

Maze Construction by Using Characteristics of Eulerian Graphs

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 to be a vertex and the curve between the two crossings (vertices) is 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 done with 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 in the research that the player obtains the solution path by filling up all the path cells on the maze and by painting the picture region. Hamada [7] and [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 a method to generate a picture maze by making use of the picture contours. In this method, all the contours are

40 extracted and are incorporated into the picture maze. It is a very flexible method and can be
 41 applicable to a wide range of binary pictures. F. J. Wong and S. Takahashi [14] constructed a
 42 hybrid picture maze from two photograph images. This maze is not based on square grids like
 43 the mazes but on the special grids computed from the image. Two photo pictures are
 44 overlapped in the maze. The maze player depicts an outline of one picture on the other picture
 45 background.
 46

47 2. MAZE STRUCTURE

48
 49 There are a number of different representation structures of a square grid maze. This paper
 50 uses one of them and is explained in this section. A computer constructed maze is usually
 51 represented by a two-dimensional array. Such a maze consists of path and wall. A path cell is
 52 either a vertex or an edge (of a graph). Fig. 1 is a sample maze array of size 7x7---where '⊙'
 53 is a vertex; 'e' is an edge, and 'W' and 'w' are walls. A vertex is placed one in every two cells
 54 horizontally and vertically. An edge represents the connection between the two vertices.
 55 Diagonally adjacent cells to a vertex are wall cells; there are four such cells around a vertex;
 56 the vertically or horizontally adjacent cells to a vertex are edge or wall cells. If the cell
 57 represents the connection between the two vertices, then it is an edge cell; otherwise it is a
 58 wall cell.

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

66 If the walls are thinned [15], the maze becomes as presented in Fig. 2. This maze, to be
 67 presented to a player, actually consists of two kinds of cells: path and wall. However, it
 68 represents the corresponding graph with vertices and edges. There are other maze
 69 representations [16]. In short, a maze is said to be a collection of vertices and edges like a
 70 graph. However, more restrictions are imposed on a maze: physical positions of vertices and
 71 edges; the number of edges to be connected to a vertex and the like.
 72

	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

	1	3	5	7
1	S			
3				
5				
7				G

Fig. 2. Maze to be presented to a player

73
 74
 75

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 a 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 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 to 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

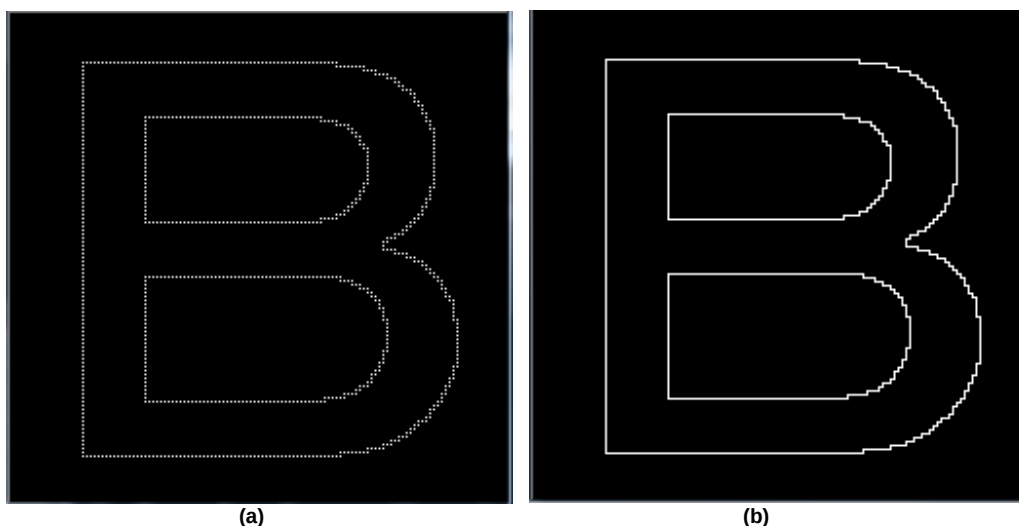


Fig. 4 Contour cycles with maze path structure: (a) dotted contour cycles; (b) Contour cycles with maze structure

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

$$E = \bigcup_{i=1}^n E_i. \quad (1)$$



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, it is possible 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 some ideal situation---there is no width for the drawn curve. Then every part of the curve should be overlapped by parts of 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.

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:

$$(x(j, i), y(j, i)), \text{ where } 1 \leq j \leq c, 1 \leq i \leq n_j \text{ and} \quad (2)$$

c is the number of cycles extracted; 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 certain number, say $n=5$, of vertices on j such that the shortest distance from a vertex to the cycle i is less than the overall average of shortest distance (d_{AS}). The shortest distance from the vertex l on cycle j to cycle i is defined as:

$$d_s(j, l, i) = \min_{1 \leq k \leq n_i} d(j, l, i, k) \quad , \text{ and} \quad (3)$$

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

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

$j \neq i$; n_j is the number of vertices on cycle j . So if there exist more than $(n=5)$ different l on cycle j , two cycles j and i are considered to be adjacent:

$$\{l | 1 \leq l \leq n_j, d_s(j, l, i) \leq a * d_{AS}, a = 2\} \quad (6)$$

The result of this bi-partitioning result is shown in Fig. 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---look 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, Fig. 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 work 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 Fig. 7 (a) and (b) correspond to the edge set of expression (1).

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 these cycle set are merged into one single cycle, then it can be the solution path of a maze which goes through every part of the original curve drawn.

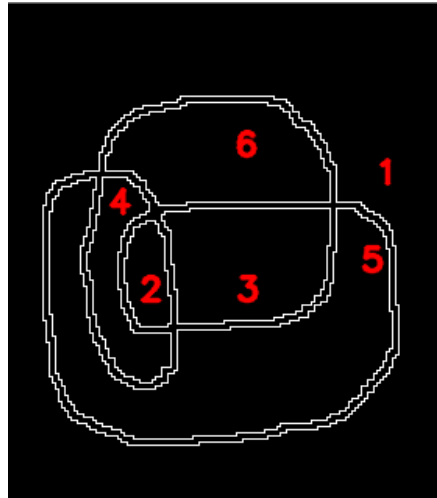
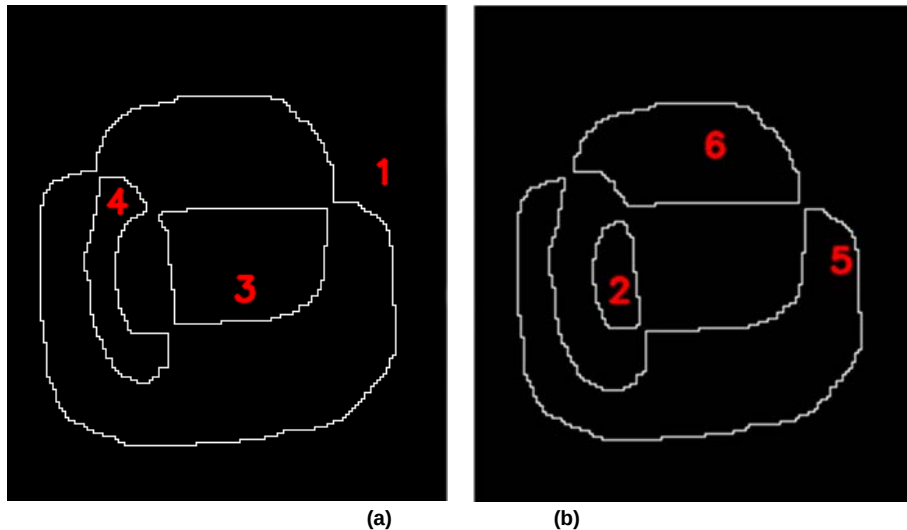


Fig. 6 Contour cycles constructed with region (cycle) numbers.



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

220

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 to 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 in Y.
7. If no cycle remains in R, then finish.
8. }
9. }
10. }
11. Exchange X and Y.
12. }

221

Fig. 8 Bi-partitioning Algorithm for contour cycles

222

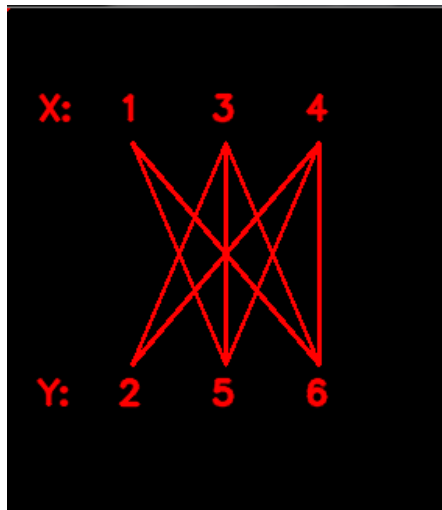


Fig. 9 Region graph bi-partitioned.

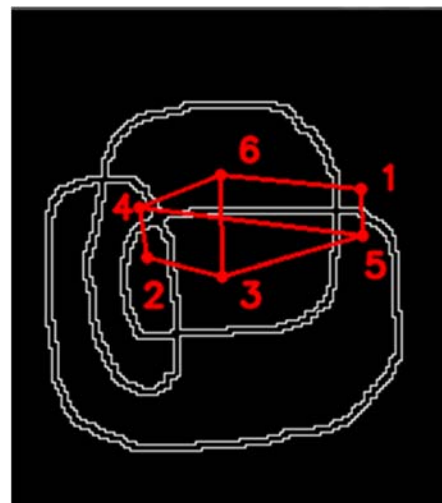


Fig. 10 Region (cycle) adjacency graph

223

224

225

6. MERGE OF CONTOUR CYCLES TO CONSTRUCT ONE SINGLE CYCLE

226

227

228

229

230

231

232

233

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. 7 (a) and (b); the intersection vertices do not usually exist; but the intersections are supposed to exist around the positions where the original curve intersects. By selecting those places and connecting the cycles repeatedly, it is possible to construct one single cycle.

234 A single cycle can be obtained by merging the cycles in Fig.7 (a), X part, as in Fig. 11 (a). The
 235 cycles in Fig. 7 (b), Y part, also can be merged in to one single cycle as Fig. 11 (b). Both of
 236 those merged cycles go through all the parts of the original curve. If a part of the cycles is cut
 237 and removed then the two ends can be the start and the goal of the solution path of the maze.
 238 See Fig. 12 (a) and (b). The merging algorithm is given in Fig. 13.
 239

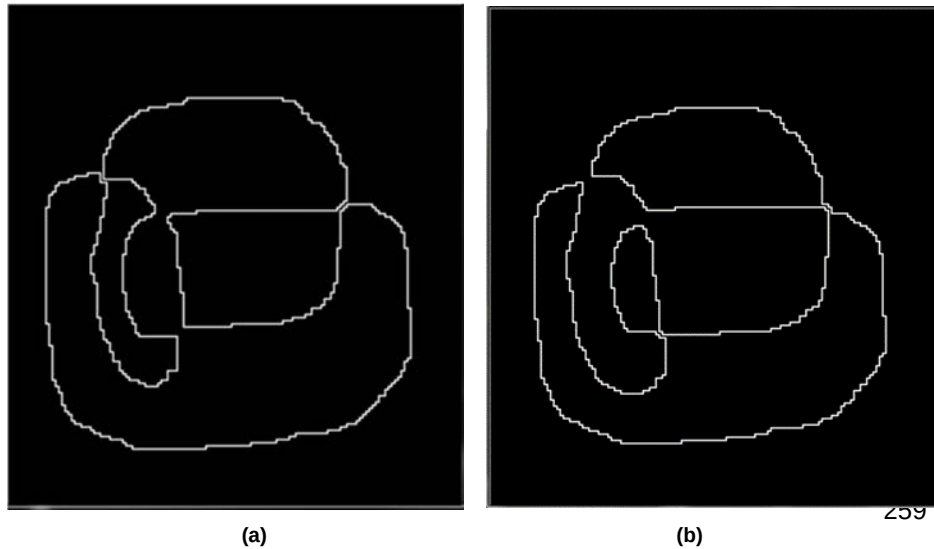


Fig. 11 Merged one single cycle: (a) X part; (b) Y part

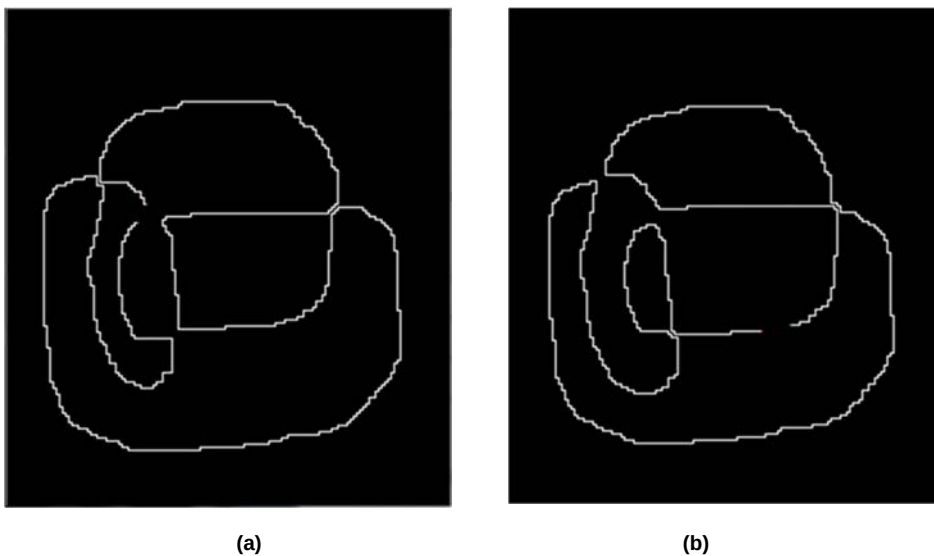


Fig. 12 Maze solution path with start and goal: (a) X part; (b) Y part

Assumption:

R: initially one of the sets of contour cycles partitioned---one of 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{
6. Merge i to m---organizing one single cycle m;
7. Remove i from R;
8. } If no cycle remains in R, then finish.
9. }
10. }

Fig. 13 Merging Algorithm

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 the part X) of the final maze with walls thinned. Fig. 15 is with the solution path.

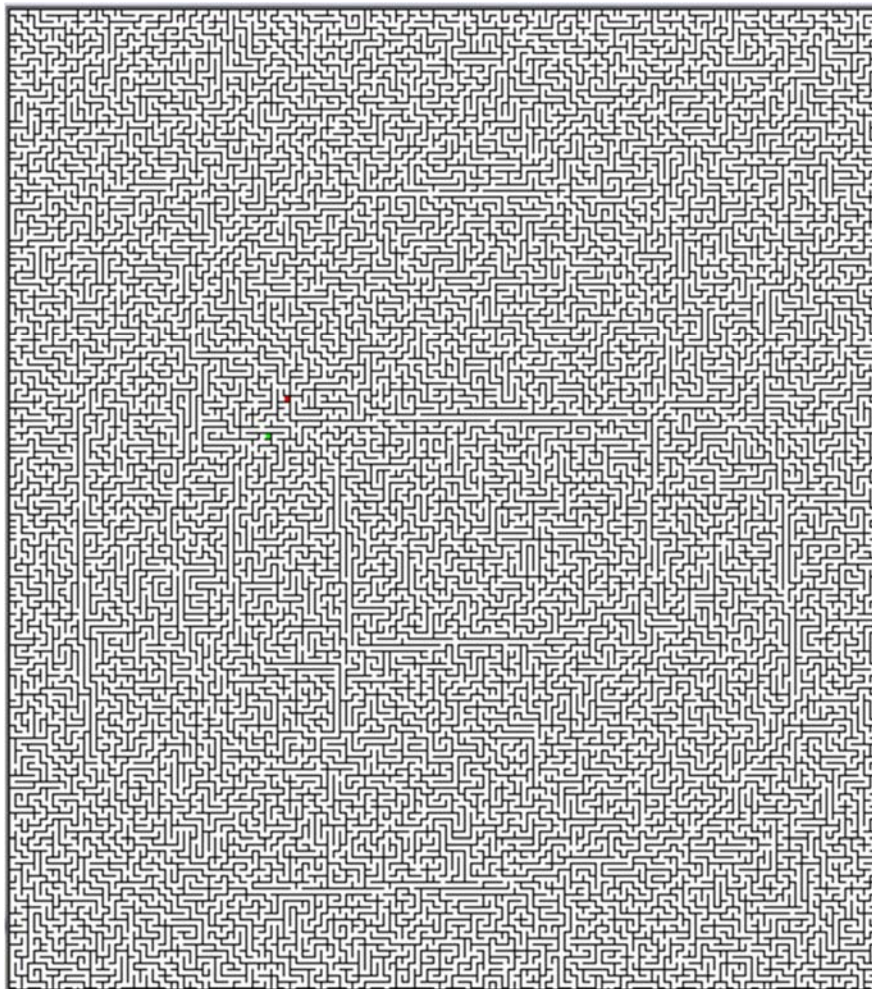
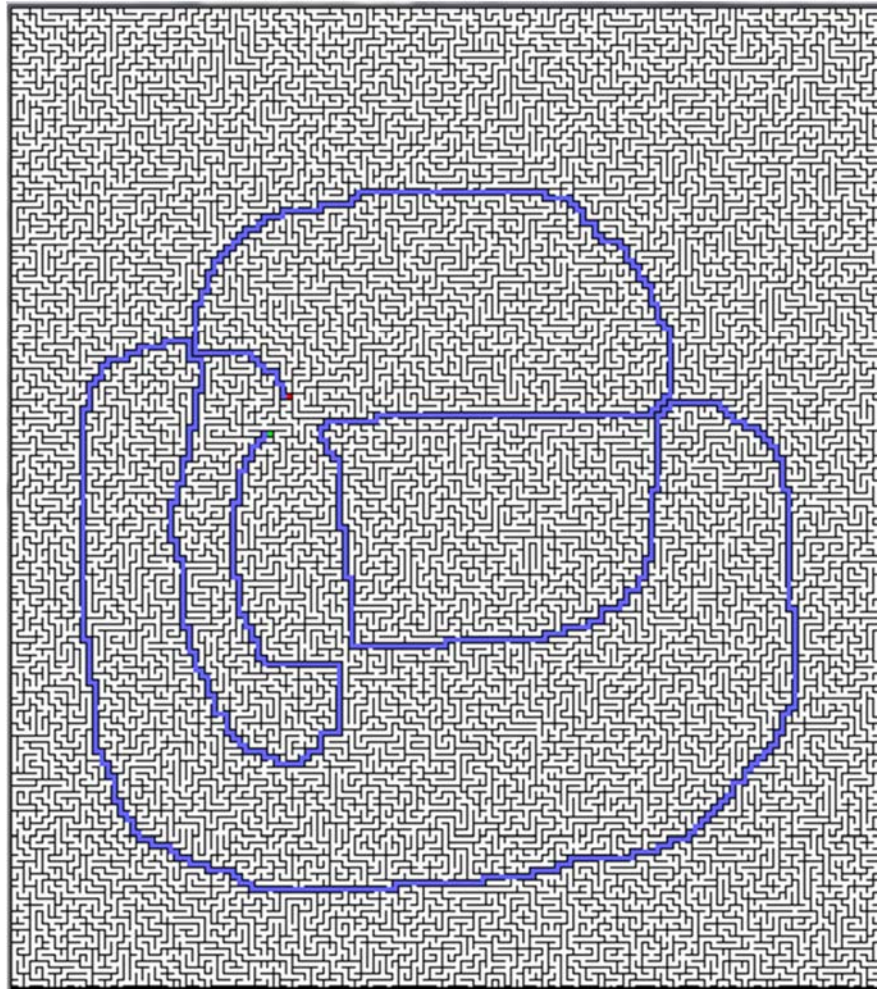


Fig. 14 Final maze with walls thinned: X part

279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298

299
300
301



302
303
304
305
306
307
308
309
310
311
312
313
314
315

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

316 8. ADDITIONAL EXPERIMENTS AND DISCUSSION

317 8.1 Curves with Odd Degree Vertices

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

321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331



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

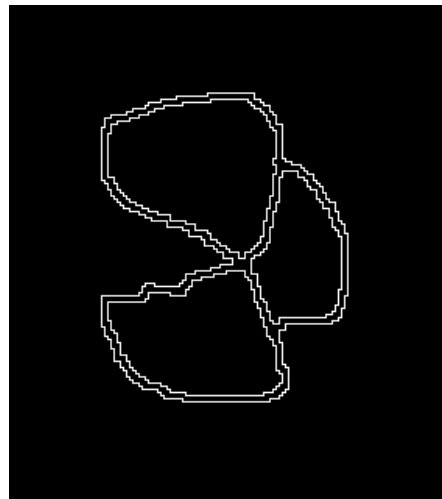


Fig. 17 Cycles extracted from a curve with odd degree intersections

332
 333
 334
 335
 336

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

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

344
 345
 346

347
348
349
350
351
352
353
354
355
356
357
358



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

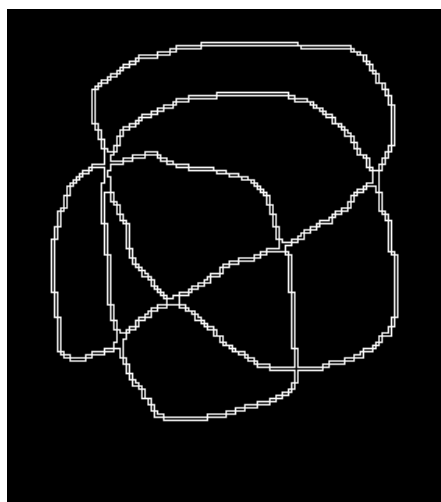
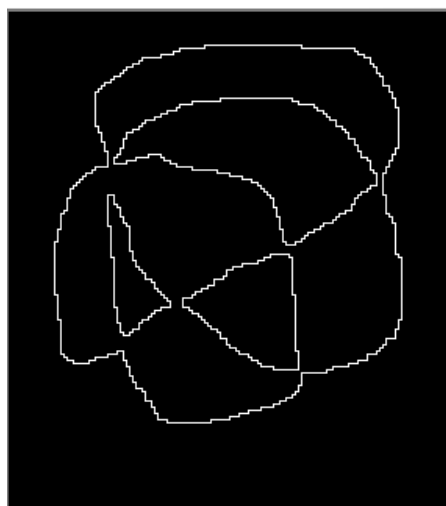
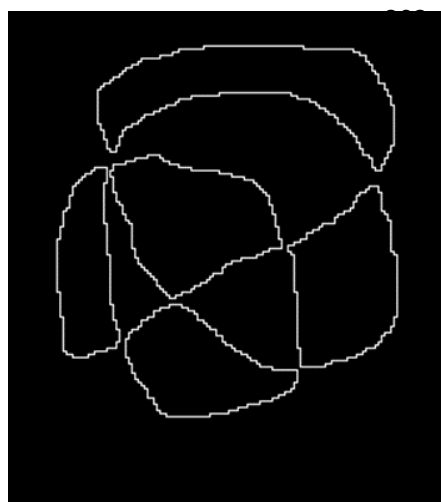


Fig. 19 Contour cycles are extracted

359
360
361



(a)

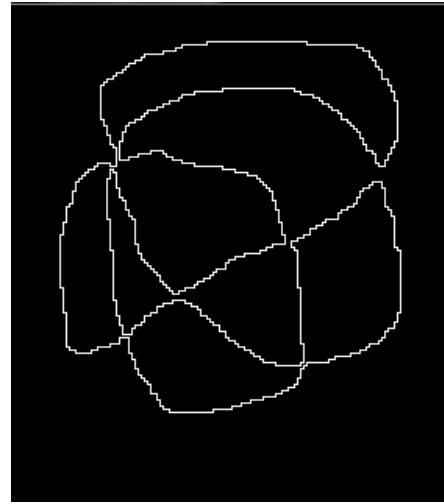
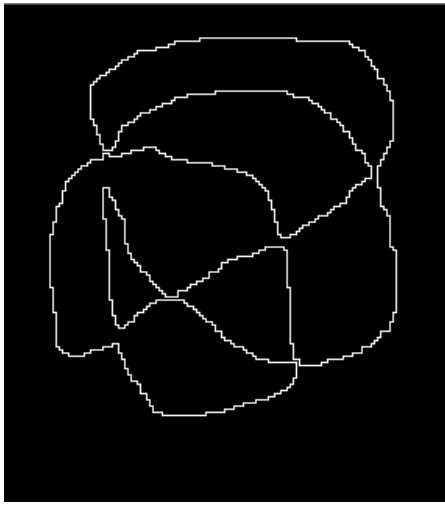


(b)

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

377
378
379

376



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

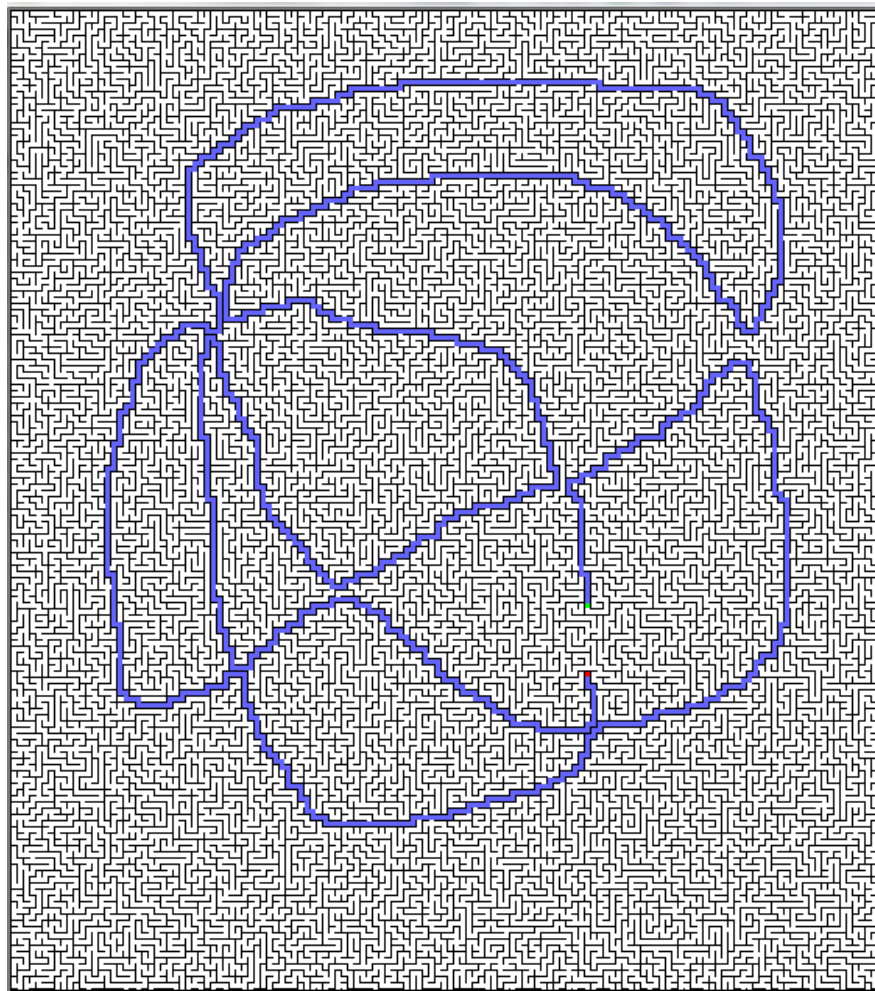


Fig. 22 Final maze with solution path (partition Y)

8.3 More Complex Drawing with Line Width of 2 Pixels

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



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

396

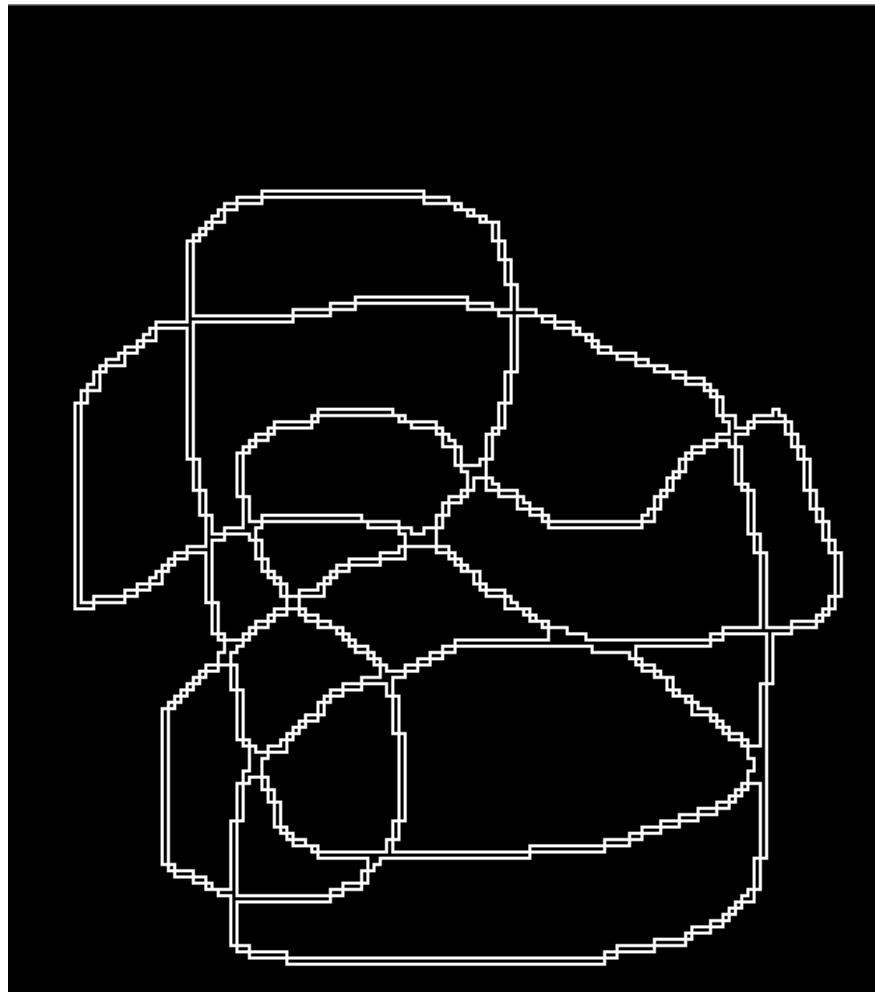
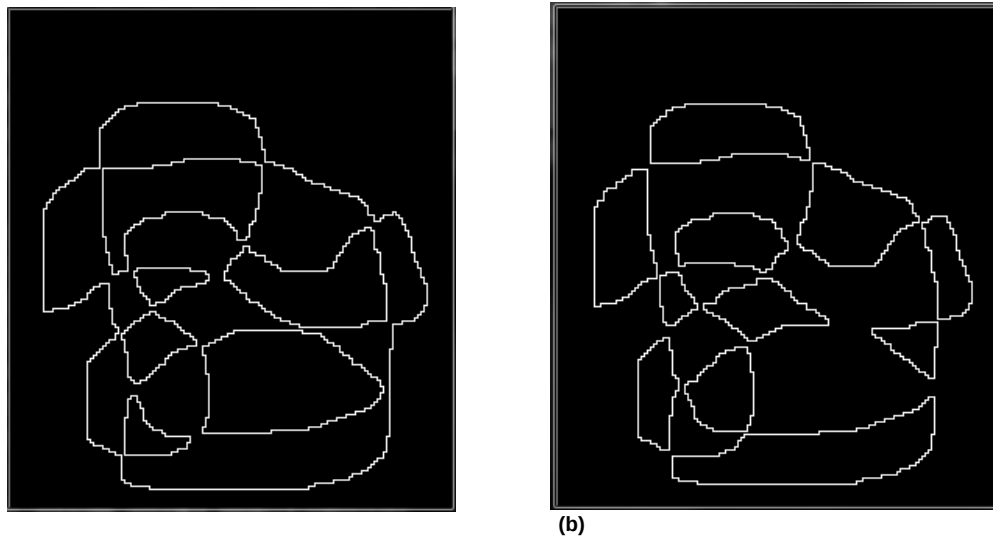


Fig. 24 Cycles are extracted: 17 cycles

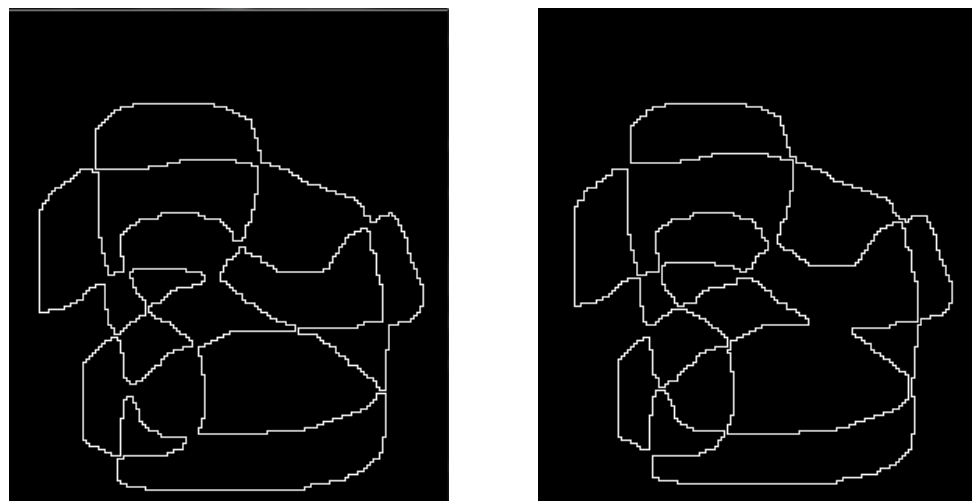
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416

417
418
419



420
421
422
423

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



424
425
426
427
428
429
430
431
432
433
434
435
436

Fig. 26 Merged cycle (a) Partition X; Partition Y

437

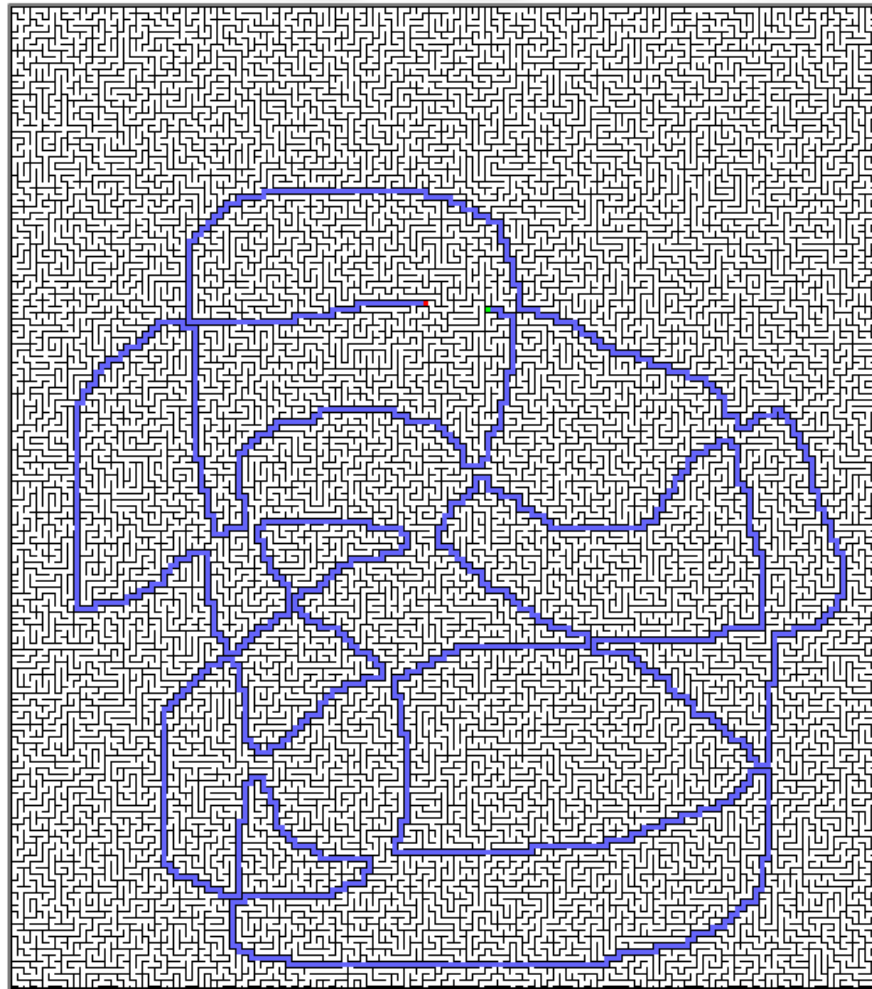


Fig. 27 Final maze with solution path (Partition X)

438

439

440

441

442

8.4 Drawing Curve with Line Width of One PIXEL

443

444

445

446

447

448

449

450

451

452

453

454

455

456

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. Fig. 30 (a) and (b) are the results of partitioning. There are 5 cycles in Fig. 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: avoidance of overlapping) was introduced. Fig. 31 (a) and (b) are the results. So Both of Fig. 31 (a) and (b) are sets of complete separate and non-overlapped cycles. Then the previous merge procedure is applicable for the merge to produce one single cycles. Fig. 32 (a) and (b) are the resulted single cycles. They look alike. At a glance, both seems to be 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.

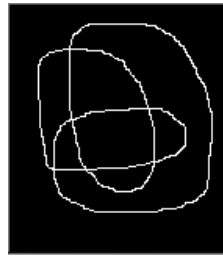


Fig. 28 Line drawing with line width of 1 pixel

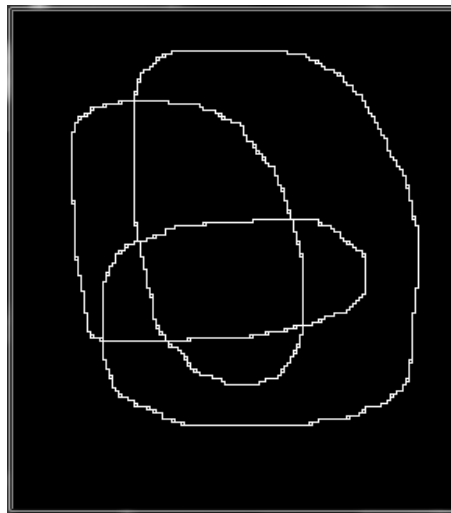
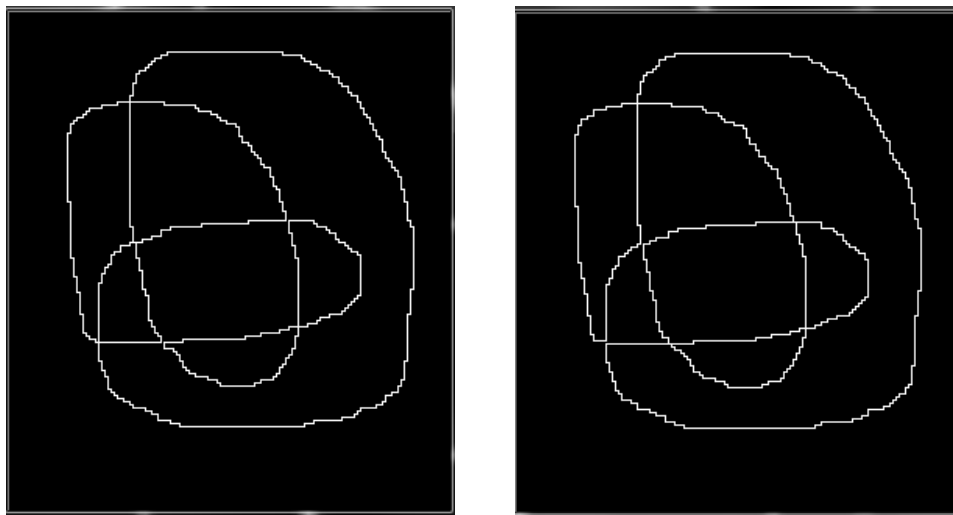
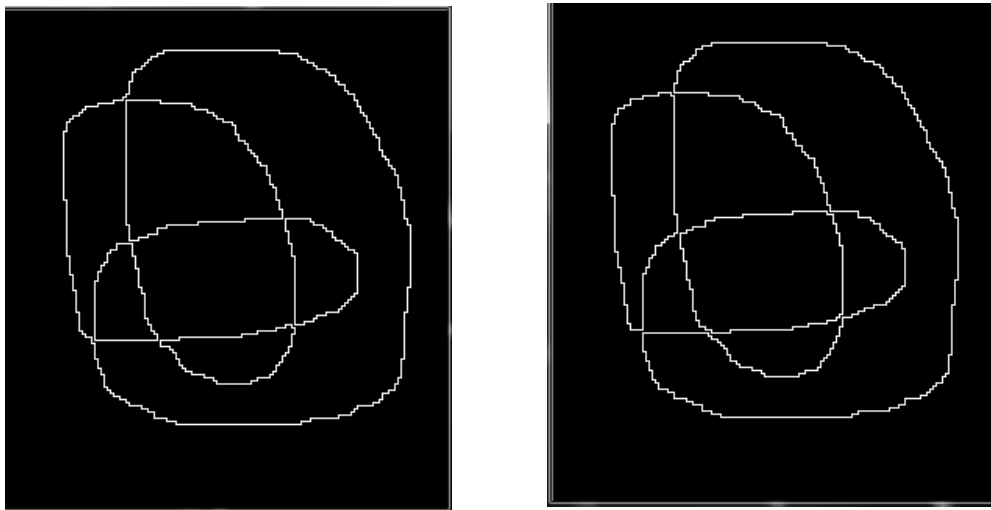


Fig. 29 Cycles are extracted (8 contour cycles), most parts overlap



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

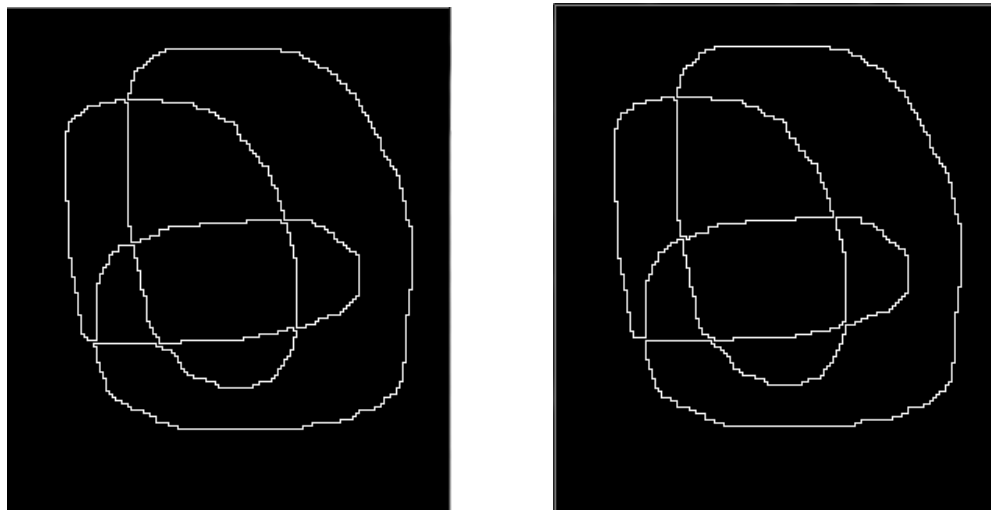
490
491



(b)

Fig. 31 Cycles are separated to avoid overlaps: (a) Partition X, 5 cycles; Partition Y, 3 cycles

492
493
494
495



(b)

Fig. 32 cycles are merged; (a) partition X; merged; (b) partition Y

496
497
498
499
500
501
502
503
504
505
506
507
508

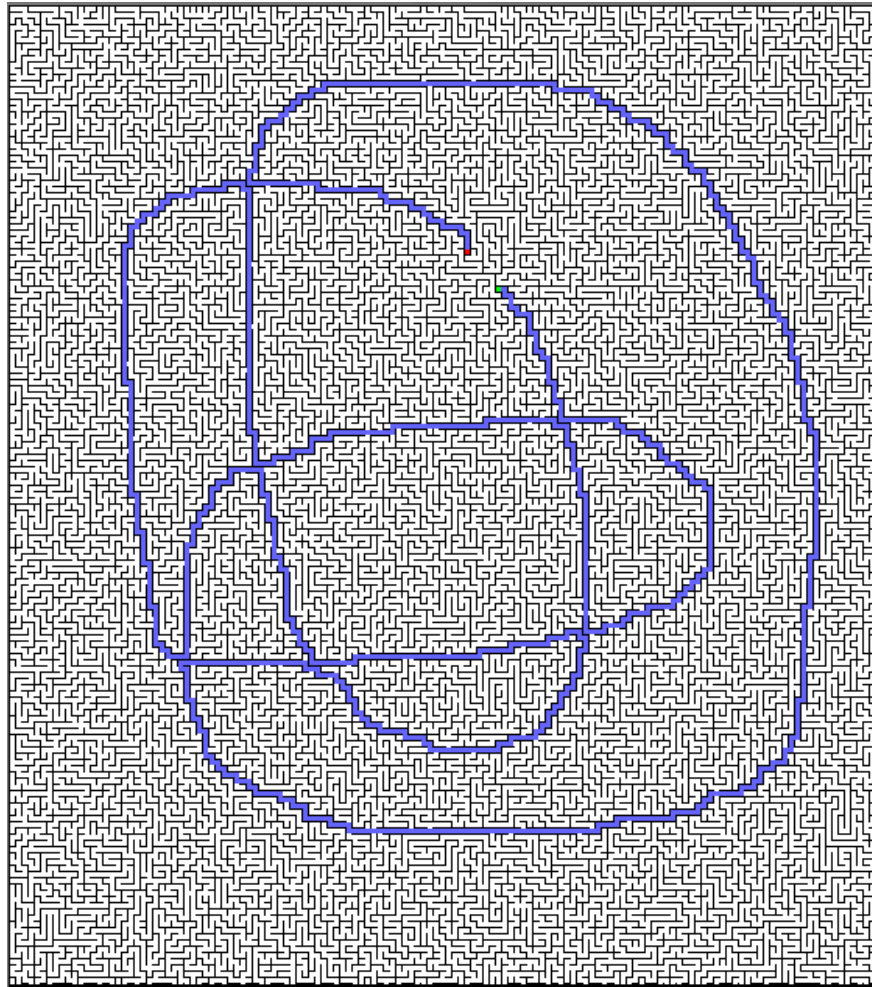
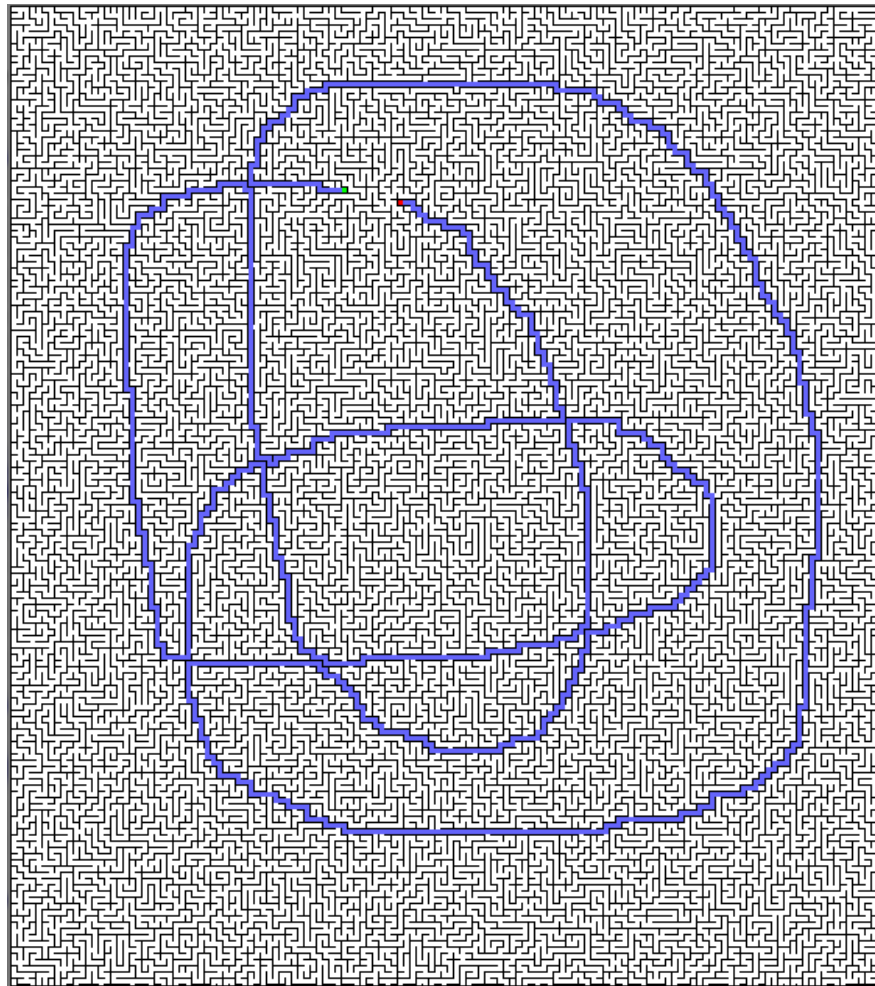


Fig. 33 Final maze for partition X with solution and start & goal

509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529

530
531
532
533
534
535
536



537
538
539
540

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

8.5 Curve Line Width and the Adjacency

541
542
543
544
545
546
547

Fig. 6 shows the contour cycles which are two parallel paths in most parts. As long 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 OpenCV[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 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. No such boundary actually exists, though.

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 value is 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 separate on the picture array. When it is converted to the maze structure, the two pixels are separate 4 cells away. For line widths of 2 and 1, they are separate 2 and 1 pixels away, respectively. So the values on Table 1 are considered to be 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 inTable 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 collusion. 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 essentially 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 cycle. Adjacency is provided by the actually extracted contour cycles. Therefore the each of the bi-partitioned cycle set can be converted to one single path, which is essentially a solution path of a maze. These two maze path go through all parts of the originally drawn curves.
- 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.

References

- [1] "Maze generation algorithm," http://en.wikipedia.org/wiki/Maze_generation_algorithm, Accessed 2013/01/09
- [2] T. Hosokawa, "Maze," <http://aanda.system.to/> Accessed 2013/01/25
- [3] J. Xu and C. S. Kaplan, "Image-Guided Maze Construction," ACM Transactions on Graphics, 26, 3, Article 29, pp.1-9, 2007
- [4] J. Xu and C. S. Kaplan, "Vortex maze construction," Journal of Mathematics and the Arts, 1, (1), pp.7-20, 2007
- [5] Conceptis puzzles, "Maze-a-PiX," <http://www.conceptispuzzles.com/>, accessed 2013/01/09
- [6] Y. Okamoto and R. Uehara, "How to make a picturesque maze," Proc. Canadian Conf. on Computational Geometry, pp.137-140, 2009

- [7] H. Hamada, <http://www.lab2.kuis.kyoto-u.ac.jp/~itohiro/Games/100301/100301-11.pdf> , accessed 20013/01/12
- [8] K. Hamada, "A Picturesque Maze Generation Algorithm with Any Given Endpoints," Journal of Information Processing, Vol. 21, No. 3, pp.393-397, 2013
- [9] T. Kurokawa, "Picture Maze Generation by Successive Insertion of Path Segment," British Journal of Mathematics and Computer Science, Vol.4, Issue.24, pp.3444-3463, 2014.12, Science Domain international, ISSN:2231-0851, <http://www.sciencedomain.org/issue.php?iid=699&id=6>
- [10] K. Ikeda and J. Hashimoto, "Stochastic Optimization for Picture Maze Generation," Jouhou Shori Gakkai Ronbunshi, Vol. 53, No.6, pp.1625-2634, 2012, in Japanese
- [11] T. Kurokawa, K. Mori, and T. Mizuno, "Automatic construction of picture maze by repeated contour connection,"Thailand-Japan Joint Conference on Computational Geometry and Graphs, pp.5-6, 2012
- [12] T. Kurokawa, "Picture maze generation and its relation to graphs," 2013 SHIKOKU-SECTION JOINT CONVENTION RECORD OF THE INSTITUTE OF ELECTRICAL AND RELATED ENGINEERS, p.231, 2013
- [13] T. Kurokawa, "Picture Maze Generation by Repeated Contour Connection and Graph Structure of Maze," Computer Science and Engineering, Vol.3, No.3, pp.76-83, 2013.12, Scientific & Academic Publishing, USA , ISSN:2163-1484 <http://article.sapub.org/10.5923.j.computer.20130303.04.html> Accessed 2014/1/20
- [14] F. J. Wong and S. Takahashi, "Flow-based Automatic Generation of Hybrid Picture Mazes," Computer Graphics Forum, 28, (7), pp.1975-1984, Wiley-Blackwell, 2009
- [15] S. Ishida, "Algorithm of automatic maze generation," <http://www5d.biglobe.ne.jp/%257estssk/maze/make.html>, (Japanese), Accessed 2013/10/8
- [16] "How to Build a Maze," <http://www.mazeworks.com/mazegen/mazetut/index.htm>, Accessed 2013/1/26
- [17] <http://www.microsoft.com/ja-jp/dev/express/default.aspx>.
- [18] <http://jaist.dl.sourceforge.net/project/opencvlibrary/opencv-win/2.3/> Accessed Oct., 2011
- [19] J. Matousek and J. Nešetřil, Invitation to Discrete Mathematics, Second Edition, Oxford University Press, New York, 2009.
- [20] Paint in Microsoft windows 7.