An $O(n \log n)$ Shortest Path Algorithm Based on Delaunay Triangulation

Gene Eu Jan, *Member, IEEE*, Chi-Chia Sun, *Member, IEEE*, Wei Chun Tsai, and Ting-Hsiang Lin

***Abstract*—In Euclidean and/or λ-geometry planes with obstacles, the shortest path problem involves determining the shortest path between a source and a destination. There are three different approaches to solve this problem in the Euclidean plane: roadmaps, cell decomposition, and potential field. In the roadmaps approach, a visibility graph is considered to be one of the most widely used methods. In this paper, we present a novel method based on the concepts of Delaunay triangulation, an improved Dijkstra algorithm and the Fermat points to construct a reduced visibility graph that can obtain the near-shortest path in a very short amount of computational time. The length of path obtained using our algorithm is the shortest in comparison to the other fastest algorithms with $O(n \log n)$ time complexity. The proposed fast algorithm is especially suitable for those applications which require determining the shortest connectivity between points in the Euclidean plane, such as the robot arm path planning and motion planning for a vehicle.**

***Index Terms*—Delaunay triangulation, Fermat points, motion planning, $O(n \log n)$, shortest path problem.**

I. Introduction

DETERMINING the shortest connectivity between points (vertices) in the Euclidean and λ-geometry planes with obstacles is a problem with a long and convoluted history in applied mathematics. Our major interest in this paper is the shortest connection between two vertices (the shortest path problem), which involves finding a shortest path between two vertices with many obstacles such that the sum of the distances of its consecutive segments is minimized. Shortest path algorithms have already been applied to the motion planning of robots and the path planning of vehicular navigation. Furthermore, it can also be applied to electronic design automation, biological cell transportation, and operation research [1]–[4].

There are several approaches to the shortest path problem in the λ-geometry and Euclidean planes. In the λ-geometry planes, there are two kinds of approaches to solve the problem: maze routing and line-probe. For the maze routing, Lee's algorithm is the most widely used 2-geometry algorithm for finding a path between two vertices [5]. An algorithm that uses a suitable data structure for uniform wave propagation in the 4-geometry to find the shortest path was proposed by Jan *et al.* [6]. However, both Lee's and Jan's algorithms still require a large amount of storage space and the performance degrades rapidly when the size of the grid increases. In order to improve the search time, Soukup [7] proposed an iterative algorithm that can guarantee finding a path if a path exists between the source and the destination, but this path may not be the shortest one. Another algorithm to improve upon the speed was suggested by Hadlock [8]. The line-probe algorithms were developed by Mikami and Tabuchi [9], and Hightower [10]. The basic idea of a line-probe algorithm is to reduce the size of the memory requirement. Unfortunately, the path generated by Mikami and Tabuchi's algorithm may not be the shortest one, and the Hightower algorithm may not be able to find a path even when such a path does exists.

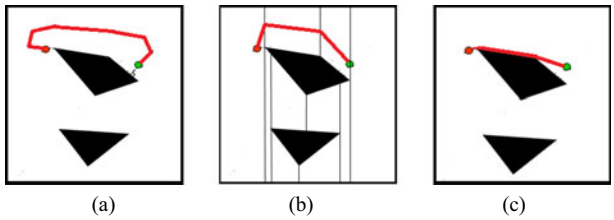


Fig. 1. Comparison between the proposed shortest path algorithm based on Delaunay triangulation and other two $O(n \log n)$ algorithms. (a) Voronoi (221). (b) Cell decomposition (160). (c) Proposed method (130).

In the Euclidean plane, there are three major approaches to finding the shortest path: cell decomposition, roadmaps, and potential field [11], [12]. The most popular cell decomposition method is *trapezoidal decomposition* [13]. This decomposition relies heavily on the polygonal representation of the planar configuration space. However, the trapezoidal decomposition method may not produce efficient paths for coverage. Two types of roadmaps are the most popular: the visibility graph and the Voronoi diagram. The visibility graph can be used searching for the shortest path and some algorithms were addressed by Rohnert [14] and Gao *et al.* [15]. Unfortunately, the visibility graph has many redundant edges and the cost of visibility graph construction is very high ($O(n^2)$). A better Voronoi diagram algorithm was proposed by Fortune [16], but the Voronoi diagram cannot find the shortest one. In this paper, we applied the concepts of Delaunay triangulation, Fermat points [17], and an improved Dijkstra's shortest path algorithm [18] to obtain the near-shortest path with $O(n \log n)$ time, where n is the number of obstacles. Furthermore, the length of path obtained by our algorithm is the shortest compared to two other $O(n \log n)$ algorithms as shown in Fig. 1 [13], [16], where Voronoi and cell decomposition result in 221 and 160 distant units, while the

Manuscript received July 22, 2012; revised January 23, 2013; accepted February 22, 2013. Date of publication April 5, 2013; date of current version February 20, 2014. Recommended by Technical Editor G. Liu.

G. E. Jan and W. C. Tsai are with the Department of Electrical Engineering, National Taipei University, New Taipei City 23741, Taiwan (e-mail: gejan@mail.ntpu.edu.tw; wctasi@mail.ntpu.edu.tw).

C.-Ch. Sun is with the Department of Electrical Engineering, National Formosa University, Yunlin County 632, Taiwan (e-mail: ccsun@nfu.edu.tw).

T.-H. Lin is with the Department of Statistics, National Taipei University, New Taipei City 23741, Taiwan (e-mail: tinghlin@mail.ntpu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMECH.2013.2252076

proposed algorithm requires only 130 units. Advantages of the presented fast algorithm are as follows.

1) A novel $O(n \log n)$ near-shortest path algorithm is presented based on the concepts of Delaunay triangulation, an improved Dijkstra algorithm, and the Fermat points of a reduced visibility graph.
2) It provides the fastest computing time among the other approaches and only has a 1.43% path distance difference statistically compared to the shortest $O(n^2)$ algorithm of the visibility graph.

This paper is organized as follows. Section II briefly introduces the concept of shortest path algorithms. In Section III, we describe the proposed idea of the triangulation-based shortest path algorithm, the performance of which is analyzed in Section IV. Section V explains two possible applications that benefit from the proposed algorithm. The experimental results are shown in Section VI, and Section VII concludes this paper.

II. Background

In this section, some well-known concepts, such as Euclidean plane, λ-geometry, Delaunay triangulation, Fermat problem, and Dijkstra's single-source shortest path algorithm, will be introduced.

Euclidean space is the Euclidean plane and 3-D space of the Euclidean geometry [19], as well as the generalizations of these notions to higher dimensions. The term "Euclidean" is used to distinguish these spaces from the curved spaces of non-Euclidean geometry and Einstein's general theory of relativity [20]. From the modern viewpoint, there is essentially only one Euclidean space for each dimension. In one dimension the Euclidean space is the real line; in two dimensions it is the Cartesian plane; and in higher dimensions it is the real coordinate space with three or more real number coordinates.

The λ-geometry allows edges with the angles of $i\pi/\lambda$, for all $i, \lambda = 2, 4, 8, \ldots$, and ∞ corresponding to rectilinear, $45^\circ, 22.5^\circ, \ldots$, and Euclidean geometries, respectively. For a 2-geometry neighborhood, we are only interested in the cells above, below, to the left, and to the right of a cell, that is, all of the cells that are distanced 1 unit from the center cell. For a 4-geometry neighborhood, each cell has eight neighbors that have distances of 1 or $\sqrt{2}$ units from the center cell [21].

In mathematics and computational geometry, a Delaunay triangulation for a set V of vertices in the plane is a triangulation DT(V) such that no vertex in V is inside the circumcircle of any triangle in DT(V). Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation, and they tend to avoid creating skinny triangles [22].

In geometry, the Fermat point of a triangle (also known as the Torricelli point) is the point at which the total distance from the three vertices of the triangle is a minimum. In order to locate the Fermat point of a triangle, the largest angle of the triangle must be less than or equal to 120°. When a triangle has an angle greater than 120°, the Fermat point is located at the obtuse angled vertex [23].

A path that interconnects the two vertices v and v' of a graph G with minimal length for all paths is called the shortest path between v and v'. Such a path is simple to determine. Starting with the vertex v, it is necessary to sequentially add vertices closet to the chosen vertices until v' is achieved [24]. Some efficient implementations of the Dijkstra's single-source shortest path algorithm are proposed using Fibonacci heaps (F-heaps) and radix heaps to find the shortest path in the graph model [18].

III. Algorithm and Illustration

In this section, we present the proposed method based on the concepts of Delaunay triangulation, the improved Dijkstra algorithm, and the Fermat points to construct a reduced visibility graph with the minimal path length in the Euclidean plane. The first step of the algorithm is to decide a source point, a destination point, and obstacles. Next, we will construct a Delaunay triangulation for the set of vertices (expressed by S including source, destination, and corner points of obstacles) and delete the edges of the Delaunay triangulation (expressed by E_t) that intersect with the obstacles. Fermat edges (expressed by E_f) are generated by connecting Fermat points to their corresponding triangles' corner points and neighboring Fermat points.

Because we have constructed the connected graph G with E_t and E_f, we can obtain the shortest path P in graph G, if it exists, using the Ahuja–Dijkstra algorithm. We also implement a function *PathShortening* to support the program. The purpose of the function *PathShortening* is to remove the redundant points (i.e., unnecessary Fermat points and corner points) from the source to the destination of P.

Function: *PathShortening*(P)

Step 1: Remove the redundant Fermat points of P.
Step 2: Generate the shortcuts of any two consecutive segments in P.
Step 3: Remove the shortcuts that pass through any obstacle.
Step 4: Sort the shortcuts by their corresponding length improvements in a descending order.
Step 5: Shorten the original path in a descending order if they are legal (no intersection with any obstacle).

END {Function of *PathShortening*}

Algorithm: The Delaunay triangulation-based shortest path

Init: Initiate the source point, destination point, and obstacles.

Step 1: Construct a Delaunay triangulation on the set of source, destination points, and corner points of obstacles in the Euclidean plane.
Step 2: Delete the edges of Delaunay triangulation that intersect with the obstacles.
Step 3: Insert Fermat points in triangles that have angles that are less than 120°.
Step 4: Connect Fermat points to their corresponding triangles' corner points and neighboring Fermat points.
Step 5: Add all obstacle edges to the connected graph G.
Step 6: Obtain the shortest path in graph G, if it exists, using the Ahuja–Dijkstra algorithm.
Step 7: Remove the redundant points from source to destination of the path by calling the function *PathShortening*

END {Algorithm of the Delaunay triangulation-based shortest path}

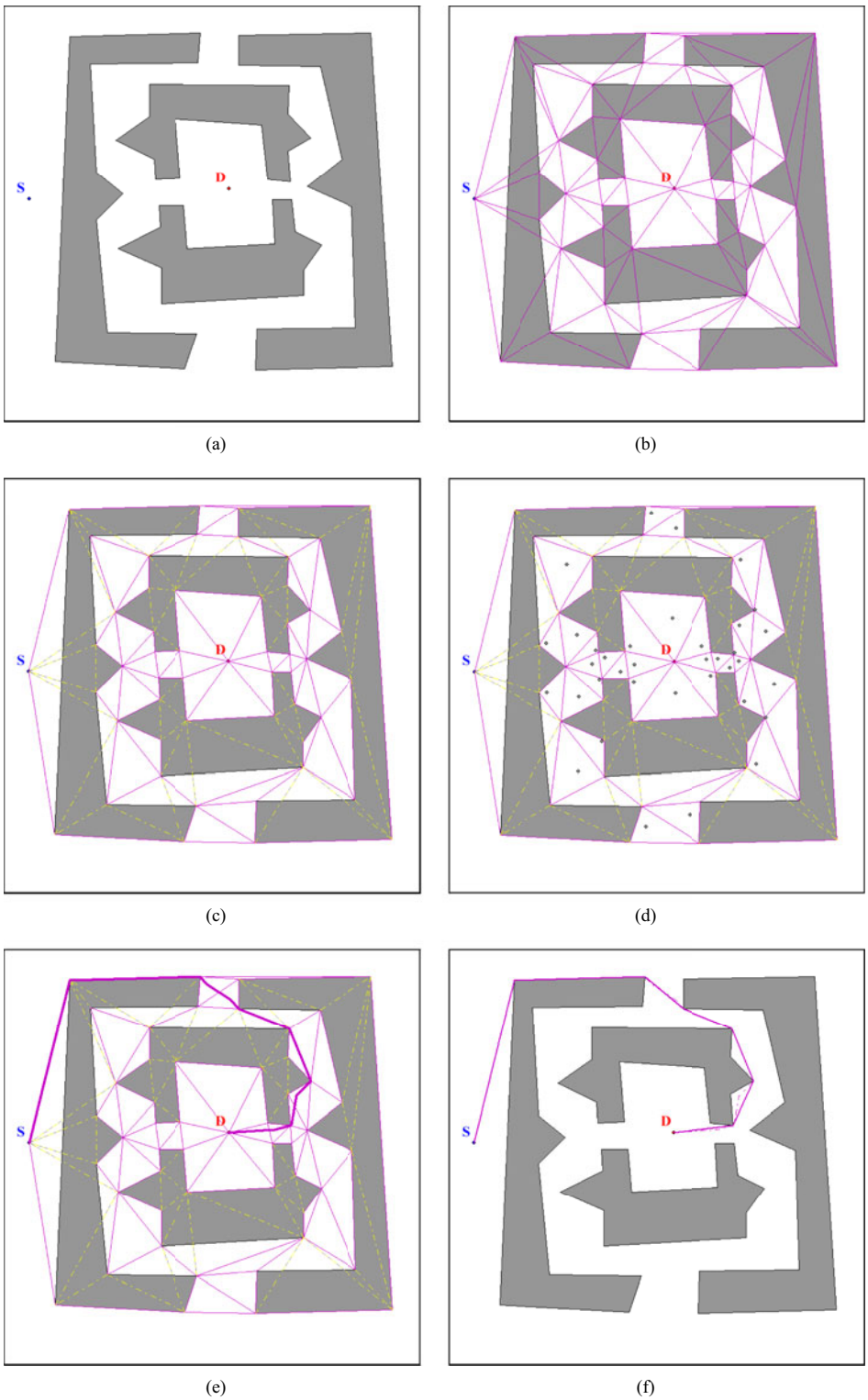


Fig. 2. Illustration of the proposed algorithm. (a) Initialization. (b) Delaunay triangulation. (c) Edge deleting. (d) Insert Fermat points. (e) Obtain the shortest path. (f) Path shortening.

Fig. 2 illustrates the detailed process. Fig. 2(a) shows the initial plane with obstacles that has a source point S and destination point D. Once Step 1 is executed, the triangles for the Delaunay triangulation will be generated as shown in Fig. 2(b). Then, the edge deleting is implemented to remove the edges which intersect with the obstacles, as shown in Fig. 2(c). Thus, the Fermat points are inserted into triangles that have angles that are less than 120° [see Fig. 2(d)]. Fig. 2(e) shows the result of the Ahuja–Dijkstra shortest path planning, and the final result after path shortening is depicted in Fig. 2(f). Fig. 3 shows one of

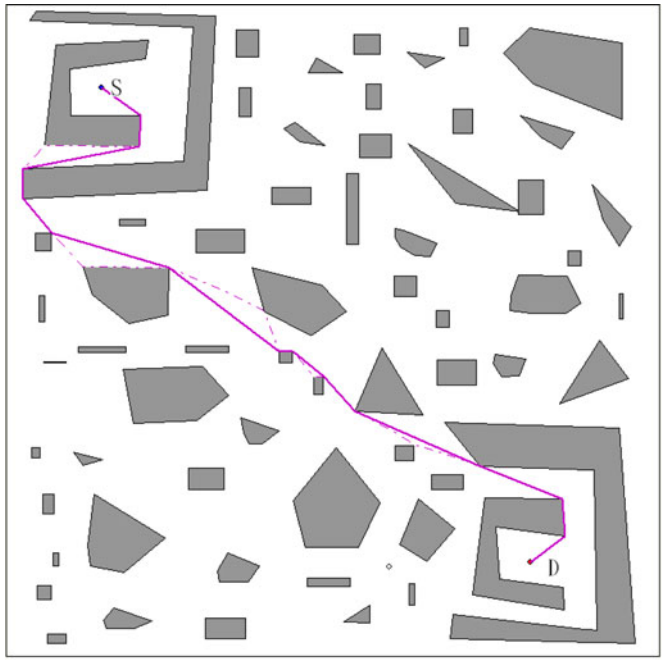


Fig. 3. Illustration of experimental result with random irregular shape obstacles.

the experimental results with random irregular shape obstacles, where the dashed line is the path without path shortening and the solid line shows the near-shortest path.

IV. Performance Analysis

Our near-shortest path algorithm in the Euclidean and λ-geometry planes with obstacles is faster than previously published algorithms.

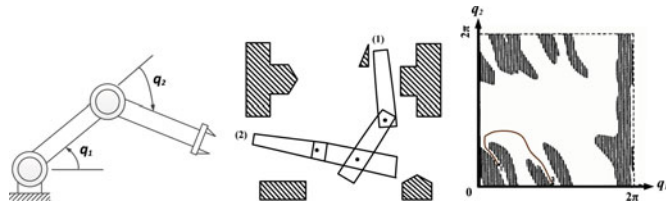
Theorem 1: The time complexity of our near-shortest path algorithm in the Euclidean planes with obstacles is $O(n \log n)$, where n denotes the number of obstacles.

Proof: We can generate DT with time of $O(n \log n)$ [16], during the edge deleting, as a result, $O(n \log n)$ dominates the time complexity in Steps 1 and 2 [25].

According to the Euler characteristic, the triangulation contains at most $2\,V - 2 - h$ triangles and $3\,V - 3 - h$ edges, where a graph has V vertices. That is, $V \leq 2 + (k_C \times n) = O(n)$, where 2 represents the source and destination, k_C is the maximal number of corner vertices of each obstacle, and h signifies the number of points on the convex hull.

This means that the number of triangles and Fermat points are bounded by $O(n)$; therefore, the number of edge connections in Step 4 is also bounded by $6 \times O(n)$ line. In Step 5, the required time to add all the edges of obstacles to the connected graph is $O(n)$. Therefore, the time complexity for Steps 3–5 is also $O(n)$.

In Step 6, the time complexity of the Ahuja–Dijkatra's algorithm using F-heaps and radix heaps is $O(E + V \log V)$, in which E is the number of edges in the graph. Meanwhile, $E = (3\,V - 3 - h) + (6 \times O(n)) = 3 \times O(n) + 6 \times O(n) = O(n)$. In conclusion, $O(E + V \log V) = O(n + n \log n) = O(n \log n)$.



(a) (b) (c)

Fig. 4. Robot arm path planning problem for rotating a planar two-revolute-joint robot arm (redraw figures from [11]). (a) Robot arm with two revolute joints. (b) Rotating the robot arm from position (1) to position (2). (c) Shortest path problem.

Finally, we will perform the path shortening from the source vertex to the destination vertex. The first substep generates the shortcuts of any two consecutive segments and determines which shortcuts do not intersect with any obstacle that is bounded by $O(n \log n)$ [25]. Next, we sort the shortcuts in a descending order with $O(n \log n)$ time.

In the final substep, we will shorten the original path with $O(n)$ time. The time complexity of Step 7 is $O(n \log n)$ because the number of vertices in the path is bounded by $O(n)$. It is concluded that the algorithm has the time complexity of $O(n \log n)$ from Steps 1 to 7.

Theorem 2: The space complexity of constructing the reduced visibility graph in the Euclidean planes with obstacles is $O(n)$, where n is the number of obstacles.

Proof: Once the Delaunay triangulation is generated, the number of triangles is less than $T = 2 \times (k_C \times n) - h - 2$, where k_C is the maximal number of corner vertices of each obstacle and h is the number of vertices on convex hull of the graph. Therefore, the space complexity is bounded by $O(n)$. Furthermore, the number of edges connecting Fermat points to triangle vertices and neighboring Fermat points is also bounded by $6 \times T$.

Among the reduced visibility graph approaches, the proposed algorithm almost outperforms other algorithms in each category from Tables I and II, except for the path length compared to the shortest $O(n^2)$ from the visibility graph algorithm.

V. Applications of the Shortest Path Problem

This section explains three applications that benefit from the proposed algorithm. The first application is robot arm path planning, as shown in Fig. 4, in which a planar two-revolute-joint robot arm rotates among obstacles. We assume that there is no mechanical stop in the joints and that q_1 and q_2 are angles associated with the joints as shown in Fig. 4(a). Therefore, a parameterization of this space is obtained by representing every configuration as $(q_1, q_2) \in [0, 2\pi) \times [0, 2\pi)$. Fig. 4(c) illustrates how to define the shortest path problem for rotating a planar two-revolute-joint robot arm from the position (1) to the position (2), as shown in Fig. 4(b). Apparently, rotating a robot arm with the least effort is an identical problem to determining the shortest path between positions (1) and (2) [11], [26].

The second application is the motion planning for a vehicle, as shown in Fig. 5, which determines how the point robots

TABLE I
COMPARISON OF DIFFERENT ALGORITHMS IN λ-GEOMETRY PLANES, WHERE $h \times w$ DENOTES THE SIZE OF GRID AND L DENOTES THE NUMBER OF LINE SEGMENTS GENERATED IN LINE-PROBE ALGORITHMS

λ–geometry	Algorithms					
	2–Geometry					4–Geometry
	Lee [5]	Soukup [7]	Hadlock [8]	Mikami [9]	Hightower [10]	Jan [6]
Time complexity	$h \times w$	$h \times w$	$h \times w$	L	L	$h \times w$
Find path if exists?	yes	yes	yes	yes	no	yes
Is the path shortest?	yes	no	yes	no	no	yes

TABLE II
COMPARISON OF DIFFERENT ALGORITHMS IN EUCLIDEAN PLANE, WHERE n DENOTES THE NUMBER OF OBSTACLES

Euclidean	Algorithms				
	Cell decomposition	Potential field	Roadmap		
	Trapezoidal decomposition		Visibility Graph	Voronoi diagram	Ours
Time complexity	$n \log n$	Not available	n^2	$n \log n$	$n \log n$
Space complexity of constructing connected graph	n	Not available	n^2	n	n
Find path if one exists?	yes	yes	yes	yes	yes
Is the path shortest?	no	no	yes	no	near shortest
Path length	$\gg 100\%$	$\gg 100\%$	100%	$\gg 100\%$	$\approx 101.4\%$

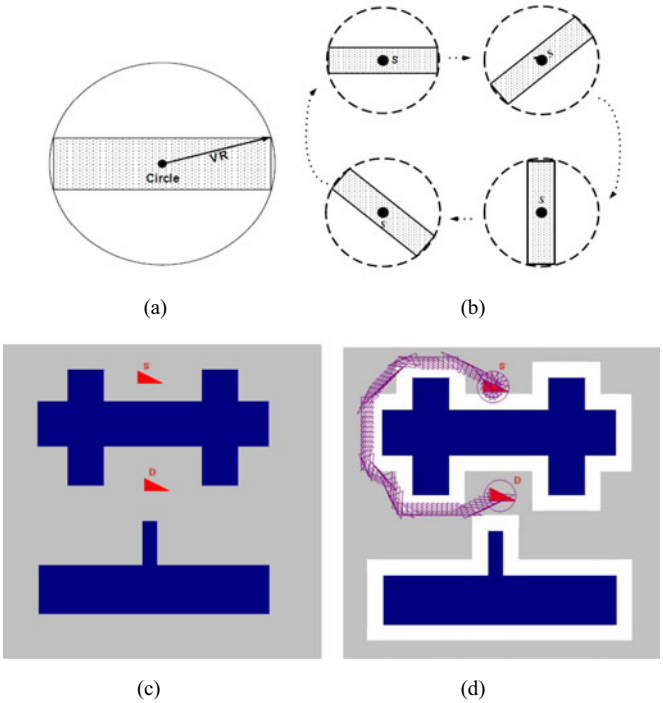
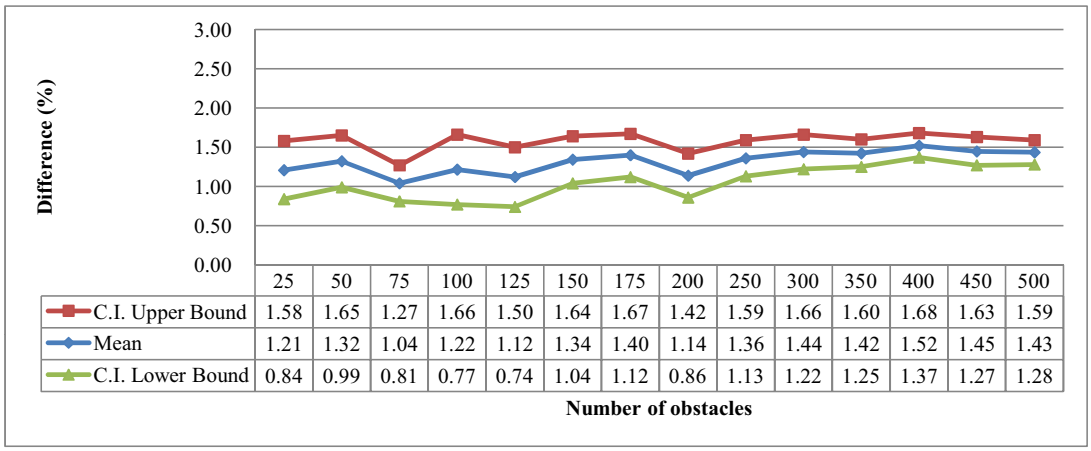


Fig. 5. Illustration of the shortest path and the virtual obstacles for a moving vehicle using the single circle model with rotation scheme [27]. (a) Point robot with virtual expansion. (b) Rotation scheme. (c) Shortest path problem. (d) Result.

shortest path planning can be applied to vehicle motion planning with virtual obstacles expansion and rotation scheme. First, in order to make sure that the vehicle will not collide with the obstacles, we must initially configure the robot and the obstacles to obtain a free workspace. Given a point robot of any shape in the workspace, we can circumscribe a smallest circle around the entire robot configuration. This is so-called the single circle model [27]. This robot can be considered as a single cell object for motion calculations, and the radius of the circumscribing circle is exactly the width of the virtual obstacles [shown in Fig. 5(a)], VR, which will be applied to the virtual obstacles expansion. Furthermore, the robots motion will be much closer to a real objects movement pattern if the rotation scheme is applied. However, to simplify the problem, most path searching algorithms only consider a robot as a point. The rotation scheme for the robot configuration using the single circle model is depicted as shown in Fig. 5(b). To verify the correctness and performance, we first assume that a vehicle needs to move from the source position S to the destination position D, as shown in Fig. 5(c). Once the virtual obstacles expansion and the rotation scheme are applied, a shortest path is obtained for the robot as shown in Fig. 5(d). Note that the rectangular or triangular object represents the robot configuration, black areas represent the obstacles, and white areas represent the virtual obstacles in these image planes. Apparently, Fig. 5(d) shows that the proposed shortest path algorithm can be also applied to intelligently rotate a mobile robot to pass a narrow passage with virtual obstacles expansion and rotation scheme.

Third, in addition to 2-D path motion planning, the proposed algorithm can also be applied to shortest path planning for vehicles and mission planning for cruise missiles in the quadric surface [28]. The basic ideas involve constructing circumscribing triangles around the obstacles to be avoided and generated the connected graph G' in the quadric surface with Delaunay triangulation edges (expressed by E'_t) and Fermat edges (expressed by E'_f). We then can obtain both the shortest paths for a vehicle in the graph G' or for a cruise missile in the graph G'' (e.g., 20 m above the G'), if they exist, using the Ahuja–Dijkstra algorithm.



	25	50	75	100	125	150	175	200	250	300	350	400	450	500
C.I. Upper Bound	1.58	1.65	1.27	1.66	1.50	1.64	1.67	1.42	1.59	1.66	1.60	1.68	1.63	1.59
Mean	1.21	1.32	1.04	1.22	1.12	1.34	1.40	1.14	1.36	1.44	1.42	1.52	1.45	1.43
C.I. Lower Bound	0.84	0.99	0.81	0.77	0.74	1.04	1.12	0.86	1.13	1.22	1.25	1.37	1.27	1.28

Fig. 6. Distribution of the path length difference (%).

TABLE III
COMPARISON BETWEEN OUR ALGORITHM AND VISIBILITY GRAPH ON AVERAGE PATH LENGTH

# of Obs.	Our Ave. Length	VG's Ave. Length	Difference	Difference (%)
25	627.75	620.39	7.36	1.21
50	616.15	608.30	7.85	1.32
75	576.27	570.37	5.90	1.04
100	612.88	605.20	7.68	1.22
125	529.09	523.14	5.95	1.12
150	526.83	519.62	7.21	1.34
175	530.22	522.52	7.70	1.40
200	561.08	554.73	6.35	1.14
250	515.73	508.62	7.11	1.36
300	547.06	539.14	7.92	1.44
350	576.30	568.03	8.27	1.42
400	529.17	521.05	8.13	1.52
450	528.92	521.30	7.62	1.45
500	523.55	516.01	7.54	1.43

TABLE IV
COMPARISON BETWEEN OUR ALGORITHM AND VISIBILITY GRAPH ON AVERAGE RUNNING TIME

# of Obs.	Our Ave. Runtime	VG's Ave. Runtime	Difference (x)
25	0.03	0.08	2.67x
50	0.09	0.55	6.11x
75	0.17	1.87	11.00x
100	0.28	4.40	15.71x
125	0.39	8.60	22.05x
150	0.56	14.93	26.66x
175	0.77	23.70	30.78x
200	0.98	35.49	36.21x
250	1.49	68.74	46.07x
300	2.10	118.61	56.37x
350	2.86	186.96	65.31x
400	3.79	279.63	73.72x
450	4.79	395.38	82.48x
500	6.10	561.87	92.16x

VI. EXPERIMENTAL RESULT

We have analyzed the performance of the proposed algorithm using several examples of the shortest path problem in the C++ language. The analysis was performed on an Intel Core i5-750 2.66 GHz Processor with 4-GB memory. We compare the proposed near-shortest path algorithm to the visibility graph on average path length with the number of random obstacles from 25 to 500 (in total 14 sets). The given parameters within a square region (e.g., 700×700) are the samples that include source point, destination point, and obstacles, and their corresponding locations are randomly generated.

Tables III and IV show the comparison results between the proposed algorithm and the visibility graph with the shortest path in terms of the average path length and average running time, respectively. The proposed algorithm cannot only reduce the computation time dramatically, but also obtain the shortest path compared to the other two $O(n \log n)$ algorithms: Voronoi and cell decomposition as illustrated in Fig. 1, where Voronoi, cell decomposition, and the proposed algorithm result in 221, 160, and 130 distant units, respectively. Using the concept of the Delaunay triangulation and Fermat points allows for searching the boundary of obstacles, whereas the Voronoi and cell decomposition algorithms will search the midlines between the obstacles within a specified safety margin. This safety margin will always result in longer paths than the proposed algorithm.

The average path difference is only 1.32%. Furthermore, according to the density of obstacles, the distribution of length differences between the heuristic and exact solutions is shown in Fig. 6. From the statistical criteria, a 95% confidence interval for the percentages of difference in lengths of two algorithms is between 1.25% and 1.43% (i.e., in total 1400 random samples of 14 sets are tested.). The individual C.I. for these 14 sets of obstacles is also presented in Fig. 6, and their differences are within 2%. Note that the differences go stable when the number of obstacles is above 250.

VII. CONCLUSION

In this paper, we proposed a novel algorithm to obtain the near-shortest path in the Euclidean plane based on the concepts of Delaunay triangulation, an improved Dijkstra algorithm and the Fermat points. The length of path obtained by our algorithm is the shortest among two other fastest $O(n \log n)$ algorithms in the literature. Compared to the other approaches of reduced visibility graph, the proposed fast algorithm almost outperforms the rest of algorithms, except the path length, compared to the shortest path algorithm of visibility graph with $O(n^2)$ time. This, however, only has a 1.43% path length difference in the

statistical analysis, which still presents a good result and the best computational time.

REFERENCES

[1] Y. Wu, D. Sun, W. Huang, and N. Xi, "Dynamics analysis and motion planning for automated cell transportation with optical tweezers," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 2, pp. 706–713, Jan. 2012.

[2] K. Harada, S. Hattori, H. Hirukawa, M. Morisawa, S. Kajita, and E. Yoshida, "Two-stage time-parametrized gait planning for humanoid robots," *IEEE/ASME Trans. Mechatronics*, vol. 15, no. 5, pp. 694–703, Oct. 2010.

[3] G. E. Jan, K. Y. Chang, and I. Parberry, "Optimal path planning for mobile robot navigation," *IEEE/ASME Trans. Mechatronics*, vol. 13, no. 4, pp. 451–460, Aug. 2008.

[4] L. Paull, S. Saeedi, M. Seto, and H. Li, "Sensor-driven online coverage planning for autonomous underwater vehicles," *IEEE/ASME Trans. Mechatronics*, to be published.

[5] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Trans. Electron. Comput.*, vol. EC-10, no. 3, pp. 346–365, Sep. 1961.

[6] G. E. Jan, K. Y. Chang, S. Gao, and I. Parberry, "A 4-geometry maze router and its application on multiterminal nets," *ACM Trans. Design Autom. Electron. Syst.*, vol. 10, pp. 116–135, Jan. 2005.

[7] J. Soukup, "Fast maze router," in *Proc. Design Autom. Conf.*, 1978, pp. 100–102.

[8] F. Hadlock, "Finding a maximum cut of a planar graph in polynomial time," *SIAM J. Comput.*, vol. 4, pp. 221–225, Sep. 1975.

[9] K. Mikami and K. Tabuchi, "A computer program for optimal routing of printed circuit connectors," in *Proc. Int. Fed. Inf. Process.*, 1968, vol. 47, pp. 1475–1478.

[10] D. Hightower, "A solution to line-routing problems on the continuous plane," in *Proc. Design Autom. Conf.*, 1969, pp. 1–24.

[11] J. Latombe, *Robot Motion Planning*. Norwel, MA, USA: Kluwer, 1991.

[12] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA, USA: MIT Press, 2005.

[13] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. New York, NY, USA: Springer-Verlag, 1985.

[14] H. Rohnert, "Shortest paths in the plane with convex polygonal obstacles," *Inf. Process. Lett.*, vol. 23, no. 2, pp. 71–76, 1986.

[15] B. Gao, D. Xu, F. Zhang, and Y. Yao, "Constructing visibility graph and planning optimal path for inspection of 2D workspace," in *Proc. IEEE Int. Conf. Intell. Comput. Intell. Syst.*, Nov. 2009, vol. 1, pp. 693–698.

[16] S. Fortune, "A sweepline algorithm for voronoi diagrams," in *Proc. 2nd Annu. ACM Symp. Comput. Geometry*, 1986, pp. 313–322.

[17] P. Spain, "The fermat point of a triangle," *Math. Mag.*, vol. 69, no. 2, pp. 131–133, Apr. 1996.

[18] R. Ahuja, K. Mehlhorn, J. Orlin, and R. Tarjan, "Faster algorithms for the shortest path problem," *J. ACM*, vol. 37, pp. 213–223, Apr. 1990.

[19] O. Byer, F. Lazebnik, and D. L. Smeltzer, *Methods for Euclidean Geometry*. Washington, DC, USA: Mathematical Association of America, 2010.

[20] A. Einstein, "Die Feldgleichungen der Gravitation," in *Sitzungsberichte der Preussischen Akademie der Wissenschaften zu Berlin*. Weinheim, Germany: Wiley-VCH Verlag, 1915, pp. 844–847.

[21] W. R. Tobler, *Cellular Geography*. Dordrecht, The Netherlands: Reidel, 1979, pp. 379–389.

[22] B. Delaunay, "Sur la sphère vide," *Bull. Acad. Sci. USSR*, no. 6, pp. 793–800, 1934.

[23] P. Fermat and M. Miller, *Pierre de Fermats abhandlungen úber maxima und minima*. Leipzig, Germany: Akademische Verlagsgesellschaft, 1934.

[24] E. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959.

[25] T. Jing, Z. Feng, Y. Hu, X. Hong, X. Hu, and G. Yan, "λ-oat: λ-geometry obstacle-avoiding tree construction with $o(n \log n)$ complexity," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 26, no. 11, pp. 2073–2079, Nov. 2007.

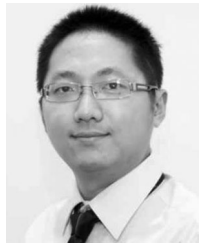
[26] X. Ding and C. Fang, "A novel method of motion planning for an anthropomorphic arm based on movement primitives," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 2, pp. 624–636, Apr. 2013.

[27] G. E. Jan, K.-Y. Chang, and I. Parberry, "A new maze routing approach for path planning of a mobile robot," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron.*, Jul. 2003, vol. 1, pp. 552–557.

[28] R. Helgason, J. Kennington, and K. Lewis, "Cruise missile mission planning: A heuristic algorithm for automatic path generation," *J. Heuristics*, vol. 7, no. 5, pp. 473–494, Sep. 2001.

Gene Eu Jan (M'00) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1982 and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, MD, USA, in 1988 and 1992, respectively.

He has been a Professor with the Departments of Computer Science and Electrical Engineering, National Taipei University, New Taipei City, Taiwan, since 2004, where he was also the Dean of the College of Electrical Engineering and Computer Science from 2007 to 2009. He has published more than 150 articles related to parallel computer systems, interconnection networks, motion planning, electronic design automation, and VLSI systems design in journals, conference proceedings, and books.



Chi-Chia Sun (M'09) received the B.S. degree in computer science and engineering from National Taiwan Ocean University, Keelung, Taiwan, in 2004 and the M.S. degree in electronic engineering from the National Taiwan University of Science and Technology,Taipei, Taiwan, in 2006. He also received the Dr.-Ing. degree with German Academic Exchange Service (DAAD) scholarship support from National Taipei University, New Taipei City, Taiwan, and National Formosa University, Huiwei City, Taiwan.

From April 2008 to March 2011, he was a Research Assistant with the Dortmund University of Technology, Dortmund, Germany, along with Prof. Dr.-Ing. Jürgen Götze. He is currently an Assistant Professor with the Department of Electrical Engineering, National Formosa University, Huwei, Taiwan. His research interests include parallel matrix computation, low-power iterative algorithms, signal processing, video compression, and VLSI/FPGA design.

Dr.-Ing. Sun is a member of the Institute of Electrical, Information, and Communication Engineers society.

Wei-Chun Tsai was born in Taiwan in 1986. He received the M.S. degree in computer science and information engineering from National Taipei University, New Taipei City, Taiwan, in 2012.

He is currently a Firmware Engineer with American Megatrends Inc. (AMI), Taipei, Taiwan.

Ting-Hsiang Lin received the Ph.D. degree in measurement, statistics, and evaluation from the University of Maryland, College Park, MD, USA.

She is currently an Associate Professor with the Department of Statistics, National Taipei University, New Taipei City, Taiwan. Her research interests include categorical variable analysis, latent variable analysis, and the zero-inflated Poisson model.