as shown in Figure 14. The graph is as follows:

The G^2 continuous path is also called the 'curvature continuous path'. In this simulation, the first-order and second-order differential values are matched at each waypoint. These values construct the continuous curvature. In Figure 14, the curvature graph is continuous.



(a) (b)

Figure 13. (a) graph of u vs. d^2x; (b) graph of u vs. d^2y

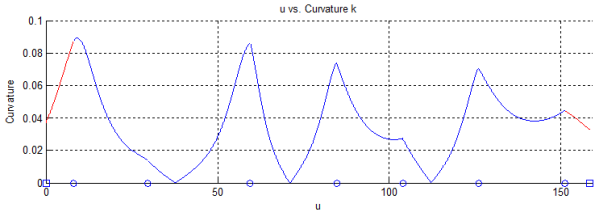


Figure 14. Graph of u vs. curvature

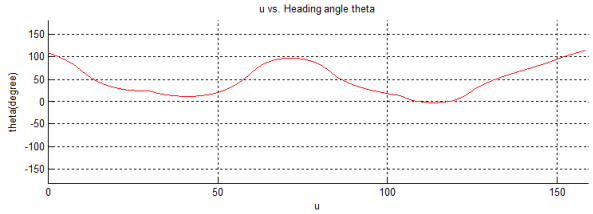


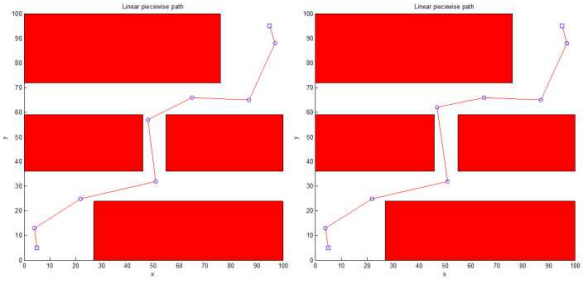
Figure 15. Graph of u vs. the heading angle

Figure 15 shows the heading angle graph. This graph has a continuous form. This means that the path can follow with continuous movement.

5.2 Simulation of the CCI algorithm

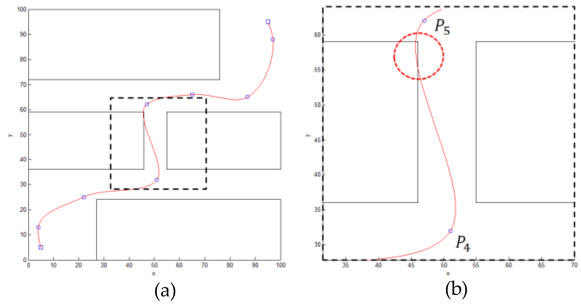
In section 5.1, the smoothed path proved the G^2 continuous path. In addition, the path was analysed using the QPMI algorithm. On the other hand, the searched waypoints can be placed at an obscure position in a real situation. In this case, the smoothed path cannot guarantee a collision-free path despite the linear piecewise path being a collision-free path. In this section, Figure 10 was modified to create the collision path for the collision detection and improvement simulation. This simulation assumes that the searched linear piecewise path is a collision-free path, but the smoothed path sees a collision. If P_5 is moved, the liner piecewise path can be modified as Figure 16 (b).

The QPMI algorithm was applied to the modified path and the path was changed to the smoothed path. On the other hand, the smoothed path has a collision despite the linear piecewise path being the collision-free path. Figure 17 shows this phenomenon.



(a) (b)

Figure 16. Original path (a) and modified path (b). Both paths are the collision-free path



(a) (b)

Figure 17. Collisions occur on the smoothed path. The red circle is the collision position.

To improve the collisions, the CCI algorithm was applied. The first collision position was P_c (46, 55.25) when u_c=81.5.

The linear piecewise equation of P_4 to P_5 is expressed as (17) and an equation of the perpendicular line is shown in equation (18):

$$y = -7.5x + 414.5 \tag{17}$$

$$y = 0.133x + 49.117 \tag{18}$$

The sub-waypoint can be obtained using equations (17) and (18). The sub-waypoint was $P_{4,5}$'(47.869,55.484). Figure 18 shows the collision position, the perpendicular line and the sub-waypoint.

Finally, the QPMI algorithm was applied including the sub-waypoint. Figure 19 demonstrates the collision-improved path. The red-dashed path is a collision-smooth path. After applying the CCI algorithm, the collision problem is solved as the blue path. This path includes P_4, P_{45}, P_5 and P_6.

Figure 20 presents the final result of this simulation. The collision position can be avoided using the path that is moved to the sub-waypoint. This path can be decided as the collision-free path using the CCI algorithm.

In this simulation, the QPMI algorithm is demonstrated. The map has a narrow passage. The PRM algorithm searches for the collision-free linear piecewise path with low continuity. The QPMI algorithm constructs the smooth

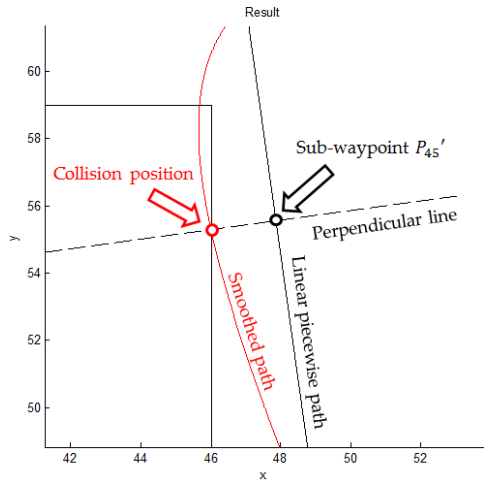


Figure 18. The collision improvement algorithm is shown. The smoothed path makes the collisions. The perpendicular line is constructed between the linear piecewise path and the collision position. A cross-position of the perpendicular line and the linear piecewise path is decided to create the sub-waypoint.

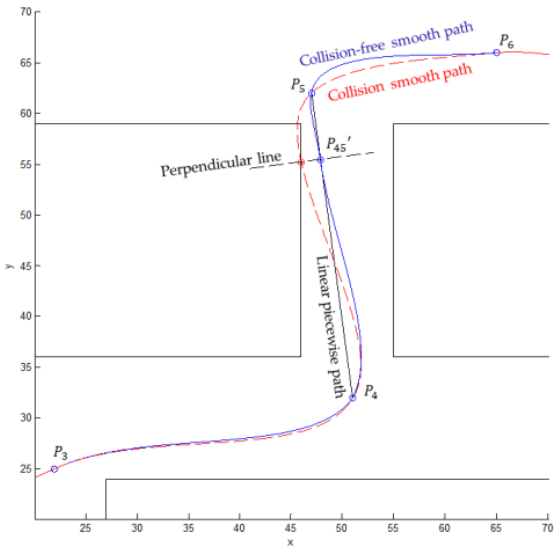


Figure 19. Collision path (red-dashed line) and collision-free path (blue line). The collision-improved path contains the sub-waypoint. As a result, the path is moved to the collision-free linear piecewise path.

path and checks the continuity. As a result, the linear piecewise path is converted to a G^2 continuous path.

To prove the CCI algorithm, the smooth path is modified to create the collision path. The first step is the detection of the collision position. In the next step, a perpendicular line is constructed between the collision-free linear piecewise path and the collision position. The sub-waypoint is decided at the cross-position on the collision-free linear piecewise path and the perpendicular line. Finally, the QPMI algorithm is applied again to create a smooth, collision-free path. The collision-free G^2 continuous path can then be obtained.

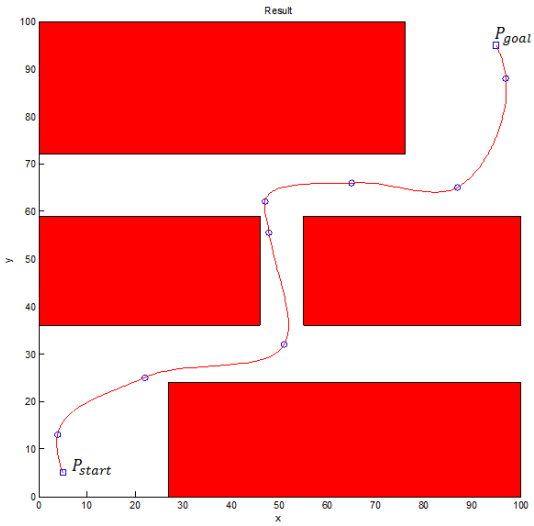


Figure 20. Collision-improved path using the CCI algorithm

6. Conclusions

Most search algorithms do not consider the continuity of the path. The QPMI algorithm aims to construct a continuous path from a searched, collision-free linear piecewise path. The general methods for constructing a continuous path is the B-spline and the Bézier curve, which are widely used in computer graphics. On the other hand, these do not contain all the waypoints because some waypoints should be used to create the control points that decide the curvature.

In this study, the QPMI algorithm was used to create the smooth path. This algorithm provides the G^2 continuous path-smoothing algorithm, the differential values, the curvature and the heading angle. These data can be used to design the control algorithm of the mobile robots or vehicles. Furthermore, the result was unique and the calculations are simple because this algorithm used only the quadratic polynomials, without trigonometric functions or high-order polynomials. Therefore, the calculations can be simple. These features do not require high-performance hardware. In addition, unlike some other path-smoothing algorithms, the QPMI algorithm constructs a smooth path containing all the waypoints. During the path planning, visiting the waypoints is important.

The QPMI algorithm was required to prove two lemmas. The first is that the planned path is unique. The second concerns the continuity of the planned path. This paper proved these two lemmas.

The searched path using the searching algorithms is a collision-free path. Although this path is a collision-free path, the smoothed path cannot be decided by the collision-free path. The CCI algorithm was proposed to check the collisions in the smoothed path. The CCI algorithm can detect the collision position using a simple method. If the smoothed path has a collision, it can be improved using this algorithm by approaching the smoothed path to the linear

piecewise path. The perpendicular line including the collision position and the linear piecewise path decides the sub-waypoint for moving the collision-smooth path to create the linear piecewise path. If a sub-waypoint is obtained, the QPMI algorithm can be applied again. As a result, the collision-smooth path is improved to create a collision-free smooth path maintaining continuity.

The goals of this paper were archived. The QPMI algorithm provided the path containing the entire waypoint, the smoothed path was approached to the linear piecewise path, the continuity checking was possible, the collision-checking and improving algorithm was proposed. the proposed algorithms are simple, unique and having simple geometry interpretation.

In this paper, the QPMI and CCI algorithm were applied to the 2D plane. These can be used for a mobile robot, vehicle, a game algorithm and for computer graphics without complex calculations. In addition, these algorithms can be expanded to a 3D space. In this case, it will be possible to use an aerial robot and aircraft to create a G^2 continuous trajectory for visiting all the waypoints.

7. Acknowledgements

This study was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (no. 2012-0005564).

8. . References

[1] 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,* vol. 12, pp. 566-580, 1996.

[2] S. M. LaValle and J. J. Kuffner Jr, "Rapidly-exploring random trees: Progress and prospects," 2000.

[3] G. E. Farin, *Curves and surfaces for CAGD [electronic resource]: a practical guide*: Morgan Kaufmann, 2002.

[4] M. Abramowitz and I. A. Stegun, "Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. National Bureau of Standards Applied Mathematics Series 55. Tenth Printing," 1972.

[5] J. Villagra, V. Milanés, J. P. Rastelli, J. Godoy, and E. Onieva, "Path and speed planning for smooth autonomous navigation," in *IEEE Intelligent Vehicles Symposium*, 2012.

[6] K. Yang, D. Jung, and S. Sukkarieh, "Continuous curvature path-smoothing algorithm using cubic B zier spiral curves for non-holonomic robots," *Advanced Robotics,* vol. 27, pp. 247-258, 2013.

[7] K. Komoriya and K. Tanie, "Trajectory design and control of a wheel-type mobile robot using B-spline curve," in *Intelligent Robots and Systems' 89. The Autonomous Mobile Robots and Its Applications. IROS'89. Proceedings., IEEE/RSJ International Workshop on*, 1989, pp. 398-405.

[8] J.-P. Laumond, "Finding Collision-Free Smooth Trajectories for a Non-Holonomic Mobile Robot," in *IJCAI*, 1987, pp. 1120-1123.

[9] A. Scheuer and T. Fraichard, "Collision-free and continuous-curvature path planning for car-like robots," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, 1997, pp. 867-873.

[10] Y.-J. Ho and J.-S. Liu, "Collision-free curvature-bounded smooth path planning using composite Bezier curve based on Voronoi diagram," in *Computational Intelligence in Robotics and Automation (CIRA), 2009 IEEE International Symposium on*, 2009, pp. 463-468.

[11] J. Pan, L. Zhang, D. Manocha, and U. C. Hill, "Collision-free and curvature-continuous path smoothing in cluttered environments," *Robotics: Science and Systems VII,* vol. 17, p. 233, 2012.

[12] U.-Y. Huh and S.-R. Chang, "A G^2 Continuous Path-smoothing algorithm Using Modified Quadratic Polynomial Interpolation," *International Journal of Advanced Robotic Systems, 11:25,*2014.

[13] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB* vol. 73: Springer, 2011.