Send Orders for Reprints to reprints@benthamscience.ae

224 *The Open Cybernetics & Systemics Journal,* 2014, *8,* 224-229

Open Access

Fast Computation of Shortest Path for Visiting Segments in the Plane

Lijuan Wang[1,2], Bo Jiang[1*], Ansheng Deng[1] and Qi Wei[2]

[1]*College of Information Science and Technology, Dalian Maritime University, Dalian, China, 116026;* [2]*Department of Information and Science, Dalian Institute of Science and Technology, Dalian, China, 116052*

Abstract: Let s and t be two points in the plane, how to compute the Euclidean shortest path between s and t which visits a sequence of segments given in the plane, is the problem to be discussed in this paper, especially, the situation of the adjacent segments intersect is the focus of our study. In this paper, we first analyze the degeneration applying rubber-band algorithm to solve the problem and introduce the algorithm for computing Euclidean shortest path with removing sufficiently small segments. Then based on rubber-band algorithm, we present a new algorithm for solving the degeneration and computing the ESP by crossing over two segments to deal with intersection and in our algorithm the adjacent segments order can be changed when they intersect. Furthermore, we have implemented the two algorithms and have applied a large test data to test them. The experiments demonstrate that our algorithm is more efficient and effective, and it has the same time complexity as the rubber-band algorithm.

Keywords: Euclidean shortest path, degeneration, rubber-band algorithm, test data.

1. INTRODUCTION

Euclidean shortest path (ESP) problem is one of the typical problems in Computational Geometry. Its main aim is to find the shortest path between two points for a given series of obstacles in Euclidean space [1]. In this paper, we mainly study the algorithms for computing the Euclidean shortest path between two points s and t of visiting a sequence of segments given in the plane. Especially, the situation of the adjacent segments intersection is the focus of our study, but when three or more line segments intersect at one point then it is not the scope of this paper. The problem can be described as follows.

Let Π be a plane. Assume that there are n>1 segments $s_i \subset \Pi$ such that $s_i \cap s_j \cap s_k = \varnothing$, for $i \neq j \neq k$ and $i,j,k = 1,2,...,n$, see Fig. (**1**), then how to compute the shortest path from s to t that visits all the given segments s_i (at least once) is our study, here, the visiting order needn't to be in the given order. In Fig. (**1**), the path linked by dotted lines is the ESP from s to t which visits 4 given segments, and the visiting order is s_1, s_3, s_2, s_4.

The ESP problem of visiting a sequence of segments has been intensively studied. When the given segments don't intersect, such as Funnel algorithm [2] proposed by D. T. Lee and F. P. Preparata in 1984, and rubber-band algorithm [3, 4] (denoted by R algorithm) proposed by Fajie Li and Reinhard Kettle in 2007. R algorithm has the $K(\varepsilon)\cdot O(n)$ time, where $K(\varepsilon)=(L_0-L)/\varepsilon$, L is the true length of the ESP of segment set S, L_0 that of an initial path, and n is the number of segments of the set S. In 2011, Wang and Huo obtained that the time complexity of R algorithm is $O(n^2)$ when n is larger (*i.e.*,$n \geq 500$) by the experiment. Furthermore, they introduced divide and conquer into R algorithm and reduced the time complexity to $O(n)$ [5]. When the given segment may intersect, many researchers have been studying more effective methods. LI Fa-jie and KLETTE Reinhard in 2008 got an approximation solution by removing sufficiently small segments within an application of R algorithm [4]. So far, no other effective method has been reported yet. In this paper, based on R algorithm, we present a new algorithm for computing the ESP of given segments by the method of crossing over two segments to deal with the intersection and in our algorithm the adjacent segments order can be changed in order to get the shorter path when they intersect. The experiments show that our algorithm is efficient and accurate for solving the problem given above.

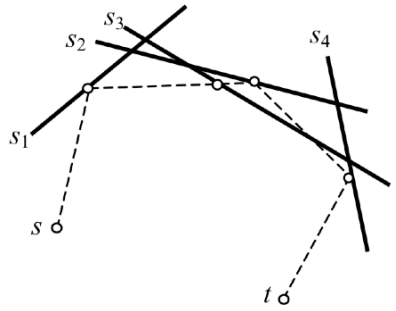

Fig. (1). ESP of visiting segments in the plane, for $n=4$.

2. THE DEGENERATION APPLYING ALGORITHM TO COMPUTE THE ESP

The basic idea of R algorithm is that it computes the shortest path by local optimization and continuous iteration.

1874-110X/14 2014 Bentham Open

Except start point and end point, all the path points from the first point will be updated one by one by each iteration. The algorithm will not stop until the total length of the path between two iterations only differs by ε (the chosen accuracy) at most [3, 4]. In R algorithm, the iteration is computed according to the given order of the segments.

When any two segments of the segment set S don't intersect, we can apply R algorithm to compute the ESP between two points s and t of visiting a sequence of segments in the given order easily [3-5], but when they intersect, it may be hard and even incorrect to get the shortest path. Taking the Fig. (**2**) for example, if the calculated path points q_i on s_i and q_{i+1} on s_{i+1} are at the intersection of s_i and s_{i+1} in current iteration, it is impossible to compute new q'_i and q'_{i+1} ($q_i \neq q_i'$, $q_{i+1} \neq q_{i+1}'$) that makes $q_{i-1}\, q'_i\, q'_{i+1}\, q_{i+2}$ shorter than $q_{i-1}q_i\, q_{i+1}\, q_{i+2}$ in the later iteration processes. Because q_i and q_{i+1} are both at the crossing of s_i and s_{i+1}, the path points q_i and q_{i+1} do not change in the later iteration process, obviously $q_{i-1}q_i\, q_{i+1}\, q_{i+2}$ isn't the optimal, except $q_{i-1,}\, q_{i,}\, q_{i+1,}\, q_{i+2}$ are on the same line. We call the situation in this example as a degeneration path of the algorithm [3-5]. When the degeneration appears, applying R algorithm will lead to failure.

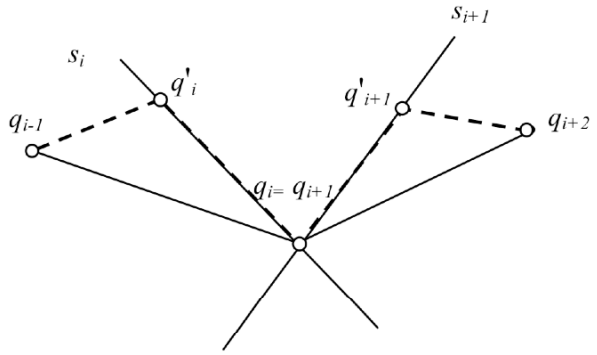

Fig. (2). The degeneration with an application of R algorithm.

Removing sufficiently small segments from segment s_i or s_{i+1} at the intersection can avoid the degeneration [4]. Based on this idea, in reference [6], the improved R algorithm, denoted by A1, is presented.

The degeneration can be avoided by applying A1 when the adjacent segments intersect, but the shortest path obtained is only an approximate because it is possible that the path through the intersection of the segments is the shortest. In this paper, based on R algorithm, we propose the method of crossing over two segments to compute path points by analyzing the position relationship between path points and segments, and we can change the order of the adjacent segments in order to get the shorter path when they intersect. The method not only can solve the degeneration effectively but also it's more efficient and accurate than A1. The experiments show that our algorithm is correct and effective.

3. THE METHOD OF DEALING WITH THE INTERSECTED SEGMENTS

When the segments do not intersect, references [3-5] has presented the solving method.

When the segments intersect, assume that the segments s_i and s_{i+1} intersect at the point C, p_{i-1} and p_{i+2} are the path points on s_{i-1} and s_{i+2} respectively. Then, there are some cases among the points p_{i-1} and p_{i+2} and the segments s_i and s_{i+1} , which are as follows.

Case 1 p_{i-1} and p_{i+2} lie on the different side of s_i and s_{i+1}

In this case, the approach in the iteration process is as follows. If $\overline{p_{i-1}p_{i+2}}$ passes through both s_i *and* $s_{i+1,}$ obviously, $\overline{p_{i-1}p_{i+2}}$ is the local shortest path, denoted by ρ, $\rho = \overline{p_{i-1}p_{i+2}}$, and the visiting order is the order of ρ passing through s_i and s_{i+1}, see Fig. (**3**). Otherwise, for Fig. (**3a**), there are many cases which can be seen in Fig. (**4**). According to R algorithm, we can choose one of the endpoints of s_i as p_i or one of the endpoints of s_{i+1} as p_{i+1} such that the local path is the shortest, and the visiting order is the same as above. Similarly, for Fig. (**3b**), it can be done as Fig. (**3a**). Since there are other possible position relationships between s_i and s_{i+1}, (*i.e.*, they are vertical), which also can be done in a similar way. In addition, in the flowing case 2 and case 3, the method of dealing with the endpoints is the same as that of Fig. (**3a**).

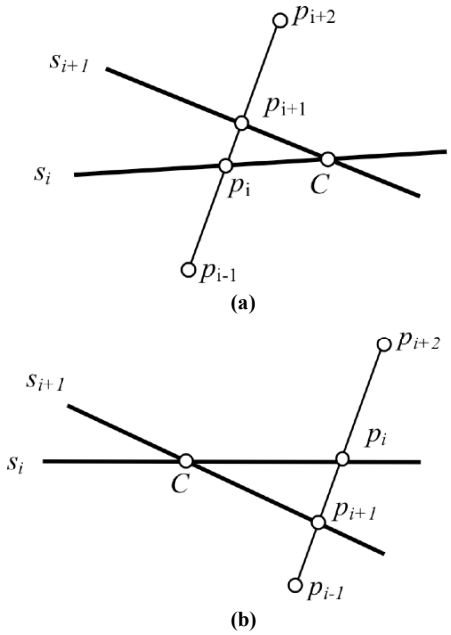

Fig. (3). $\overline{p_{i-1}p_{i+2}}$ passes through both s_i and s_{i+1}.

Case 2 p_{i-1} and p_{i+2} lie on the different side of s_i (s_{i+1}) and on the same side of $s_{i+1}(s_i)$

In this case, it needs to compute the reflection point r on the line segment s_i or s_{i+1} to find the shortest path [7, 8], see Fig. (**5**).

The approach in the iterative process is as follows. If $\overline{p_{i-1}p_{i+2}}$ only intersects with one of segments s_i and s_{i+1} , the reflection point r lies on the segment which does not intersect with $\overline{p_{i-1}p_{i+2}}$, $\rho = \overline{p_{i-1}r} + \overline{rp_{i+2}}$, and the visiting order is the order of ρ passing through s_i and $s_{i+1.}$

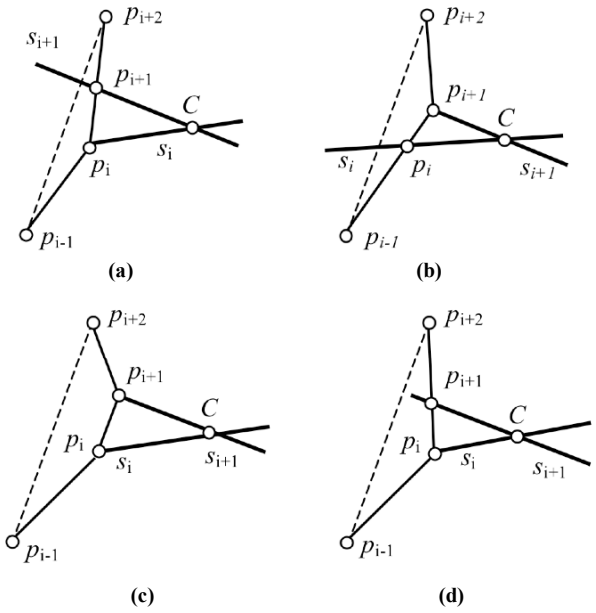

Fig. (4). $\overline{p_{i-1}p_{i+2}}$ does not intersect with s_i or s_{i+1}.

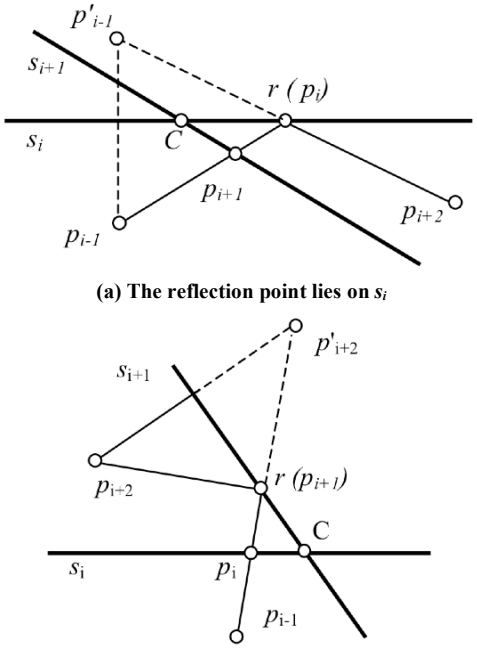

Fig. (5). p_{i-1} and p_{i+2} lie on the different side of s_i (s_{i+1}) and on the same side of s_{i+1}(s_i).

Case 3 p_{i-1} and p_{i+2} lie on the same side of s_i and s_{i+1}

In this case, there are two cases which are shown in Figs. (**6**) and (**7**).

(1) the case of computing one reflection

The two segments can be visited by computing a reflection point (see Fig. **6**). In this case, obviously, $\rho = \overline{p_{i-1}r} + \overline{rp_{i+2}}$. It is worth noting that when ρ passes through s_i or s_{i+1} twice, we can select the nearest intersection to C as the p_i or p_{i+1} in order to deal with endpoints easily.

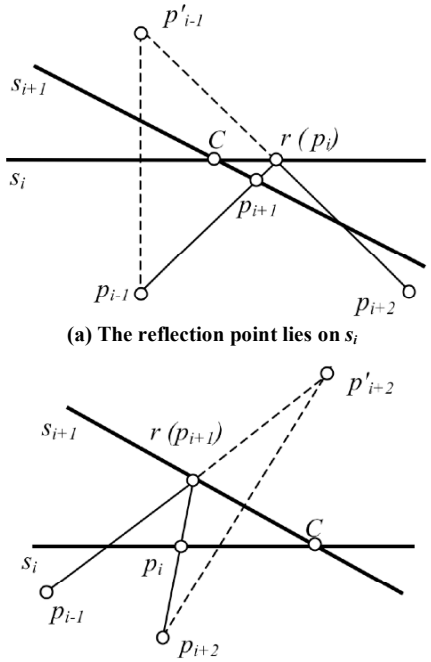

Fig. (6). p_{i-1} and p_{i+2} lie on the same side of s_i and s_{i+1} (One reflection point).

(2) the case of computing two reflections

It needs to compute two reflections when the two segments can't be visited by computing a reflection point. The approach is as follows. We make the symmetric point of p_{i-1} on s_i and the symmetric point of p_{i+2} on s_{i+1}, denoted by p'_{i-1} and p'_{i+2} respectively.

In this case, there are three situations when computing the local shortest path (see Fig. **7**).

① if $\overline{p'_{i-1}p'_{i+2}}$ intersects with s_i and s_{i+1} at the point p_i and p_{i+1} respectively, and p_i and p_{i+1} are on the same side of p_{i-1} and p_{i+2}, they are the two reflections and also the path points. If $\overline{p_{i-1}p_i}$ does not intersect with $\overline{p_{i+1}p_{i+2}}$, the shortest path is $\rho = \overline{p_{i-1}p_i} + \overline{p_ip_{i+1}} + \overline{p_{i+1}p_{i+2}}$, and the visiting order is the order of ρ passing through s_i and s_{i+1}(see Fig. (**7a**). Otherwise, we need to change the visiting order of s_i and s_{i+1} in order to get the shortest path, then $\rho = \overline{p_{i-1}p_{i+1}} + \overline{p_{i+1}p_i} + \overline{p_ip_{i+2}}$ (see Fig. (**7b**).

② if $\overline{p'_{i-1}p'_{i+2}}$ passes through the intersection C, in this case, the reflection points coincide with C and they are also the path points (p_i=p_{i+1}=C), $\rho = \overline{p_{i-1}C} + \overline{Cp_{i+2}}$, and the visiting order is the order of ρ passing through s_i and s_{i+1}. It easily can be proved by making two auxiliary points p'_i and p'_{i+1} on s_i and s_{i+1} (see Fig. (**7c**).

③ if $\overline{p'_{i-1}p'_{i+2}}$ doesn't pass through the intersection C and $\overline{p'_{i-1}p'_{i+2}}$ is different from the side of $\overline{p_{i-1}C}$ and $\overline{Cp_{i+2}}$, $\rho = \overline{p_{i-1}C} + \overline{Cp_{i+2}}$, the path points p_i and p_{i+1} coincide with C, and the visiting order is the order of ρ passing through s_i

and s_{i+1} (see Fig. **7d**). This can be proved as follows. In Fig. (**7d**), we take two points p'_i and p'_{i+1} on s_i and $s_{i+1,}$ since $\overline{p'_{i-1}p'_i} + \overline{p'_i p'_{i+1}} + \overline{p'_{i+1}p'_{i+2}} = \overline{p_{i-1}p'_i} + \overline{p'_i p'_{i+1}} + \overline{p'_{i+1}p_{i+2}}$, $\overline{p'_{i-1}C} + \overline{Cp'_{i+2}} = \overline{p_{i-1}C} + \overline{Cp_{i+2}}$, obviously, $\overline{p'_{i-1}C} + \overline{Cp'_{i+2}} < \overline{p'_{i-1}p'_i} + \overline{p'_i p'_{i+1}} + \overline{p'_{i+1}p'_{i+2}}$, then $\overline{p_{i-1}C} + \overline{Cp_{i+2}} < \overline{p_{i-1}p'_i} + \overline{p'_i p'_{i+1}} + \overline{p'_{i+1}p_{i+2}}$, $\rho = \overline{p_{i-1}C} + \overline{Cp_{i+2}}$ follows.

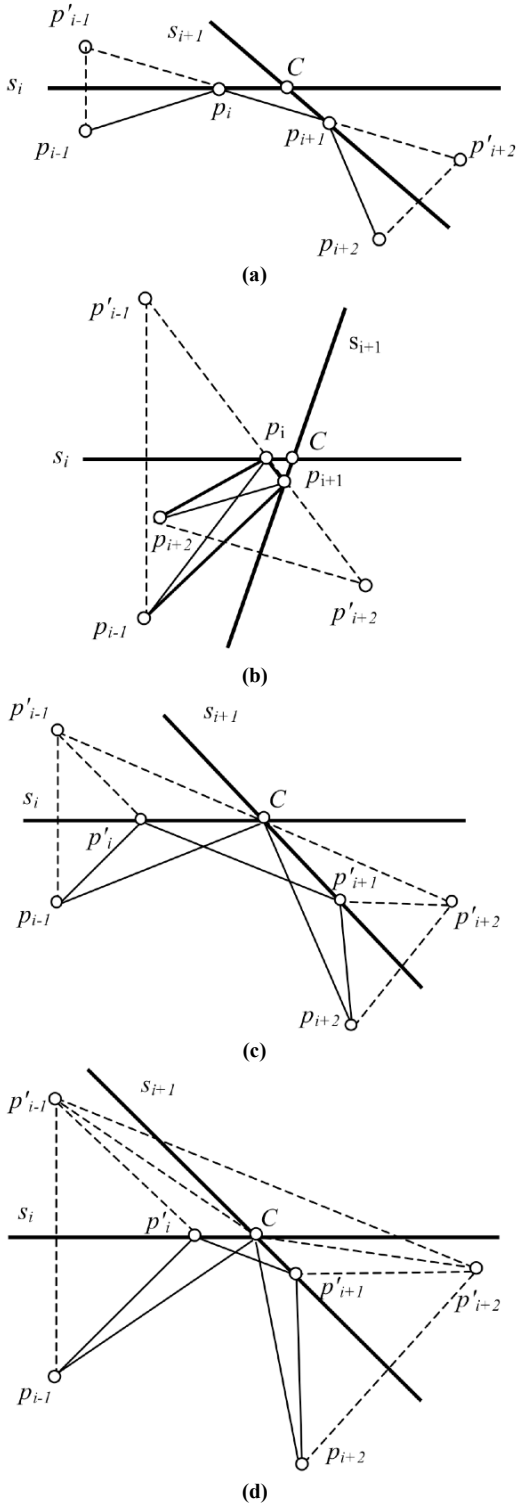

Fig. (7). p_{i-1} and p_{i+2} lie on the same side of s_i and s_{i+1} (Two reflection points).

4. NEW ALGORITHM

In this paper, we adopt the method of crossing over two segments when two segments intersect. The basic idea is as follows. Assume that s_i interests with $s_{i+1,}$ we can compute the path point p_i on s_i and the path point p_{i+1} on s_{i+1} according to the position relationship between the path points p_{i-1} and p_{i+2} and the segments s_i and $s_{i+1.}$ Next, we judge the position relationship of s_{i+1} and $s_{i+2,}$ if $s_{i+1} \cap s_{i+2} = \varnothing$, we compute the path point p_{i+2} on s_{i+2} and p_{i+3} on s_{i+3}. Otherwise, we compute the path point p_{i+1} on s_{i+1} and p_{i+2} on s_{i+2}.

The new algorithm is denoted by A2, which consists of Main procedure, Update procedure, ESPByOne procedure and ESPByTwo procedure. Main procedure is developed based on R algorithm which is used to calculate the iteration process, Update procedure is used to compute the path points in each iteration, ESPByOne is the procedure of crossing over one segment to compute the path points, the detail has been presented in reference [7], ESPByTwo is the procedure of crossing two segments to compute the path points, which is the core of this paper. ESPByTwo procedure can be described as follows.

ESPByTwo(p_{i-1}, p_{i+2}, s_i, s_{i+1}) procedure.

Here, s_i and s_{i+1} are two segments, p_{i-1} and p_{i+2} are two path points on s_{i-1} and s_{i+2}. The procedure is used to compute the new path points p_i and p_{i+1} on s_i and s_{i+1}.

Let $s = \overline{p_{i-1}p_{i+2}}$, L denote the path computed by ESPByOne procedure, and c is the intersection of s_i with s_{i+1}.

Case 1 p_{i-1} and p_{i+2} *lie on* the different side of s_i and s_{i+1}

if ($s_i \cap s \neq \varnothing$ && $s_{i+1} \cap s \neq \varnothing$), assume that r_1 and r_2 are the intersections of s_i with s, and s_{i+1} with s respectively, then let $p_i = r_1$ and $p_{i+1} = r_2$, and if ($\left|\overline{p_{i-1}r_1}\right| > \left|\overline{p_{i-1}r_2}\right|$) then call swap (s_i, s_{i+1}) and call swap(p_i, p_{i+1}).

Case 2 p_{i-1} and p_{i+2} *lie on* the different side of s_i (s_{i+1}) and on the same side of s_{i+1}(s_i)

if ($s_i \cap s \neq \varnothing$ || $s_{i+1} \cap s \neq \varnothing$)

if ($s_i \cap s \neq \varnothing$) then call ESPByOne(p_{i-1}, p_{i+2}, s_{i+1}), assume that r_1 and r_2 are the intersections of L with s_i, and L with s_{i+1} respectively.

Let $p_i = r_1$, $p_{i+1} = r_2$ and $sTemp = \overline{r_2 p_{i+2}}$,

if ($s_i \cap sTemp \neq \varnothing$) then call swap (s_i, s_{i+1}),and call swap (p_i, p_{i+1}).

else ESPByOne(p_{i-1}, p_{i+2}, s_i), assume that r_1 and r_2 are the intersections of L with s_i, and L with s_{i+1} respectively.

Let $p_i = r_1$, $p_{i+1} = r_2$ and $sTemp = \overline{p_{i-1}r_1}$.

Table 1. The Results of 10 Segments Sets by A1 and A2.

Segment Sets	A1 Algorithm		A2 Algorithm	
	Iteration Times	ESP	Iteration Times	ESP
100	161	36754	69	35647
200	376	83234	293	62814
300	386	126239	297	94627
400	420	167955	401	123784
500	454	203113	426	156598
600	472	247585	434	167352
700	486	284348	447	189651
800	497	321843	458	204536
900	512	358296	487	246725
1000	520	358401	490	276593

if ($s_{i+1} \cap sTemp \neq \varnothing$) then call swap (s_i , s_{i+1}), and call swap (p_i , p_{i+1}).

Case 3 p_{i-1} and p_{i+2} *lie on* the same side of s_i and s_{i+1}

ESPByOne(p_{i-1} , p_{i+2} , s_i) , let r_1 denotes the intersection of L with s_i and $sTemp = \overline{p_{i-1}r_1}$.

if ($s_{i+1} \cap sTemp \neq \varnothing$), and r_2 is the intersection, then let $p_i = r_1$, $p_{i+1} = r_2$, call swap (s_i , s_{i+1}) and call swap (p_i , p_{i+1}).

else ESPByOne(p_{i-1} , p_{i+2} , s_{i+1}), let r_2 denotes the intersection of L with s_{i+1} , and $sTemp = \overline{p_{i+2}r_2}$.

if ($s_i \cap sTemp \neq \varnothing$) , and r_1 is the intersection, then let $p_i = r_1$, $p_{i+1} = r_2$, call swap (s_i , s_{i+1}) and call swap (p_i , p_{i+1}).

else we make the symmetric point of p_{i-1} on s_i and that of p_{i+2} on s_{i+1} , denoted by p'_{i-1} and p'_{i+2} respectively, let $sTemp = \overline{p'_{i-1}p'_{i+2}}$, r_1 and r_2 denote the intersection s_i with $sTemp$, and s_{i+1} with $sTemp$.

if (p_{i-1} and c lie on the same side of $sTemp$) then let $p_i = p_{i+1} = c$.

else let $p_i = r_1$, and $p_{i+1} = r_2$.

if ($\overline{p_{i-1}r_1} \cap \overline{p_{i+1}p_{i+2}} \neq \varnothing$) then call swap (s_i , s_{i+1}) and call swap (p_i , p_{i+1}).

5. THE ANALYSIS OF RUNNING RESULT

We have implemented A1 and A2 with C++ program and have applied 10 randomly generated segment sets to test the algorithms, and the size of segments is 100, 200,…,1000 respectively. The experiment's results show that our algorithm is correct and efficient, which can be shown in Table **1**. From the Table **1**, we can see that the ESP of A2 is shorter than that of A1 and the iteration times of A2 is less than that of A1.

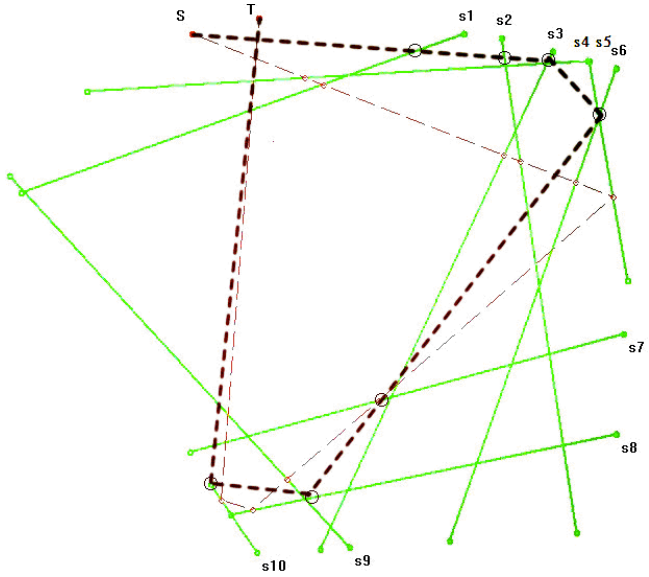

Fig. (8). The running result of 10 segments of A1 and A2.

To make the result clearly visible, we only present the result of 10 segments (see Fig. **8**). The solid lines are randomly generated segments, the path linked by the thin-dotted-lines from *S* to *T* is the ESP obtained by the A1 and the path linked by the thick-dotted-lines from *S* to *T* is the ESP obtained by the A2. It's obvious that the ESP obtained by A2 is shorter than ESP obtained by A1. In this example, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10} is the initial given order of segments. After running with our algorithm, the output is s_4,s_1,s_3,s_2,s_6,s_5,s_7,s_9,s_8,s_{10}. We can see the visiting order has been changed and we can get the shorter path and the faster running time. Furthermore, this example contained three position relationships between path points and segments dis-

cussed above, for example, (p_1, s_3, s_2, p_6), (p_2, s_6, s_5, p_7) and (p_9,s_8,s_{10},T) are the case1, case2 and case3 respectively.

6. CONCLUSION

In this paper, based on R algorithm, we present a new algorithm for computing the Euclidean shortest path of visiting a sequence of segments given in the plane. When the adjacent segments intersect, applying the method of crossing over two segments to deal with the intersection can effectively solve the degeneration caused by R algorithm. Furthermore, our algorithm can efficiently and accurately compute the ESP of given segments. Since dealing with degeneration has not increased the time complexity of R algorithm, it has the same time complexity as R algorithm.

This research has made preliminary results, the situation of three or three more line segments intersect at one point is an open problem [9].

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (No.61173034, 61272171) and the General Project of Liao Ning Province Science and Research (No. L2012487).

REFERENCES

[1] M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, Computational Geometry: Algoriths and Applications (3rd Edition), Springer-Verlag Publishers: Berlin, 2009, pp. 10-11.

[2] D. T. Lee and F. P. Preparata, "Euclidean shortest paths in the presence of rectilinear barriers," *Networks*, vol. 14, no. 3, pp. 393-410, September.1984.

[3] F. Li and R. Klette, "Exact and approximate algorithms for the calculation of shortest paths," *IMA J. Manag. Mathem.*, vol. 17, no.1, pp.134-138, 2006.

[4] F. Li and R. Klette, "Shortest path algorithms for sequences of polygons," *CAAI Trans. Intell. Syst.*, vol. 3, no. 1, pp. 23-30, 2008.

[5] L. Wang, L. Huo, and D. He, "An improved algorithm for euclidean shortest paths of visiting line segments in the plane," *J. Conver. Inform. Tech.*, vol. 6, no. 6, pp. 119-125, 2011.

[6] L. Wang, B. Jiang, Q. Wei, D. He, "Research on the algorithm for euclidean shortest paths of visiting line segments in the plane," *ICIC Express Lett.*, vol. 8, no. 6, pp. 1683-1687, 2014.

[7] X. Tan, "Fast computation of shortest watchman routes in simple polygon," *Inform. Process. Lett.,* vol. 77, no. 1, pp. 27-33, 2001.

[8] A. Dumitrescu, J. S. B. Mitchell and P. Żyliński, "Watchman Route for Line and segments, " *Algorithm. Theory – SWAT 2012 Lect. Notes Comp. Sci.*, vol .7357, no. 14, pp. 36-47, May. 2012.

[9] M. Dror, A. Efrat, A. Lubiw and J. S. B. Mitchell, "Touring a sequence of polygons," In Proc. 35th Annu. ACM Sympos. Theory Comput., California, USA, 2003, pp. 473-482.

Received: September 22, 2014 Revised: November 30, 2014 Accepted: December 02, 2014