Malaysian Journal of Mathematical Sciences 15(1): 45–59 (2021)

MALAYSIAN JOURNAL OF MATHEMATICAL SCIENCES

Journal homepage: http://einspem.upm.edu.my/journal

Multi-Objectives Path Planning using Bézier Curve

Safaruddin, M. S. and Misro, M. Y. *

School of Mathematical Sciences, Universiti Sains Malaysia, 11800 Gelugor, Pulau Pinang, Malaysia

E-mail: yushalify@usm.my
* *Corresponding author*

Received: 23 April 2020
Accepted: 20 December 2020

ABSTRACT

This study presents Dijkstra's algorithm using permutation method. A directed graph that contains nodes with a specific initial and end position will be connected at every edge with their own weights. The constructed algorithm plays an important role to determine the shortest path by using the concepts of Dijkstra's algorithm. All the nodes in the shortest paths will be used as the control points of Bézier curves during its construction. Then, the shortest path will be interpolated using Bézier curves with control points to provide a smooth path planning curve.

Keywords: Dijkstra's algorithm, Bézier curve, permutation, shortest path, directed graph.

1. Introduction

Path planning is often used in robotics, whether in online or offline environments by Yin et al. (2017). It can be divided into two types, which are global and local path planning. Both path plannings have two things in common, which are the initial position and the final destination. The differences between global and local path planning are the navigation of robots from the initial position in the process of reaching the goal and avoiding collision. Obstacles are categorised into two, which are static and dynamic obstacles (Pandey and Parhi, 2017).

Static obstacles refer to non-moving obstacles where the robotic environment is known in advance with the help of localization maps before reaching its final destination. This localization map and non-moving obstacles are described as the classical approach by Raja and Pugazhenthi (2012). Meanwhile, dynamic obstacles occur when we do not have any specific information about the obstacles in between the starting point and the ending point. Hence, by using evolutionary optimization algorithm, the path can be executed on a real-time basis, which will give an edge over the classical approach. Robots can be programmed to deal with obstacles via several methods. The methods that are frequently used are Rapidly Exploring Random Tree (RRT) by Bircher et al. (2017) and Probabilistic Road Map (PRM) by Akbaripour and Masehian (2017).

These methods can generate polygonal networks of the collision-free path. Raja and Pugazhenthi (2012) stated that the paths can be calculated by considering multi-objectives, such as the shortest distance and the time of travel. The robotic movements can be polygonal-wise or a consecutive combination of connecting straight lines. Therefore, the optimum polygonal path planning needs to be converted into a curvy path to reduce the sharp robotic movement, thus reducing the traveling time while robotics manoeuvre from one place to another.

Bézier curves are one of the major discoveries in Computer Aided Geometric Design (Misro et al., 2017a,d). Bézier curves play an important role to determine the curvy path from the initial nodes to the ending nodes. Recently, there are a lot of research on path planning using Bézier curve as in Cimurs et al. (2017) and Zhang et al. (2015). Dijkstra's algorithm helps to find the shortest paths between the source and the final nodes on a graph. It was created by a Dutch computer scientist, Edsger Wybe Dijkstra (Schrijver, 2012). Previous study had constructed path planning using Bézier curve and Dijkstra's Algorithm; in Zhou et al. (2011), piecewise linear path planning is implemented based on Voronoi's diagram and Dijkstra's algorithm before smoothing the path using curve connecting procedure. In this research, the optimum shortest curvy path method can be achieved after finding all the possible paths on a graph using permutation method and by determining all the nodes (control points) for Bézier curves. This study also aims to interpolate a smooth curve with the calculated shortest path, and it will involve a directed graph with non-negative weights.

2. Methodology

2.1 Directed Graph and Non-Negative Weight

Let $G = (V, E)$ be a graph, where V is a set of all the nodes (or vertices) and E is a set of all the edges. Between the edges, there are non-negative weights that exist in the graph. This paper aims to find the shortest path by using the total number of weights for a single path. Thus, an algorithm must be created to calculate the total number of weights on a graph. Therefore, the concept from Dijkstra's algorithm is adapted with non-negative weights.

2.2 Permutation

Permutation is a way of arranging the order of different elements. Notation of $P(s, r)$ is used to represent the number of ways to arrange r elements from a set of s elements. If s is a positive integer and r is an integer with $1 \leq r \leq n$, then there are

$$P(s, r) = s(s-1)(s-2)...(s-r+1), \tag{1}$$

where r-permutation of a set with s distinct elements (Rosen and Krithivasan, 2012). The number of different permutations of s objects, where there are s_1 indistinguishable objects of type 1, s_2 indistinguishable objects of type 2, and s_j indistinguishable objects of type j, is

$$\frac{s!}{s_1!s_2!...s_j!}. \tag{2}$$

Noted that, for the 1^{st} until the r^{th} position, the first element in the 1^{st} position can be chosen in n number of ways. For the 2^{nd} position, since one of the numbers had been chosen in the 1^{st} position, thus the 2^{nd} position can be chosen in $s-1$ number of ways. This will go on until the last position, which is r^{th} term where there are $(s-(r-1))$ number of ways to choose. By multiplying all the terms, we have

$$P(s,r)=s(s-1)(s-2)...(s-r+1).$$

The number of permutations can be determined when the s_1 objects of type one can be placed among the s positions in $C(s,s_1)$ ways, leaving $s-s_1$ positions free. Then, the objects of type two can be placed in $C(s-s_1,s_2)$ ways, leaving $s-s_1-s_2$ positions free. Continue placing the objects of type three, ..., type $j-1$, until the last stage, s_j objects of type j can be placed in $C(s-s_1-s_2-...-s_j-1,s_j)$ ways. Hence, by the product rule, the total number of different permutations are

$$C(s,s_1)C(s-s_1,s_2)...C(s-s_1-s_2-...-s_j-1,s_j) \tag{3}$$

$$=\frac{s!}{s_1!(s-s_1)!}\frac{(s-s_1)!}{s_2!(s-s_1-s_2)}...\frac{(s-s_1-...s_{j-1})!}{s_j!0!} \tag{4}$$

$$=\frac{s!}{s_1!s_2!...s_j!}.$$

2.3 Bézier Curve

Given a set of control points $P_0,P_1,...,P_n$, the corresponding Bézier curve is defined as

$$z(t)=\sum_{i=0}^{n}P_iB_i^n(t), \qquad 0\leq t\leq 1, \tag{5}$$

where $B_i^n(t)$ is the Bernstein polynomial and P_i is the i^{th} vectors of control points. The Bernstein polynomials of degree n are defined explicitly by

$$B_i^n(t)=\left\{\begin{array}{ll}\frac{n!}{i!(n-1)!}t^i(1-t)^{n-i}, & \mathit{if}\, 0\leq i\leq n,\\ 0, & \text{otherwise},\end{array}\right\} \tag{6}$$

where $i=0,1,...,n$ and the binomial coefficients are denoted by $\binom{n}{i}$. Some of the properties of Bernstein polynomial are given as follows:

Partition of Unity
According to Farin (2002), for any value of t, the sum of the Bernstein polynomial is equal to one.

$$z(t)=\sum_{i=0}^{n}B_i^n(t)=1. \tag{7}$$

Non-Negativity

The Bernstein polynomials are non-negative on the interval [0,1], it is defined in Marsh (2006) to be

$$B_i^n(t) \geq 0, \qquad t \in [0,1]. \tag{8}$$

The non-negativity is rather obvious from the definition of the Bernstein polynomials. Bézier curve also has several properties that needs to be satisfied in order to construct a smooth curve. Some of the properties of Bézier curve that we want to highlight in this research are:

Endpoint Interpolation

A Bézier curve always passes through the first and the last control points. From Equation (5), for $t = 0$ and $t = 1$, the Bézier curve, $z(t)$ will be

$$\begin{aligned} z(0) = & \quad P_0, \\ z(1) = & \quad P_n. \end{aligned} \tag{9}$$

Convex Hull Property

Figure 1 shows a Bézier curve that is constructed using 5 different nodes (control points). The blue dotted line is the control polygon. The Bézier curve lies inside the control polygon, therefore this Bézier curve satisfies the convex hull property. As introduced in Misro et al. (2019a), the curvature can be used to analyse how the curves behave for each curve interpolation. The radius of curvature r is given by $r = 1/\kappa$. When r approaches ∞, the curve becomes a straight line as $\kappa(t) = 0$ where $t \in [0,1]$. The curvature $\kappa(t)$ of the curve can be evaluated as

$$\kappa(t) = \frac{z'(t) \times z''(t)}{\|z'(t)\|^3}. \tag{10}$$

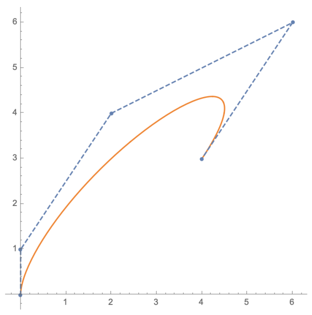

Figure 1: Convex hull of Bézier curve.

3. Path Planning Algorithm

For a given start and endpoint, the objective is to produce the shortest path using permutation method and to interpolate the shortest path to achieve a smooth Bézier curve. This algorithm starts by finding all the possible paths of the directed graph. The shortest path is determined by choosing the optimum number of total weights. Weights in each path are represented by distance and time of travel between each node.

This algorithm has four steps in total. The first step is to design a directed graph that resembles movement environments, which consists of several nodes and its weights that will be stored in a matrix form. The second step is to find all possible paths using permutation method. The third step is to find the shortest path by calculating the number of weights for each path. Lastly, the fourth step is to interpolate a smooth Bézier curve with all the control points and to determine a smooth path planning using curvature analysis.

3.1 First Step: Produce a Directed Graph with several Nodes and its Weights

For this directed graph, 16 nodes were set as the dimension size of a robotic environment. All 16 nodes with weights will be stored inside a 4 x 4 matrix. Note that every edge inside the graph is connected with every node and contains its weight that are randomly generated. Multi-objective had been considered in this research as follows:

1. The distance between each node.
2. and, the time travel between each node.

If node 1 and node 2 are connected and contains value 2 for distance and value 3 for time travel, so the value will be in this form

$$matrix[1,2] = \{2,3\}. \qquad (11)$$

Thus, the directed graph is as shown in Figure 2 below with node 1(bottom right) as the starting node and node 16 (top left) as the ending node.

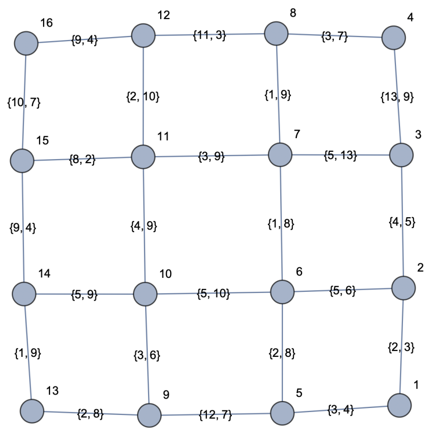

Figure 2: The directed graph with a total of 16 nodes and its own weight on every edge connected.

3.2 Second Step: Finding All the Possible Paths of the Directed Graph

The second step will receive the number of nodes visited in the North and West direction. This direction is arbitrarily chosen and can be extended into North, South, East, and West, but the number of nodes visited for every possible path will not be the same. Therefore, for this study, two directions are set. Once the algorithm receives the number of nodes visited for each direction, the two sets of North and West directions will be stored. For example, if the number of nodes visited in North and West is 3, then the direction sets will be given as follows

$$\{w, w, w\} \, and \, \{n, n, n\} \tag{12}$$

where n is the North direction and w is the West direction. Then, the algorithm will combine these two sets into a new set as shown below

$$\{w, w, w, n, n, n\} \, . \tag{13}$$

Next, the algorithm will permute Equation (12) to get all the possible paths in North and West direction.

3.3 Third Step: Finding the Shortest Path by Calculating the Lowest Total Number of Weights

After retrieving all the possible paths, the total number of weights and the nodes visited for each path will be recorded. Recall that every weight has been stored in a matrix; this algorithm will call every weight based on the nodes visited in North and West direction for all possible paths in a loop function. The loop function will check the direction for each permuted path and extract the values inside the matrix. Then, the number of weights for each node visited will be totaled up. Once all the total number of weights for each possible path had been calculated, the algorithm will find the lowest total number of weights by evaluating the minimum total number of weights. Thus, the lowest total number of weights for the possible path is the shortest path in the directed graph.

3.4 Fourth Step: Interpolate a Smooth Bézier Curve with all the Control Points

After the shortest path had been identified with the respective nodes visited, the algorithm will set all the visited nodes as control points. Then, by using Equation (6), the shortest path can be interpolated into a smooth Bézier curve from the initial node to the goal node. The curvature profile can be computed using Equation (10). The value of the curvature for each curve can be compared to obtain the best path planning.

4. Result and Discussion

Recall that there are distance and time travelled as multi-objectives in the directed graph. We separate the results into two categories

1. One Dimensional Weight, and
2. Two Dimensional Weights.

4.1 One Dimensional Weight

4.1.1 Shortest Path based on the Total Number of Distance

Figure 3 shows a directed graph constructed with 16 nodes and has its weight on every edge connected (left) and its Bézier curve interpolation (right). The red line in Figure 3 (left) shows the shortest path planning based on the total number of distance. $\{1, 5, 6, 7, 11, 12, 16\}$ show the sequence of nodes visited with the total distance of 20. Bézier curve interpolation can be constructed in Figure 3 (right) using the same nodes as in Figure 3 (left) as its control points. The curve preserves its endpoint interpolation property as in Equation (9). This endpoint property was very useful in order to ensure that the path planning starts and ends at the same points as desired. The blue dotted lines are the convex hull while the blue circles are the control points for the Bézier curve as in Figure 3 (right). This curve lies within the convex hull and the domain for this curve is $t \in [0, 1]$, thus, all the Bernstein polynomials are non-negative. All the Bernstein polynomial are the coefficient of the expression $1 = (t + (1 - t))^n$. Hence, the sum is equal to one and it satisfies the partition of unity.

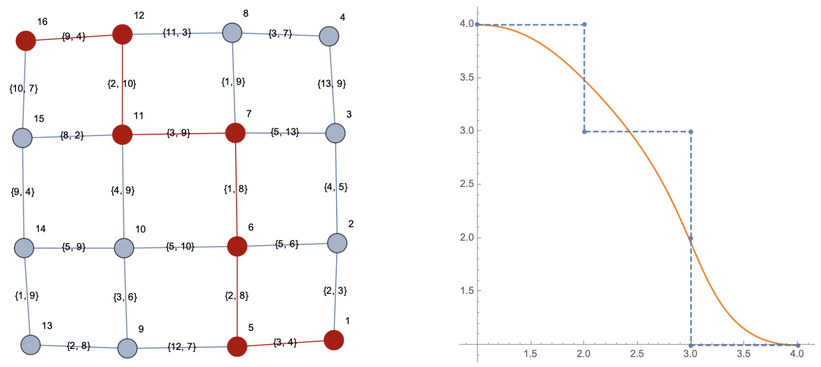

Figure 3: The shortest path (left) and the smooth Bézier curve with minimum total number of distances (right).

4.1.2 Shortest Path based on the Total Number of Time Travelled

Figure 4 (left) shows the shortest path based on the total number of time travelled and Figure 4 (right) shows the interpolation of Bézier curve. The curves are constructed by using the same nodes visited as in Figure 4 (left) which is $\{1, 2, 3, 4, 8, 12, 16\}$ with 34 as the total number of time travelled.

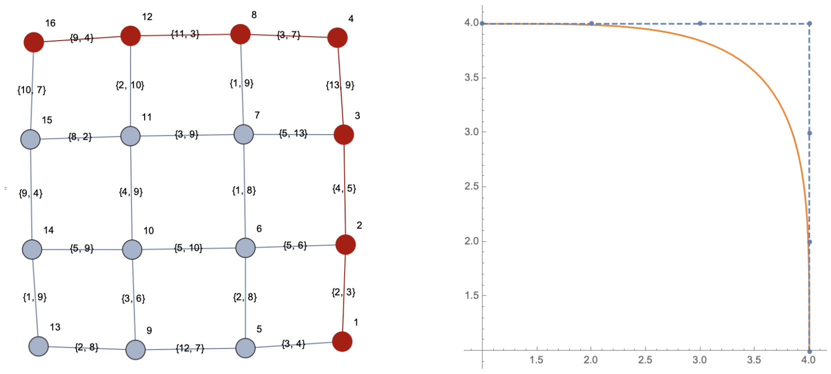

Figure 4: The shortest path (left) and the smooth Bézier curve with minimum total number of time travelled (right).

4.2 Two Dimensional Weights

For a two-dimensional weight, the calculation is based on the total number of combined weights which are the distance and the time travelled. Then, the algorithm will find the minimum total number of the combined weights to retrieve the shortest path as shown in Figure 5. Note that there are two shortest paths based on the total number of distance and time travelled using the same weight for each node.

The shortest path in Figure 5 (left) has a shorter distance but longer travel time, while the shortest path on Figure 5 (right) has a shorter travel time but longer distance. Even so, they both have the same total number of distance and time travelled combined. Note that shortest paths of Figure 5 are not the same as the shortest path in Figure 3 (left) and Figure 4 (left). This happened due to the algorithm that calculated the combined total number of distance and time travelled to produce the shortest path as in Figure 5. Meanwhile, in Figure 3 (left) and Figure 4 (left), the algorithm calculated the total number of distance and time travelled separately.

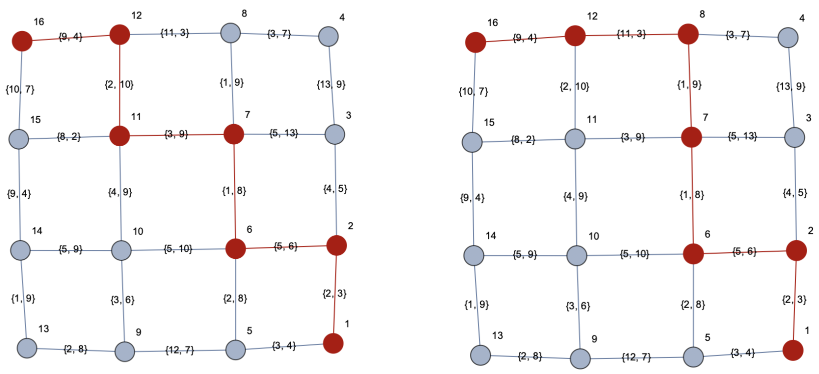

Figure 5: The shortest paths of combined weight with preferable distance (left) and preferable travel time (right).

Next, the shortest path will be interpolated into smooth Bézier curves as shown in Figure 6 in correspondent to the same shortest paths in Figure 5. From Figure 6, we cannot identify the best interpolating path planning curve based only on the combined total numbers of distance and time travelled. Therefore, we require an extra mechanism to determine the best interpolation curve using curvature as in Equation (10).

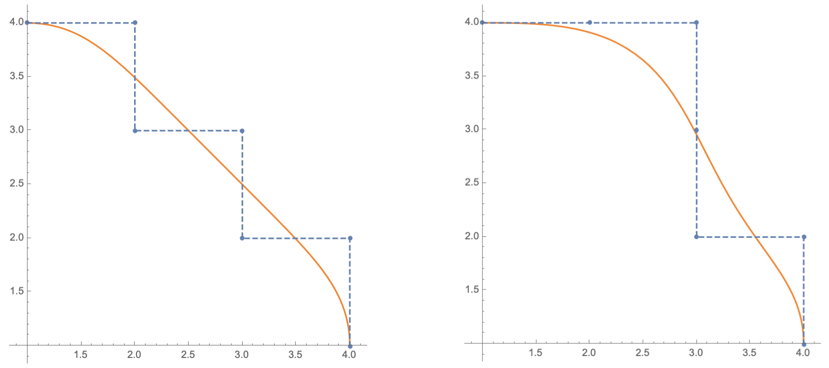

Figure 6: The Bézier curve interpolation of combined weight with preferable distance (left) and preferable travel time (right).

Based on Figure 6, it can be observed that both curves have the same total number of weight but different nodes visited, thus the control points for its Bézier curves are different. This led to a different curvature profiles for the Bézier curves in Figure 7 respectively. By comparing both curvature profiles in

Figure 7, we can identify the best curve interpolation for path planning in two-dimensional weight even though the total combined weight of distance and time travelled are the same. From this result, Figure 6 (left) with a more preferable distance will yield a very smooth path planning based on the curvature profile in Figure 7 (left).

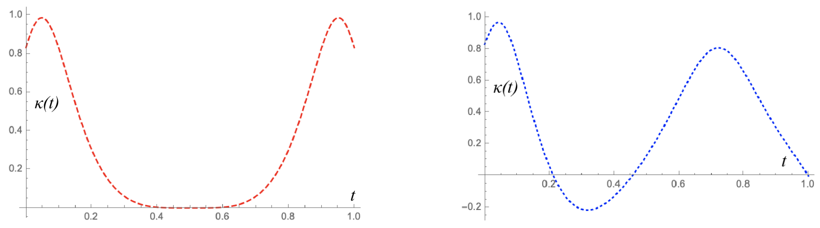

Figure 7: The curvature profile for the shortest path based on the total number of distance (left) and time travelled (right) in Figure 6 respectively.

The curvature in Figure 7 (left) does not have a negative value compared to Figure 7 (right), while both curvatures have the same maximum curvature values. The absolute curvature values between the maximum and the minimum showed that Figure 7 (left) has less value compared to Figure 7 (right). Furthermore, considering the number of turns for interpolation curves in Figure 6, Figure 6 (left) will give you less number of turns compared to Figure 6 (right). Hence, we can conclude that the shortest path in Figure 5 (left) with a more preferable distance will provide the best path planning.

Apart from the information on the distance and time travelled, all the interpolation of Bézier curves managed to satisfy the important properties in all cases. The shortest paths are generated from the nodes visited that consists of weights on its edge. Thus, the number of turns of interpolating curve for smooth path planning based on the shortest path generated for each case will be a useful information for robotics movement. The number of turns for the robotics is crucial to determine the speed and to ensure a smooth manoeuvre before reaching the final destination.

5. Conclusion

This paper presents an algorithm to find the shortest path for a multi-objectives environment using permutation method based on the concepts in Dijsktra's algorithm, as well as interpolating the shortest path into a smooth Bézier curve. The shortest path depends on the number of weights between each node. The lower the number of weights, the higher the possibility for the nodes to be visited. Once the shortest path is calculated, a smooth Bézier curve can be constructed. The theory in this paper can be applied for an unmanned military vehicle (UMV), where a smooth curve is needed especially in a multi-objectives environment that may occur due to topological factors, fuel consumption, and air resistance. For future work, this research can be extended using spatial path planning for drone route application. On top of that, Bézier curve based on smooth path planning may also be extended to find the curvatures (Misro et al., 2017b,c) versus speed (Ibrahim et al., 2017, Misro et al., 2019b).

Acknowledgement

This research was supported by Ministry of Higher Education Malaysia through Fundamental Research Grant Scheme (FRGS/1/2020/STG06/USM/03/1) and School of Mathematical Sciences, Universiti Sains Malaysia. The authors are very grateful to the anonymous referees for their valuable suggestion.

References

Akbaripour, H. and Masehian, E. (2017). Semi-lazy probabilistic roadmap: a parameter-tuned, resilient and robust path planning method for manipulator robots. *The International Journal of Advanced Manufacturing Technology*, 89(5-8):1401–1430.

Bircher, A., Alexis, K., Schwesinger, U., Omari, S., Burri, M., and Siegwart, R. (2017). An incremental sampling-based approach to inspection planning: the rapidly exploring random tree of trees. *Robotica*, 35(6):1327–1340.

Cimurs, R., Hwang, J., and Suh, I. H. (2017). Bezier curve-based smoothing for path planner with curvature constraint. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 241–248. https://doi.org/10.1109/IRC.2017.13.

Farin, G. (2002). *A history of curves and surfaces. In G. Farin, J. Hoschek and M. S. Kim (Eds.). Handbook of Computer Aided Geometric Design*, pages 1–21. Elsevier. https://doi.org/10.1016/B978-044451104-1/50002-2.

Ibrahim, M. F., Misro, M. Y., Ramli, A., and Ali, J. M. (2017). Maximum safe speed estimation using planar quintic Bézier curve with C^2 continuity. *AIP Conference Proceedings*, 1870:050006. http://dx.doi.org/10.1063/1.4995916.

Marsh, D. (2006). *Applied Geometry for Computer Graphics and CAD*. London: Springer Science & Business Media.

Misro, M. Y., Ramli, A., and Ali, J. M. (2017a). Quintic trigonometric bézier curve with two shape parameters. *Sains Malaysiana*, 46(5):825–831.

Misro, M. Y., Ramli, A., and Ali, J. M. (2017b). S-shaped and c-shaped transition curve using cubic trigonometric Bézier. *AIP Conference Proceedings*, 1870(1):050005. https://doi.org/10.1063/1.4995915.

Misro, M. Y., Ramli, A., and Ali, J. M. (2019a). Extended analysis of dynamic parameters on cubic trigonometric Bézier transition curves. In *2019 23rd International Conference in Information Visualization–Part II*, pages 141–146. doi: 10.1109/IV-2.2019.00036.

Misro, M. Y., Ramli, A., and Ali, J. M. (2019b). Quintic trigonometric Bézier curve and its maximum speed estimation on highway designs. *AIP Conference Proceedings*, 1974(1):020089. https://doi.org/10.1063/1.5041620.

Misro, M. Y., Ramli, A., Ali, J. M., and Hamid, N. N. A. (2017c). Cubic trigonometric Bézier spiral curves. In *2017 14th International Conference on Computer Graphics, Imaging and Visualization*, pages 14–20. doi: 10.1109/CGiV.2017.27.

Misro, M. Y., Ramli, A., Ali, J. M., and Hamid, N. N. A. (2017d). Pythagorean hodograph quintic trigonometric Bézier transtion curve. In *2017 14th International Conference on Computer Graphics, Imaging and Visualization*, pages 1–7. doi: 10.1109/CGiV.2017.26.

Pandey, A. and Parhi, D. R. (2017). Optimum path planning of mobile robot in unknown static and dynamic environments using fuzzy-wind driven optimization algorithm. *Defence Technology*, 13(1):47–58.

Raja, P. and Pugazhenthi, S. (2012). Optimal path planning of mobile robots: A review. *International Journal of Physical Sciences*, 7(9):1314–1320.

Rosen, K. H. and Krithivasan, K. (2012). *Discrete Mathematics and Its Applications: with Combinatorics and Graph Theory*. New York, NY: McGraw-Hill Companies.

Schrijver, A. (2012). On the history of the shortest path problem. *Documenta Mathematica*, 17:155–168.

Yin, C., Xiao, Z., Cao, X., Xi, X., Yang, P., and Wu, D. (2017). Offline and online search: UAV multiobjective path planning under dynamic urban environment. *IEEE Internet of Things Journal*, 5(2):546–558. doi: 10.1109/JIOT.2017.2717078.

Zhang, L., Sun, L., Zhang, S., and Liu, J. (2015). Trajectory planning for an indoor mobile robot using quintic Bezier curves. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 757–762. doi: 10.1109/ROBIO.2015.7418860.

Zhou, F., Song, B., and Tian, G. (2011). Bézier curve based smooth path planning for mobile robot. *Journal of Information and Computational Science*, 8(12):2441–2450.