British Journal of Mathematics & Computer Science
9(6): 453-472, 2015, Article no.BJMCS.2015.217
ISSN: 2231-0851

SCIENCEDOMAIN *international*
www.sciencedomain.org

Maze Construction by Using Characteristics of Eulerian Graphs

Tomio Kurokawa[1*]

[1]*Department of Information Science, Aichi Institute of Technology, 1247 Yachigusa, Yagusa-cho, Toyota 470-0392, Japan.*

Article Information

DOI: 10.9734/BJMCS/2015/18125
Editor(s):
(1) Doina Bein, Applied Research Laboratory, The Pennsylvania State University, USA.
Reviewers:
(1) Krasimir Yordzhev, South-West University, Bulgaria.
(2) Anonymous, Jawaharlal Nehru Technological University, Anantapur, India.
Complete Peer review History: http://sciencedomain.org/review-history/9770

Original Research Article

Received: 05 April 2015
Accepted: 07 May 2015
Published: 15 June 2015

Abstract

This paper describes the method to construct a maze for which the solution path goes along with the line of a closed one stroke drawn curve. If the crossing of the curve is considered a vertex and the curve between the two crossings (vertices) an edge, the closed curve becomes an Eulerian graph. Since maze does not allow the solution path to intersect itself, the proposed method constructs the path with no crossing; but it goes through every part of the original curve only once. In the process of the path construction, it makes use of the characteristics of Eulerian Graph. The theory was developed and a number of experiments were successfully conducted in some different situations.

Keywords: Maze; closed one stroke curve; contour cycle; Eulerian graph; bipartite graph.

1 Introduction

Maze generation by computing with square grids has been around for some years. The article [1] showed various algorithms to generate a square grid based maze, usually used, relating with graph theory. Hosokawa [2] presented a number of maze generation algorithms with the relation between path, wall, and coordinates. Xu and Kaplan [3] illustrated complex mazes emphasizing aesthetics for the maze. They also introduced a picture maze [4] called 'Maze-a-pix' by illustrating a sample picture maze, which is provided by Conceptis, Ltd. [5]. However, the method of the picture maze generation was not given. The internet site of the company shows various picture mazes and states a history of computer generated picture maze. Okamoto and Uehara [6] demonstrated an automatic generation algorithm for a picture maze by constructing a Hamiltonian path which fills up the picture region. However, the path is generated on the 2-by-2 extended picture array. The maze is so constructed that the player obtains the solution path by filling up all the path

*Corresponding author: kurokawa@aitech.ac.jp;

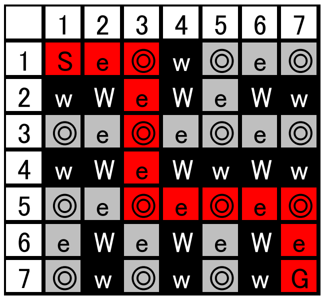
cells on the maze and by painting the picture region. Hamada [7-8] improved the method [6] by enabling to place the start and goal at different positions. Kurokawa developed a more efficient algorithm of construction of Hamiltonian path, with a rigorous proof, to generate a picture maze and introduced a generalized idea for maze path construction [9]. Ikeda and Hashimoto [10] illustrated a picture maze generation method by using simulated annealing. Kurokawa, Mori, and Mizuno [11-13] took a different approach and demonstrated the method to generate a picture maze by making use of the picture contours. In this method, all the contours are extracted and are incorporated into the picture maze. It is a very flexible method and can be applicable to a wide range of binary pictures. F. J. Wong and S. Takahashi [14] constructed a hybrid picture maze from two photograph images. This maze is not based on the square grid type mazes but on the special grids computed from the image. Two photo pictures are overlapped in the maze. The maze player depicts an outline of one picture on the other picture background.

2 Maze Structure

There are a number of different representation structures of a square grid maze. This paper uses one of them and it is explained in this section. A computer constructed maze is usually represented by a two-dimensional array. Such a maze consists of path and wall. A path cell is either a vertex or an edge (of a graph). Fig. 1 is a sample maze array of size 7x7---where '◎' is a vertex; 'e' is an edge, and 'W' and 'w' are walls. A vertex is placed one in every two cells horizontally and vertically. An edge represents the connection between the two vertices. Diagonally adjacent cells to a vertex are wall cells; there are four such cells around a vertex; the vertically or horizontally adjacent cells to a vertex are edge or wall cells. If the cell represents the connection between the two vertices, then it is an edge cell; otherwise it is a wall cell.

The connection between the two vertices is 4-way, vertical or horizontal; and no diagonal connection is allowed. The entire path routes in a maze organize a spanning tree. A vertex and an edge alternately appear along a path. The maze solution of Fig. 1 is the path from the start (S) to the goal (G), which is one of the paths from a vertex (the tree root) to another vertex (a leaf). The maze structure is kept maintained even if the maze is shifted in any direction one or more positions on the array. However, in Fig. 1, vertices are placed on (odd, odd) coordinates.

If the walls are thinned [15], the maze becomes as presented in Fig. 2. This maze, to be presented to a player, actually consists of two kinds of cells: path and wall. However, it also represents the corresponding graph with vertices and edges. There are other maze representations [16]. In short, a maze is said to be a collection of vertices and edges like a graph. However, more restrictions are imposed on a maze: physical positions of vertices and edges; the number of edges to be connected to a vertex and the like.



	1	2	3	4	5	6	7
1	S	e	◎	w	◎	e	◎
2	w	W	e	W	e	W	w
3	◎	e	◎	e	◎	e	◎
4	w	W	e	W	w	W	w
5	◎	e	◎	e	◎	e	◎
6	e	W	e	W	e	W	e
7	◎	w	◎	w	◎	w	G

Fig. 1. Maze structure

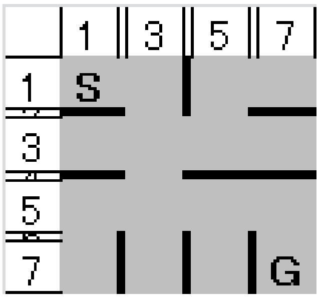


Fig. 2. Maze to be presented to a player

3 Preceding Studies: Organizing Contour Cycles from Binary Pictures

MS Visual C++ 2010 Express [17] and OpenCV library [18] were used as the programming tools in the experiments. The paper [13] describes how to construct contour cycles with maze path structure for a given binary picture such as Fig. 3. The procedure is expressed as:

1. Prepare a binary picture like (Fig. 3).
2. Extract all the contours of the picture (Fig. 3). OpenCV [18] provides software to extract contours for binary pictures. Each contour is represented by a sequence of coordinate addresses. The connectivity of the contours is of 8-way.
3. The 8-way is converted to the 4-way along all contours (Fig. 3). This is to adjust the structure to that of a maze.
4. Each pixel of the contours is considered to be a maze vertex and to be placed one in every two cells vertically and horizontally on a maze array (Fig. 4 (a)). Dotted contours are organized.
5. Filling all the gaps between the two previously adjacent vertices completes all the contour cycles of the original picture. The cycles are of maze path structure (Fig. 4 (b)).

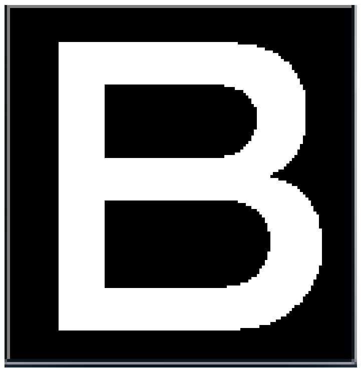
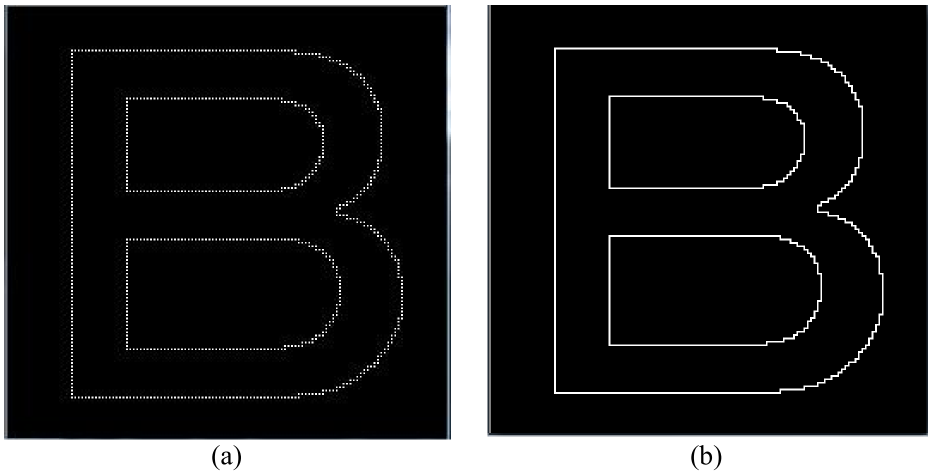


Fig. 3. Original binary picture



(a) (b)

Fig. 4. Contour cycles with maze path structure: (a) dotted contour cycles; (b) contour cycles with maze structure

4 Eulerian Graph and One Stroke Closed Drawing

Consider a graph $G = (V, E)$. A closed walk in G is called an Eulerian tour if it contains all the vertices and all the edges exactly once (while vertices can be repeated) in G. And a graph possessing a closed Eulerian tour is called an Eulerian graph [19]. Theorem of Eulerian graph states that a graph $G = (V, E)$ is Eulerian if and only if it is connected and each vertex has an even degree, where V is the set of vertices and E is the set of edges of the graph [19].

Consider the closed line drawing in Fig. 5, which is obtained by Microsoft Paint [20]. Assume that each of the curve intersections is a vertex and that the curve between two intersections is an edge. Then the line drawing can be considered as a graph. Since the drawing is of one stroke, the graph is connected and each vertex has even number of edges. Accordingly, the graph is Eulerian. For more natural situation, each edge between two intersection vertices should be an alternating sequence of an edge and a vertex, starting and ending with an edge like with the maze structure of Sections 2 and 3. So the closed one stroke drawing is Eulerian given the above relation between the line drawing and Eulerian graph. A Lemma of the above theorem of Eulerian graph [19] states that if the graph G has all degrees even, then the edge set E can be partitioned into disjoint subsets $E_1, E_2, \dots, E_k$, where each E_i is the edge set of a cycle and

$$E = \cup_{i=1}^{n} E_i. \qquad (1)$$

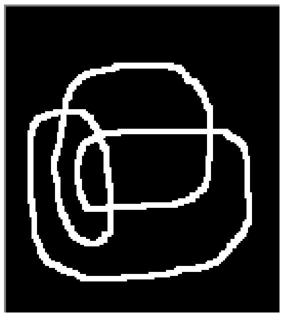


Fig. 5. Closed one stroke drawing with line width of 3 pixels

5 Contour Extraction and Selection of a Set of Cycles Covering the Entire Drawing

Applying the method of Section 3 to a binary drawn curve like Fig. 5 enables to extract the boundaries of the curve and construct the contour cycles (Fig. 6) of maze structure [13]. The cycles in Fig. 6 are numbered as given by OpenCV [18]. The cycles cover every region segment surrounded by the original curve. That is, every region surrounded by the original curve is surrounded by one of the contour cycles. This means in turn that every part of the drawn curve is covered doubly by two different contour cycles. That is, every line part (center) of the original curve is put between two adjacent cycles.

Assume an ideal imaginary situation, where the drawn curve has zero width and then every part of the curve is overlapped by the two different cycles, the adjacent cycles. This means that there are two sets of cycles in Fig. 6, each of which constitutes Eulerian graph. What is more, the two cycle sets are different but occupy the same vertices and edges as a whole.

Consider the graph of the vertex set and the edge set such that the vertex set consists of the vertices representing the region surrounded by the drawn curve and the edge set consists of the edges which exist between the two vertices if the two regions (equivalent to the cycles) are adjacent. This graph can be called a region graph or a cycle graph because the region and its surrounding cycle essentially mean the same concept.

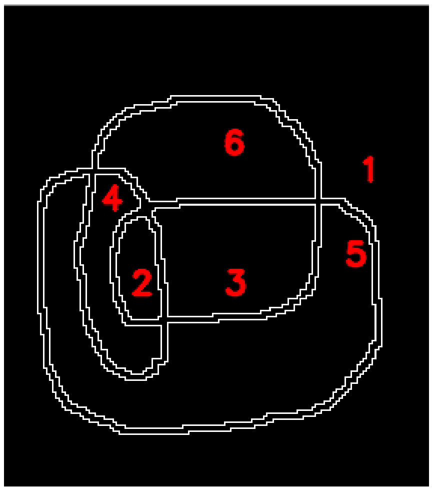


Fig. 6. Contour cycles constructed with region (cycle) numbers

Since the original curve graph is Eulerian, the region graph must be bipartite and 2-colorable. The cycles can be separated into two groups based on the adjacency of the cycles. This adjacency can be checked by the distance between the two cycles. Therefore, the bi-partitioning is possible.

In order to explain the adjacency between two cycles, it is necessary to define how the data of a cycle is organized. The data of cycle *j* is a sequence of coordinates defined as:

$$(x(j,i),y(j,i)),\ where\ 1 \le j \le c\ and\ 1 \le i \le n_j; \tag{2}$$

c is the number of cycles extracted; and n_j is the number of vertices on cycle *j*.

The adjacency checking can be done by a distance computation between two cycles. Two cycles *j* and *i* are considered, here, to be adjacent if there are more than a certain number, say *n*=5, of vertices on *j* such that the shortest distance from the vertex *l* on *j* to the cycle *i* is less than the overall average of the shortest distance (d_{AS}) multiplied by a certain number *a* (=2). The shortest distance from the vertex *l* on cycle *j* to cycle *i*, $d_S(j,l,i)$, is defined as:

$$d_S(j,l,i) = \min_{1\le k\le n_i} d(j,l,i,k), \tag{3}$$

$$d_{AS} = \frac{1}{\sum_{j=1}^{c} n_j}\sum_{j=1}^{c}\sum_{l=1}^{n_j} \min_{1\le i\le c} \min_{1\le k\le n_i} d(j,l,i,k),\ \text{and} \tag{4}$$

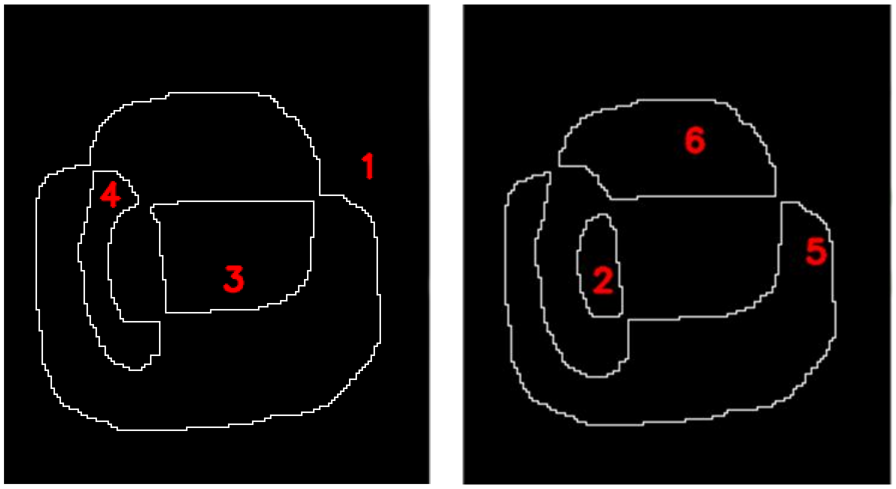
$$d(j,l,i,k) = \sqrt{(x(i,k)-x(j,l))^2 + (y(i,k)-(y(j,l))^2}\ ,\ where \tag{5}$$

$j \ne\ i$; n_j is the number of vertices on cycle *j*. So if more than *n* (=5) such different vertices on *j* exist, the two cycles *j* and *i* are considered adjacent. If *S* is such a set of *l*, it is expressed by (6):

$$S = \{l \mid 1 \le l \le n_j \text{ and } d_S(j,l,i) \le a * d_{AS},\ \text{where } a = 2\}. \tag{6}$$

If the size of *S* is larger than *n*, cycles *j* and *i* are considered adjacent.

The result of this bi-partitioning is shown in Figs. 7 (a) partition *X* and (b) partition *Y*. Fig. 8 shows the bi-partitioning Algorithm. Since the separation is based on the adjacency between two cycles, the two groups should occupy the same vertices and edges---looking perfectly alike assuming the original curve is of zero width. It is because that there are exactly same vertices and edges in the cycle sets in the extracted contours cycles. However, Figs. 7 (a) and (b) were obtained from the actual drawing with curve width of 3 pixels. Accordingly, they look somewhat different. The shapes also may look different from the original curve. But the drawing is Eulerian enough so that bi-partitioning works well. If the experiment is done with curve width of 1 pixel, they should look more alike. The experiments with narrower curves are demonstrated in later sections. Both of the cycle sets of Figs. 7 (a) and (b) correspond to the edge set of expression (1).



(a) **(b)**

Fig. 7. Bi-partitioned contour cycles: (a) partition *X*; (b) partition *Y*

Fig. 9 is the bi-partitioned graph obtained by examining the adjacency between every two cycles of Fig. 6; and Fig. 10 is the region adjacency graph equivalent to Fig. 9. If one of these cycle sets are merged into one single cycle, then it can be the solution path of the maze which goes through every part of the originally drawn curve. There are two such cycle sets. So there are at least, two such merged cycles. Depending on the merge, different merged cycles can be constructed.

6 Merge of Contour Cycles to Construct One Single Cycle

Previous section provides the method to obtain a set of cycles which cover the entire drawing. In Eulerian graph, the set of cycles constitute one single circuit which passes through all the edges, each only once but some vertices with some repetition. However, the actually extracted cycles are usually separate existence as shown in Fig. 6 or Figs. 7 (a) and (b). Therefore, there are usually no common intersection vertices between the actually extracted cycles. However, the intersection vertices are supposed to exist around the positions where the original curve intersects. By finding those places and connecting the cycles repeatedly, it is possible to construct one single cycle.

A single cycle can be obtained by merging the cycles in Fig.7 (a) as in Fig. 11 (a); the cycles in Fig. 7 (b), also can be merged in to one single cycle as Fig. 11 (b). Both of those merged cycles go through all the parts of the original curve. If a part of the cycles is cut and removed then the two ends can be the start and the goal of the solution path of the maze. See Figs. 12 (a) and (b). The merging algorithm is given in Fig. 13.

```
Bipartition Algorithm
Assumption:
R: a set of contour cycles extracted
n: number of cycles in R
X, Y: partition sets of the bipartite graph, both sets are empty at the beginning; they will
have the resulting vertices of bipartition.
1.      Remove an arbitrary cycle j from R and put it into X.
2.      Repeat {
3.        For each j in X do {
4.          For each i in R do {
5.            If j and i are adjacent, do {
6.              Remove i from R and put it into Y.
7.              If no cycle remains in R, then finish.
8.            }
9.          }
10.       }
11.       Exchange X and Y.
12.     }
```

Fig. 8. Bi-partitioning algorithm for contour cycles

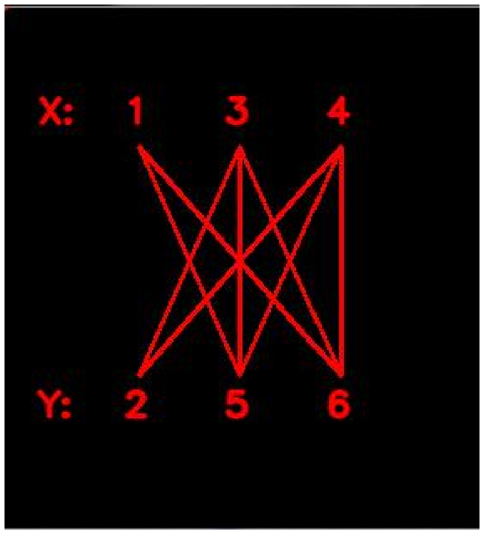


Fig. 9. Region graph bi-partitioned

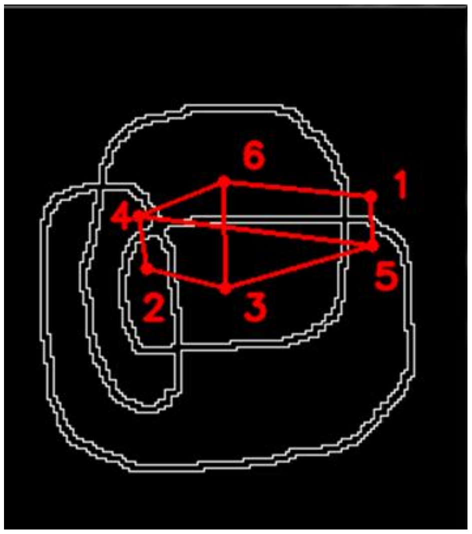


Fig. 10. Region (cycle) adjacency graph

7 Final Procedure to Finish the Maze

With a solution path for a maze, it is easy to construct the maze. Adding random dead end branches to the solution path, maze can be constructed. The final maze to be presented to a player is usually the one with walls thinned. Fig. 14 illustrates one example (for partition X) of the final maze with walls thinned. The small red and green vertices are the points of the start and goal. Fig. 15 is with the solution path.

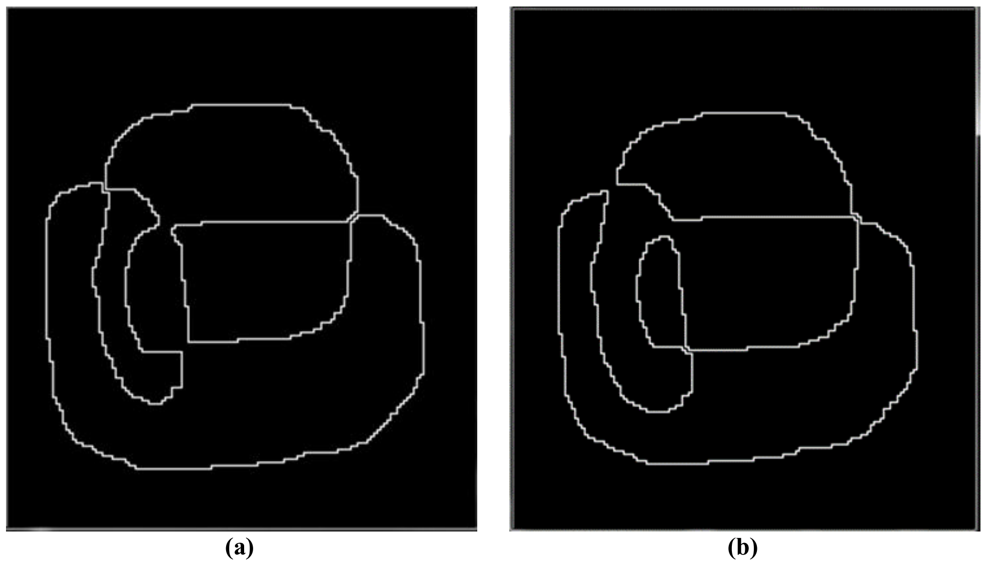


Fig. 11. Merged one single cycle: (a) partition *X*; (b) partition *Y*

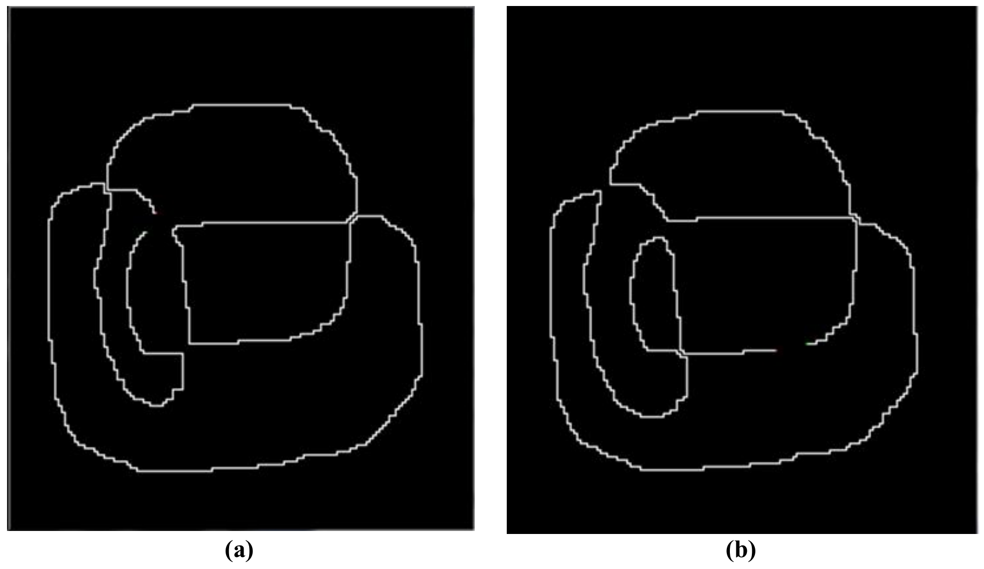


Fig. 12. Maze solution path with start and goal: (a) partition *X*; (b) partition *Y*

8 Additional Experiments and Discussion

8.1 Curves with Odd Degree Vertices

Though it is not an ordinary one stroke curve drawing, a drawing with three curve lines extending from a crossing (Fig. 16) was tried. The contour cycles (Fig. 17) were extracted. Since it is not Eulerian (though semi-Eulerian), bipartite partitioning was not possible.

```
Assumption:
R: initially one of the sets of contour cycles partitioned---X or Y.
M: initially empty; will have the resulting single cycle.
1.  Remove an arbitrary cycle m from R and put it in M and name it m.
2.  Repeat {
3.    For each cycle i in R do {
4.      Find the nearest two points between the two cycles---m and i.
5.      If the distance between m and i is shorter than the predetermined length, do{
          Merge i to m---organizing one single cycle m;
6.        Remove i from R;
7.        If no cycle remains in R, then finish.
8.      }
9.    }
10. }
```

Fig. 13. Merging algorithm

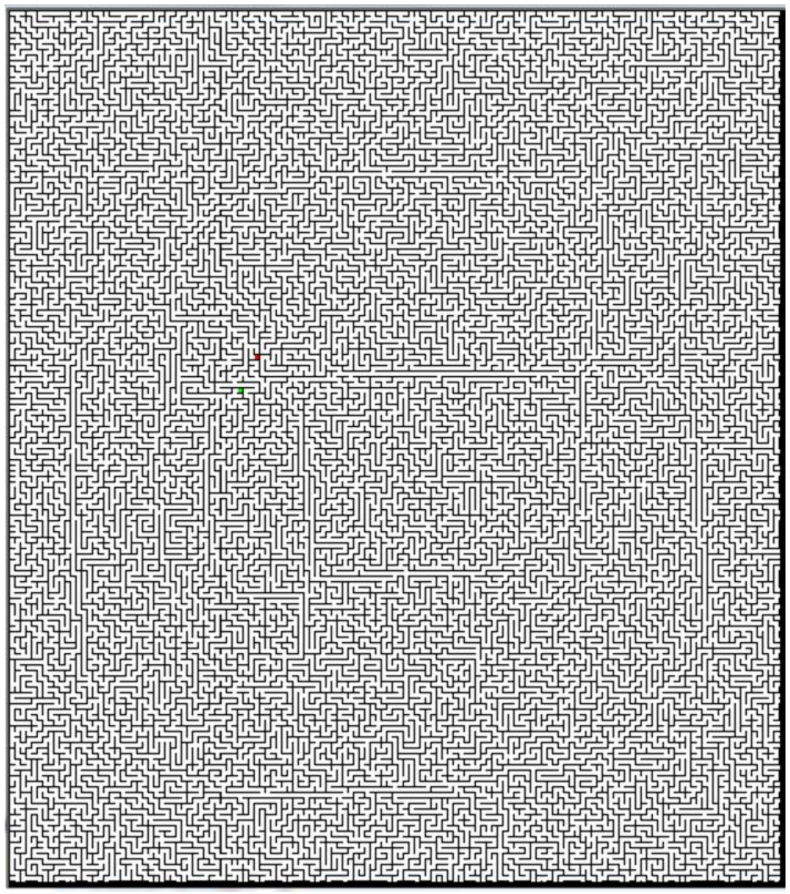


Fig. 14. Final maze with walls thinned: partition *X*

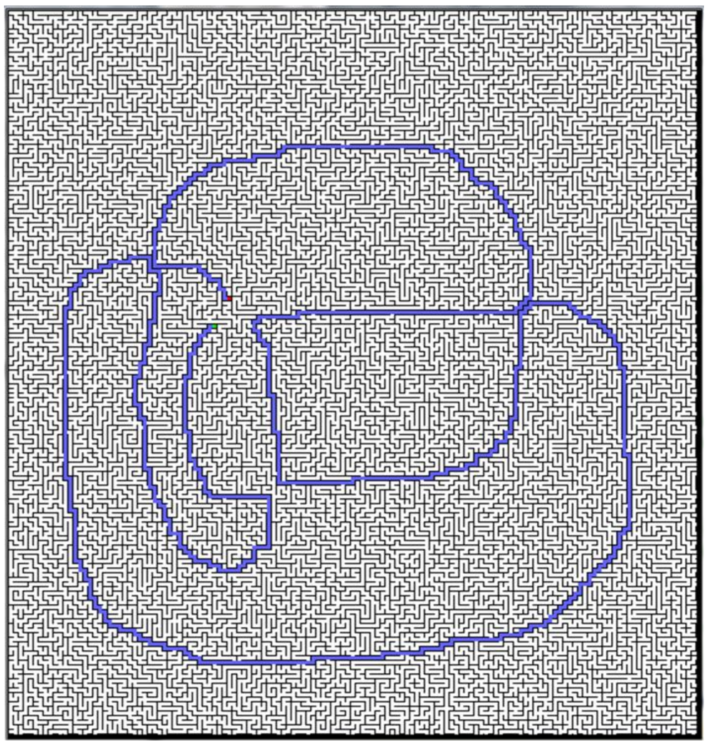


Fig. 15. Final maze with walls thinned: start and goal are colored red and green: partition *X*

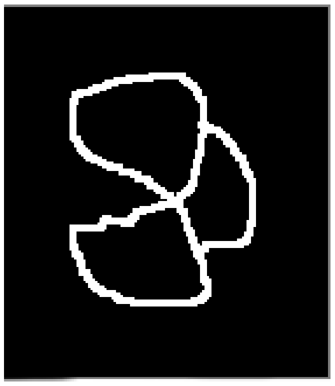


Fig. 16. Drawn curve with two odd degree vertices (intersections)

8.2 More Curves Crossing at a Point (6 Degree Vertex)

A curve with the crossings of three lines at a point (Fig. 18) was tried. Since it is with all vertices having even degree, bipartite partitioning is expected to be possible. In order to try to obtain the maze solution path which looks more similar to the original drawn curve, the line width was set to 2 pixels. Fig. 19 is the extracted contour cycles. Some cycles overlap, but they constitute independent different cycles. Therefore, it is possible to make the separation (bi-partitioning). Fig. 20 (a) and (b) are the results of partitioning; Fig. 21 (a) and (b) are the results of the merge; and Fig. 22 is the final maze with the solution path for partition *Y*. It worked well.

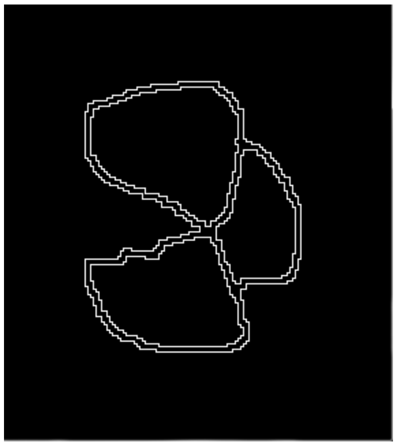


Fig. 17. Cycles extracted from a curve with odd degree intersections

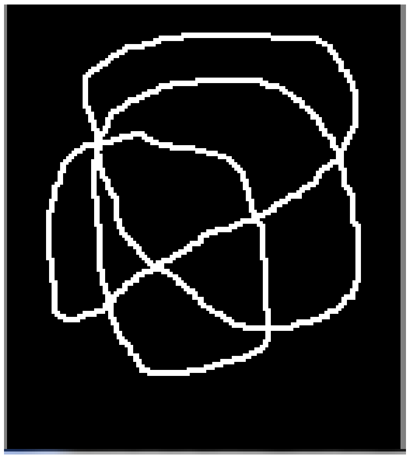


Fig. 18. Line drawing with three line curves crossing at a point

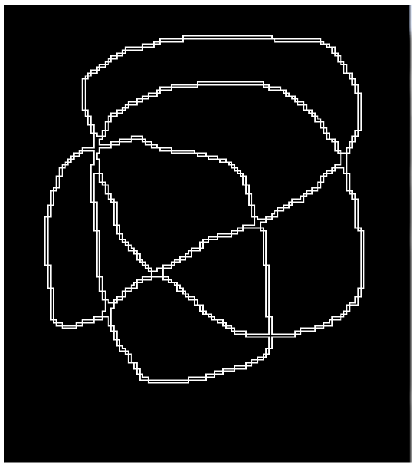


Fig. 19. Contour cycles are extracted

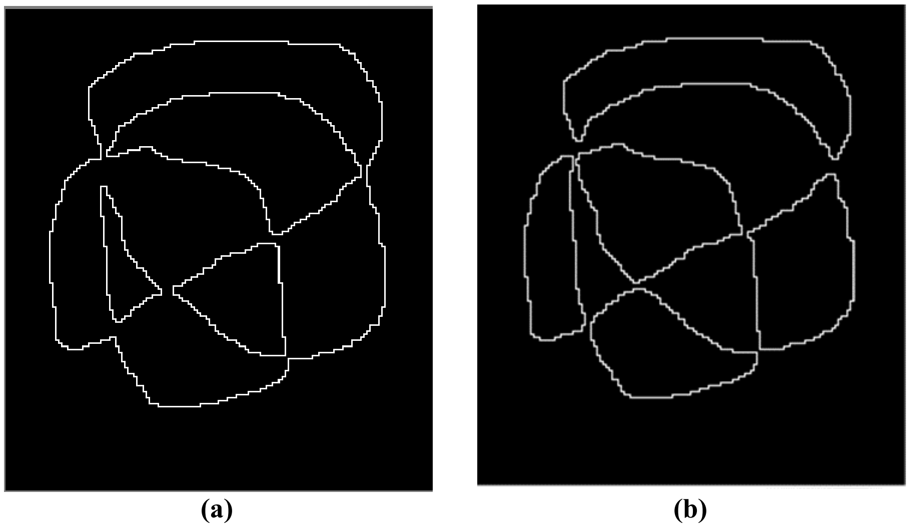


Fig. 20. Set of cycles for three curves crossing at a point: (a) partition *X*; (b) partition *Y*

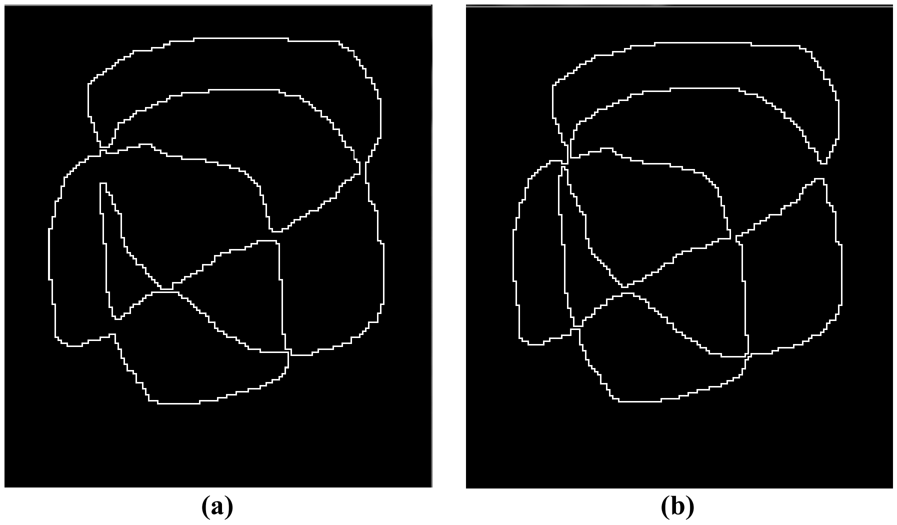


Fig. 21. Merged cycles: (a) partition *X* merged; (b) partition *Y* merged

8.3 More Complex Drawing with Line Width of 2 Pixels

A more complex drawing was tried (Fig. 23). Fig. 24 is the extracted cycles. There are 17 cycles. Figs. 25 (a) and (b) are the results of the partitioning; Figs. 26 (a) and (b) are the results of the merge; and Fig. 27 is the final maze with the solution path for partition *X*.

8.4 Drawing Curve with Line Width of 1 Pixel

Line drawing with line width of 1 pixel was tried (Fig. 28). Fig. 29 is the extracted contour cycles. Most parts of the cycles overlap. There are actually 8 cycles. Those cycles are independent and separate but overlap. Figs. 30 (a) and (b) are the results of partitioning. There are 5 cycles in Figs. 30 (a) and 3 cycles in (b). But in those figures, some parts of the cycles overlap. Accordingly, the same merging procedure used in the previous subsection is not applicable. An additional procedure (cycle separation: elimination of overlapping) was introduced. Figs. 31 (a) and (b) are the results. Both of Figs. 31 (a) and (b) are sets of complete separate and non-overlapped cycles. Then the previous merge procedure becomes applicable for the merge to produce one single cycles. Figs. 32 (a) and (b) are the resulted single cycles. They look alike. At a glance, both look identical. This is very persuasive to say that there are two Eulerian graphs in the extracted contour cycles if the closed one stroke drawn curve is with zero width, which is virtually a mathematically defined graph. Figs. 33 and 34 are the final mazes, both with start and goal. They also look alike.

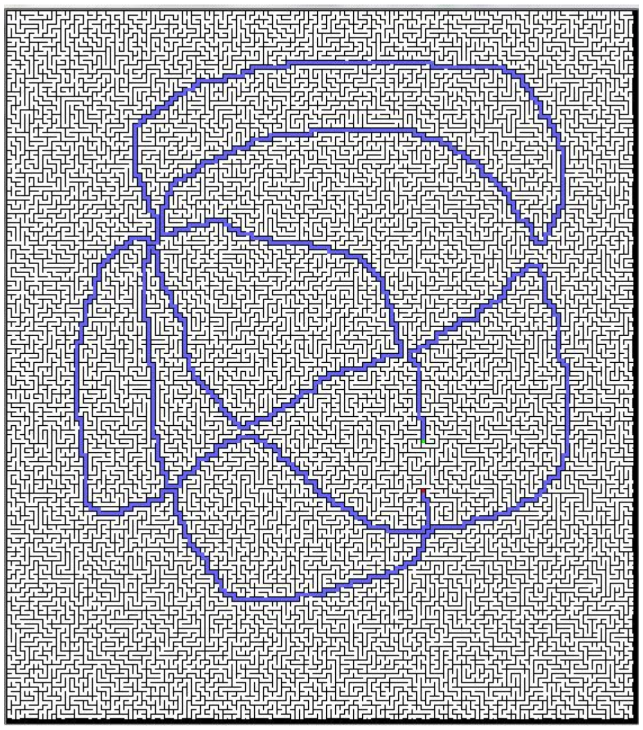


Fig. 22. Final maze with solution path (partition *Y*)

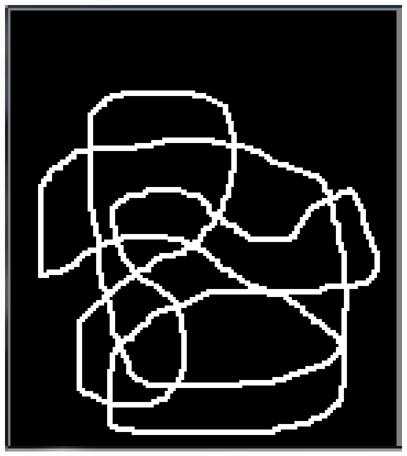


Fig. 23. Complex drawn curve with line width of 2 pixels

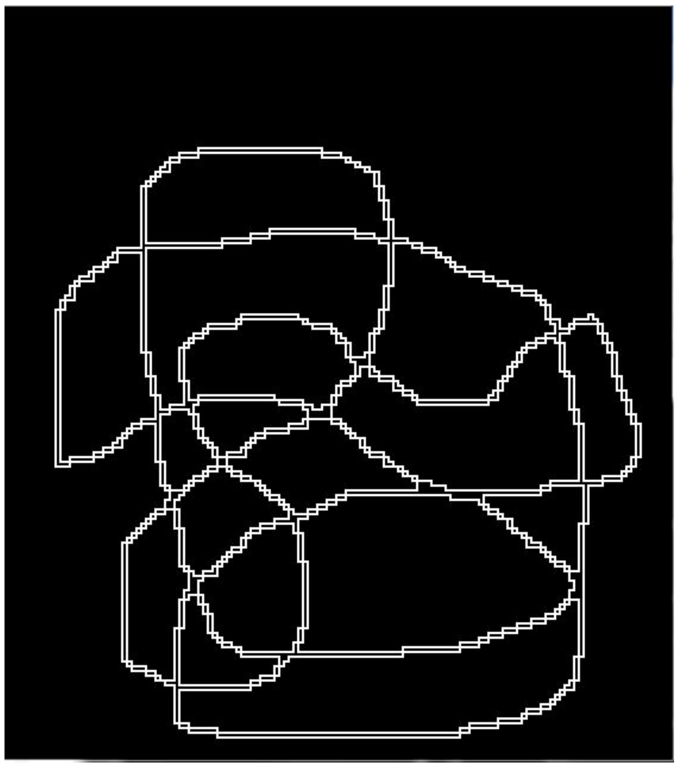
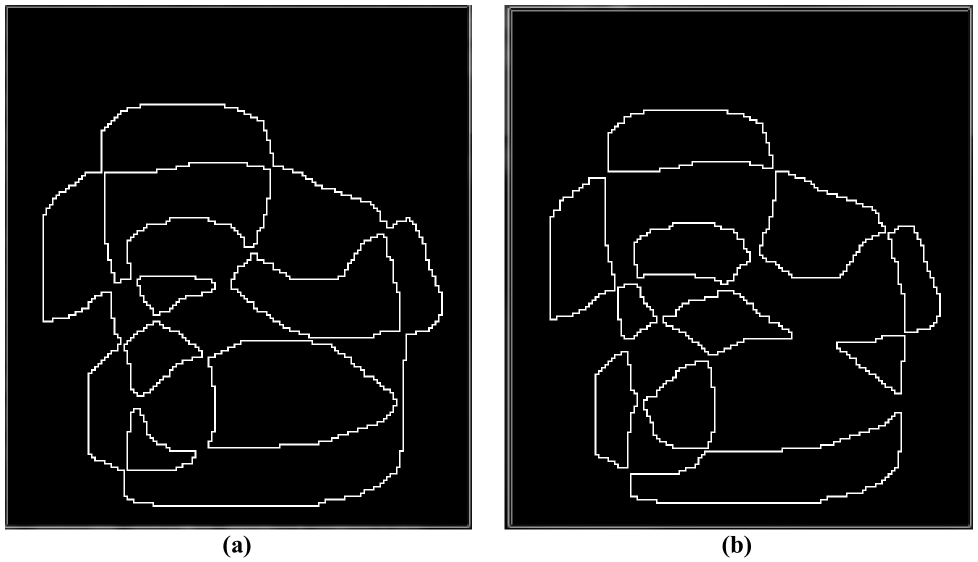


Fig. 24. Cycles are extracted: 17 cycles



(a) **(b)**

Fig. 25. Set of cycles: (a) partition *X*; (b) partition *Y*

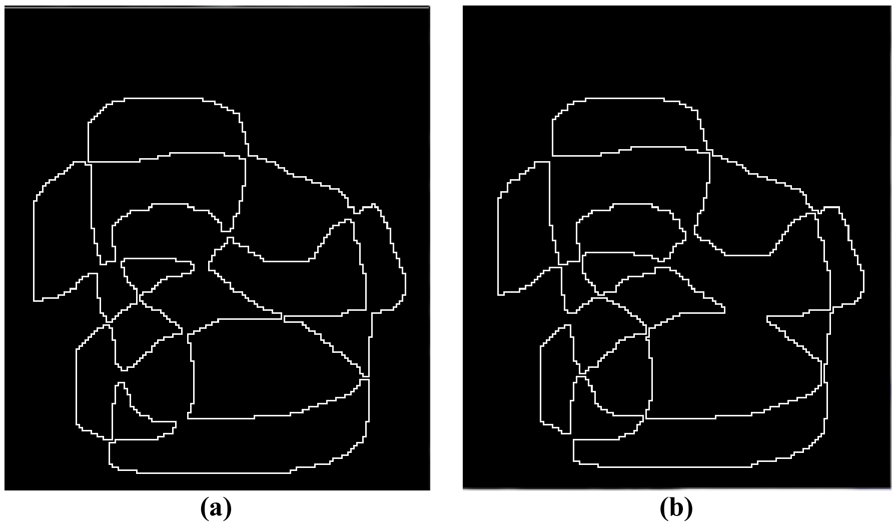


Fig. 26. Merged cycle: (a) partition *X*; (b) partition *Y*

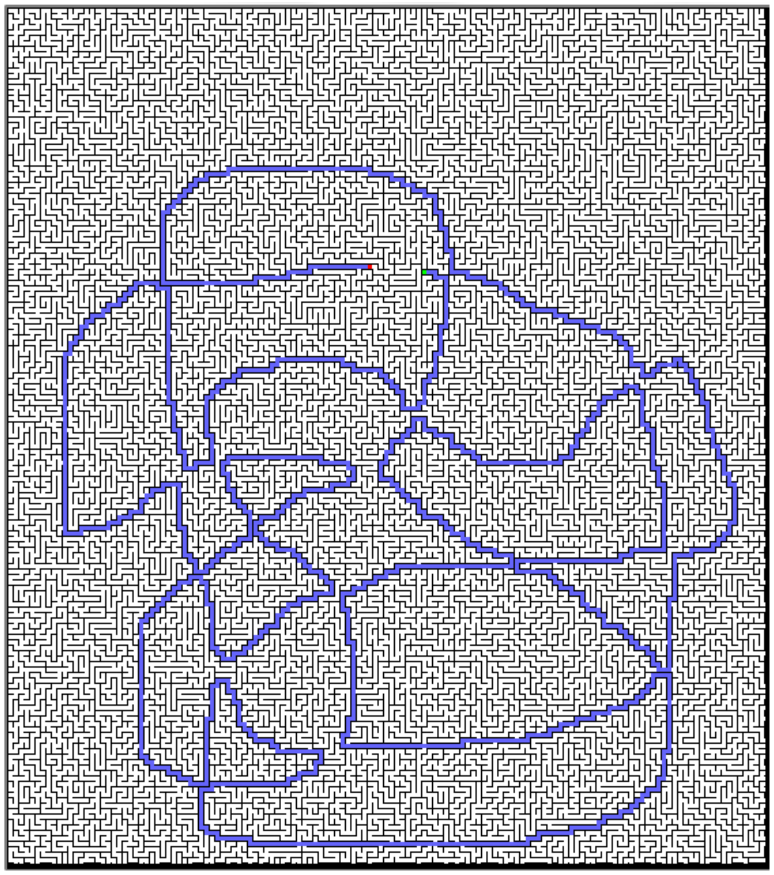


Fig. 27. Final maze with solution path (partition *X*)

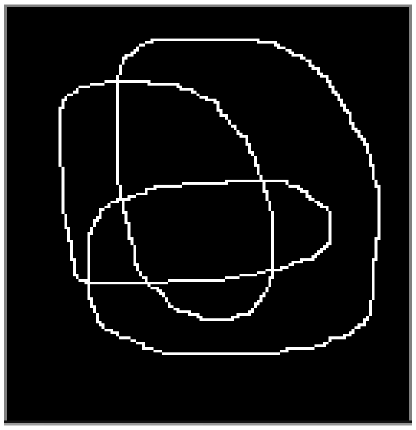


Fig. 28. Line drawing with line width of 1 pixel

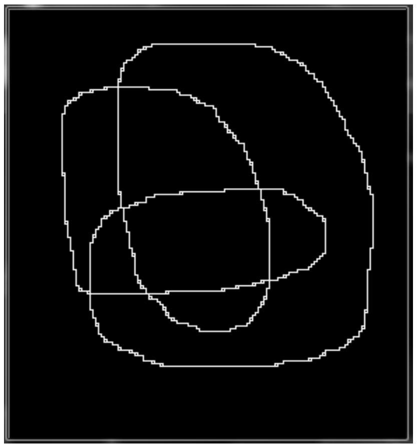


Fig. 29. Cycles are extracted (8 contour cycles), most parts overlap

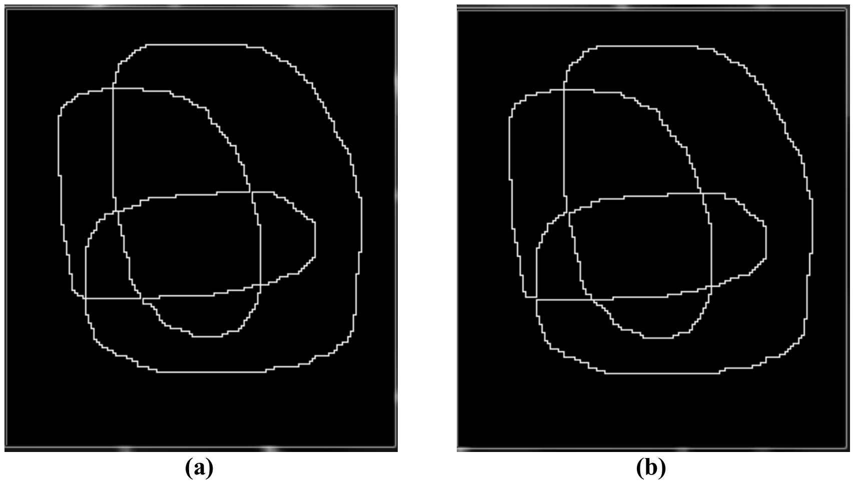


Fig. 30. Set of cycles partitioned---some vertices and edges overlap around some intersections: (a) partition *X*, 5 cycles; (b) partition *Y*, 3 cycles

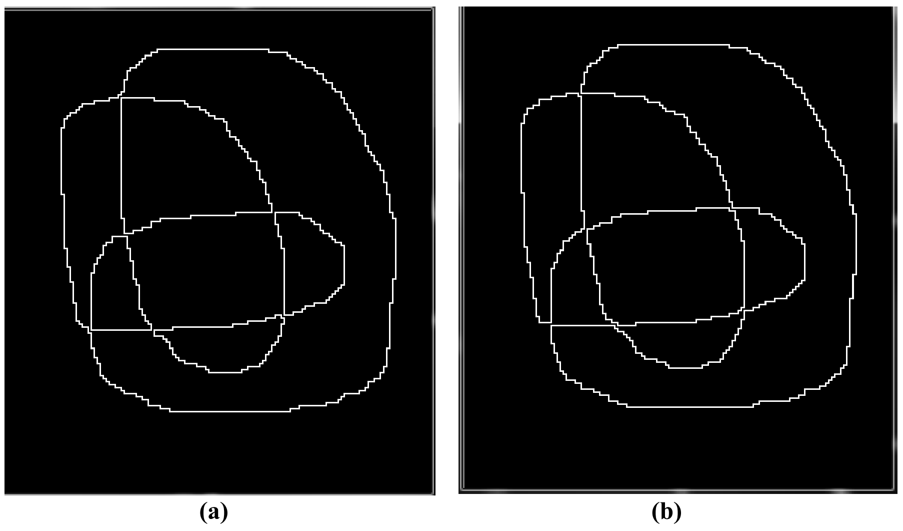


Fig. 31. Cycles are separated to remove overlaps: (a) partition *X*, 5 cycles; partition *Y*, 3 cycles

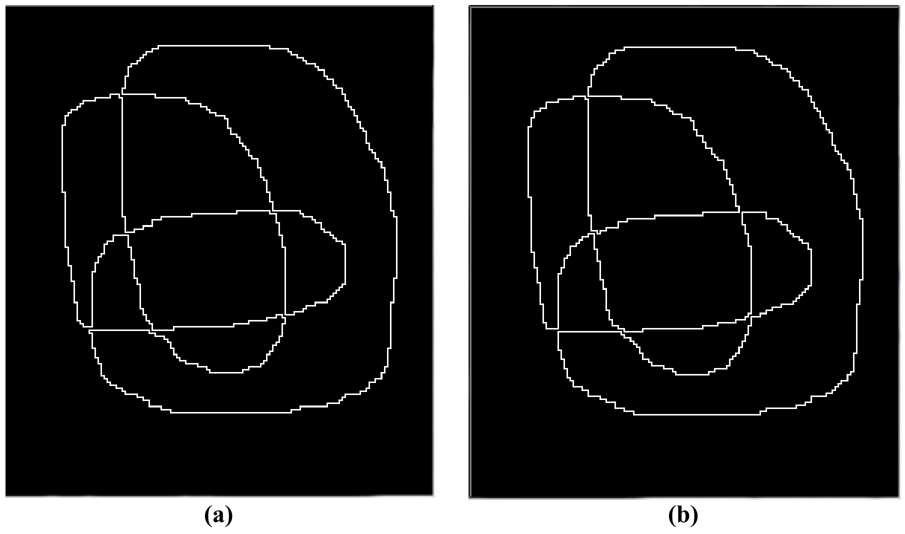


Fig. 32. Cycles are merged: (a) partition *X*; (b) partition *Y*

8.5 Curve Line Width and the Adjacency

Fig. 6 shows the contour cycles which are two parallel paths in most parts. As long as the observation is concerned, the cycle line positions are inside the drawn lines but on the boundary between the drawn curve and the region for the white foreground lines, which are based on the software Open CV [18]. The above fact is true for different curve widths even for the width of one pixel, which is shown in Fig. 28. As long as a boundary exists, contour line is generated. Therefore one curve line always have two contour lines generated on both sides, possibly same positions. When the curve width is one pixel, most parts of the two lines overlap. Two contour cycle lines are presumably generated along the boundary even if its width is zero. They supposedly occupy the same positions. No such boundary actually exists, though.

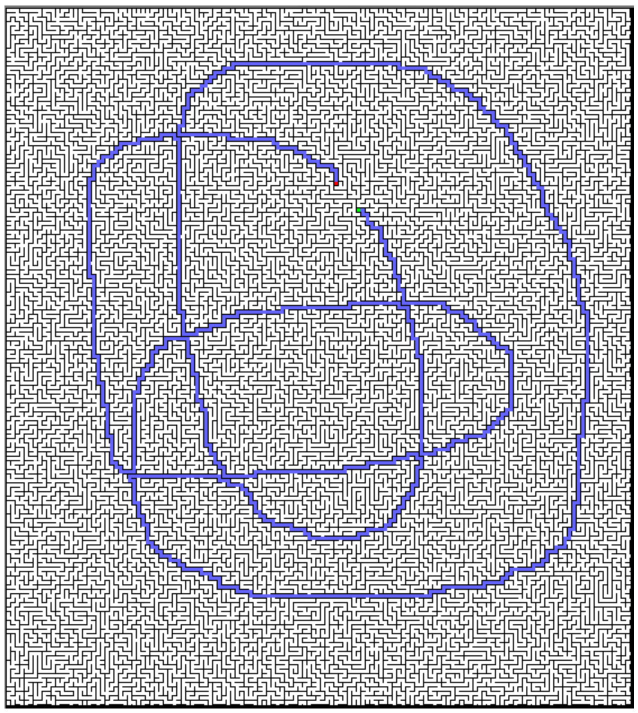


Fig. 33. Final maze for partition *X* with solution and start & goal

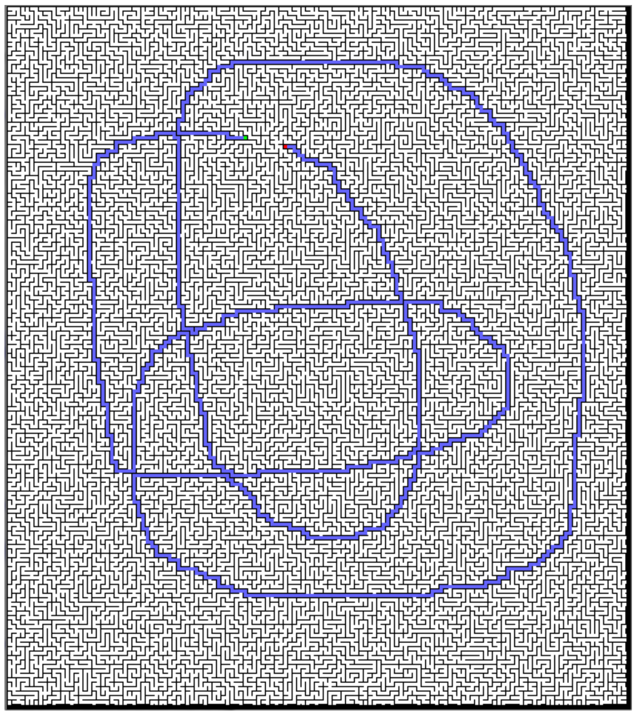


Fig. 34. Final maze for partition *Y* with solution and start & goal

In Section 5, the adjacency between two cycles were checked by the distance between the two cycles. The criteria need not be very strict for most cases. But in the previous experiments, the overall average of shortest distance from a vertex on a cycle to another cycle as (4) is first computed. The values are different depending on the drawn curve especially due to the width of the curve. This distance was used for the criteria in deciding if two cycles are adjacent. The average of shortest distance for Figs. 6, 24 and 29, respectively, is as in Table 1. For the drawn curve line with width of 3 pixel, the boundary lines are two pixel apart on the picture array. When it is converted to the maze structure, the two pixels are apart 4 cells away. For line widths of 2 and 1, they are separate 2 and 0 pixels away, respectively. So the values on Table 1 are considered reasonable. And there is no inconsistency between looking and the computed values. For Fig. 24, the nearest two pixels along the opposing boundary curves are mostly two cells away. But some may be overlapped. For Fig. 29, the two pixels are mostly overlapped but in some part, they are one or two pixels away. So values in Table 1 are reasonable.

Table 1. Average of shortest distance (d_{AS})

	Fig. 6	Fig. 24	Fig. 29
Line width (in pixel) for the original curve	3	2	1
d_{AS}	4.3	1.84	0.51

9 Conclusion

There are two significant remarks for this conclusion. The first is about the theoretical matter and the second is the practical realization of the maze construction using the first.

1) The boundary contours extracted from the closed one stroke drawn curve virtually constitute two Eulerian circuits. They are essentially made up of the identical vertices and edges but constitute different circuits. They constitute two different edge cycle sets, each of which is made of disjoint edge cycle sets of Eulerian graph. The two sets can be separated or bi-partitioned based on the adjacency between the regions surrounded by the cycles. Adjacency is provided by the actually extracted contour cycles. Therefore, the each of the bi-partitioned cycle set can be converted to one single cycle, which is essentially the solution path of the maze. These two maze paths go through all parts of the originally drawn curve.
2) The above theory is demonstrated by a number of experiments---including the bi-partition algorithm and the merge procedure. The thinner the originally drawn curve, the closer looking to the original curve the solution path of the maze becomes.

Competing Interests

Author has declared that no competing interests exist.

References

[1] Maze generation algorithm. Available:http://en.wikipedia.org/wiki/Maze_generation_algorithm, (Accessed 2013/01/09)

[2] Hosokawa T. Maze. Available:http://aanda.system.to/ (Accessed 2013/01/25)

[3] Xu J, Kaplan CS. Image-guided maze construction. ACM Transactions on Graphics. 2007;26(3):1-9. Article 29.

[4] Xu J, Kaplan CS. Vortex maze construction. Journal of Mathematics and the Arts. 2007;1(1):7-20.

[5] Conceptis puzzles. Maze-a-PiX. Available:http://www.conceptispuzzles.com/ (Accessed 2013/01/09)

[6] Okamoto Y, Uehara R. How to make a picturesque maze. Proc. Canadian Conf. on Computational Geometry. 2009;137-140.

[7] Hamada H. Available:http://www.lab2.kuis.kyoto-u.ac.jp/~itohiro/Games/100301/100301-11.pdf (Accessed 20013/01/12)

[8] Hamada K. A picturesque maze generation algorithm with any given endpoints. Journal of Information Processing. 2013;21(3):393-397.

[9] Kurokawa T. Picture maze generation by successive insertion of path segment. British Journal of Mathematics and Computer Science. 2014;4(24):3444-3463. Science Domain International. ISSN:2231-0851, Available:http://www.sciencedomain.org/issue.php?iid=699&id=6

[10] Ikeda K, Hashimoto J. Stochastic optimization for picture maze generation. Jouhou Shori Gakkai Ronbunshi. 2012;53(6):1625-2634. In Japanese

[11] Kurokawa T, Mori K, Mizuno T. Automatic construction of picture maze by repeated contour connection. Thailand-Japan Joint Conference on Computational Geometry and Graphs. 2012;5-6.

[12] Kurokawa T. Picture maze generation and its relation to graphs. Shikoku-Section Joint Convention Record of the Institute of Electrical and Related Engineers. 2013;231.

[13] Kurokawa T. Picture maze generation by repeated contour connection and graph structure of maze. Computer Science and Engineering. 2013;3(3):76-83. Scientific & Academic Publishing, USA. ISSN: 2163-1484. Available:http://article.sapub.org/10.5923.j.computer.20130303.04.html (Accessed 2014/1/20)

[14] Wong FJ, Takahashi S. Flow-based automatic generation of hybrid picture mazes. Computer Graphics Forum. 2009;28(7):1975-1984. Wiley-Blackwell.

[15] Ishida S. Algorithm of automatic maze generation. Available:http://www5d.biglobe.ne.jp/%257estssk/maze/make.html, (Japanese) Accessed 2013/10/8

[16] How to Build a Maze. Available:http://www.mazeworks.com/mazegen/mazetut/index.htm, Accessed 2013/1/26

[17] Available:http://www.microsoft.com/ja-jp/dev/express/default.aspx

[18] Available:http://jaist.dl.sourceforge.net/project/opencvlibrary/opencv-win/2.3/ Accessed Oct., 2011

[19] Matousek J, Nesetril J. Invitation to discrete mathematics. Second Edition, Oxford University Press, New York; 2009.

[20] Microsoft Corporation. Microsoft paint. Windows 7 Professional; 2009.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://sciencedomain.org/review-history/9770