

Generation of Checkered Patterns and Their Variations by Making Use of Eulerian Graph Features

Tomio Kurokawa^{1*}

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

ABSTRACT

A variety of pictures of checkered pattern graphics appear on the Internet. However, rarely seen is how those graphics were produced. This paper shows various checkered patterns and their variations generated automatically by a computer program, making use of features of Eulerian Graph. Given a line drawing of binary images, the procedure extracts contours of the line figures. The contours in turn form a set of contour cycles of the regions surrounded by the original lines of the figures, which are supposed to form an Eulerian Circuit. The drawing can be various combinations of closed lines -- circles, rectangles, ellipses, any of closed drawings. Bi-partitioning the contours into partitions (X, Y), filling inside the regions of X and Y, the procedure outputs the checkered patterns or their likes. Various patterns were successfully produced in a number of experiments as well as demonstrating that the algorithm is very robust and that some of the patterns are aesthetic. Since the checkered patterns are used in the Emblem of Tokyo Olympic 2020, it is timely to do the demonstration about the design of checkered patterns and their relations with computer programs and mathematics.

Keywords: Checkered patterns, Closed one stroke curve, Eulerian graph, Contour cycles, Region adjacency graph, Dual, Bipartite graph

1. INTRODUCTION

Checkered patterns (Ichimatsu Moyo in Japanese) are to appear, or it might be better saying to have appeared on the Emblem of Tokyo Olympic Games 2020 [1]. Checkered patterns are recognized as traditional arts and widely known in Japan as well as in the world.

Images of various checkered patterns [2] can be seen on the Internet. But we could rarely see articles written about how they are drawn, especially the methods to automatically draw the patterns by a computer program. This paper is intended to demonstrate various checkered patterns and their variations automatically generated by a computer program using the method [3], which was previously published but without demonstrations of generating checkered patterns.

The generation method of checkered patterns in this study first concerned with contour extraction of binary images, especially binary line drawing. This contour extraction is realized by using the OpenCV library [4]. The specific function of contour extraction of the library is based on the study by Suzuki and Abe [5]. In the study, they provide a method to obtain the topological structure and the contours of the binary image at the same time. In the software [4], the topological structure is realized in a tree and the contours are by sequences of coordinates. They call the contour extraction as "border following". It provides two types of borders -- "outer border" and "hole border." These two are important for the present study. Researches concerning "border following" also appear in [6-8].

* Corresponding author: kurokawa@aitech.ac.jp

Suzuki and Abe reported that their study [5] was extended from [6]. This contour extraction using the library [4] has been successfully used in [9-11].

Another matter about this study after extraction of contours is bi-partitioning of the contours. The construction of bipartite graph usually appears in a textbook, such as [12]. But in the present study, the extracted contours are bi-partitioned into two sets, just separating the contours into two sets. The final step is the actual region painting, to which the present study does not pay too much attention.

2. OUTLINE OF THE STUDY AND CONTOUR EXTRACTION OF LINE FIGURES

2.1 OUTLINE OF OVERALL PROCEDURE

The procedure of this study first extract contours from an original binary line figure which is supposed to constitute Eulerian Circuit [12]. Intersections of the lines of the figure are considered as vertices and the lines between the intersections are edges. The extracted contours are the cycles which surround each of the regions surrounded by the originally drawn curves. If two contours surrounded regions are adjacent, then the corresponding two contours are adjacent, that means the two contours are very near or very close. Based on this nearness between contours, the contours can be bi-partitioned. Bi-partitioning algorithm [3] just separates them into X-partition and Y-partition. The regions of one partition are painted with a color and those of another partition are with another color -- 2-coloring, which produces the final checkered patterns or their variations.

2.2 CONTOUR EXTRACTION OF LINE FIGURES

The contour extracting procedure of this study, which appears in [10], works as follows:

1. Prepare a line figure binary image.
2. Execute the contour extraction function of the library [4] for the image with obtaining a tree, which holds the result.
3. Visit each node in the tree and obtain a set of series of coordinates of the contours.

The extracted contours are organized as a tree [5,10] represent the topological features of the contours and series of two dimensional coordinates for each contour. The root node of the tree represents the outermost contour of the line figure, the so called "outer border" [5]. All the nodes other than the root are just one level under the root, representing the inside contours, or "hole borders" [5] for the case of the figures of the present study. All points of contours, regardless outer or inside, are on the lines of the original figures. The outer contour points are represented counter-clockwise and the inside contour clockwise.

The obtained contours are the set of contour cycles which surround each of the regions surrounded by the originally drawn figure curves. The processing of this set of contours is explained in the next section.

3. RE-PHRASING AND MODIFYING ALGORITHM OF PREVIOUS STUDIES

The previous study [3] provides a method which generates a maze, where the solution path is the given one-stroke curve itself. In the study, an algorithm was introduced to make the bi-partitioning of the contours extracted from the one-stroke curve. The curve need not be literally one-stroke. Any curve having the crosses (or vertices) with an even number of lines (edges) can be handled. This includes a diagram made from a number of closed curves. That is, the line figure such as Fig. 1, grids, or a combination of a number of rectangles or closed curves

* Corresponding author: kurokawa@aitech.ac.jp

are accepted. For such a figure, all crosses (vertices) have even number of edges, where extracted contours can be partitioned into sets [3].

Fig. 1 is an original drawing of two rectangles, the line width of which is 1, from which checkered patterns were produced. These rectangles were drawn by using OpenCV library with line thickness of 1. The program with VC++ and OpenCV for this study first extracts the contours of the lines from the figure. They are shown in Fig. 2. The contours are on the lines of the original figure. However, they are also the contours of regions surrounded by the drawn curves (lines). Actually, they are closed, hence cycles. They are numbered from 1 to 6 by the OpenCV program library as shown in Fig. 2. No.1 is for the contour of the outer region, or the outer region itself. They are double lines everywhere on the curves. However, those curves of Fig. 2 do not look double. It is because the contours overlap everywhere on the curves. Hence, Fig. 2 looks exactly the same as the original curves, Fig. 1. This means that all the contour lines are on the original image lines.

Since a figure made of closed lines (foreground white line) constitutes an outer region, a connected figure lines and a number of hole regions, the extracted contours of the figure lines should be made of an "outer" contour and a number of "hole" contours, both on the original figure lines. Since the figure is made of lines of one-pixel width, all contour line parts are on the original one-pixel figure lines. That's why contours look like consisting of single lines, but actually completely overlapped. Accordingly, every line part is made of two borders — either an "outer border" and a "hole border", or two "hole borders". Despite extracting contours of line figure are considered a kind of unusual processing, results were successful and reasonable. The contour extraction function of OpenCV [4] and its theoretical base [5] should be highly evaluated.

Since the dual [13] of an Eulerian planar graph is a bipartite graph, the region adjacency graph (RAG), which can be obtained by complete checking of contour adjacency, shown as yellow in Fig.3, where the regions are considered as vertices and the edges exist between the adjacent regions, constitutes the bipartite graph. This process of obtaining RAG is not in the bi-partitioning algorithm [3] in subsection 3.1.

The algorithm introduced in [3] can be applied to make the partitions (X, Y) and is rephrased with more clarity:

3.1 BI-PARTITIONING ALGORITHM

INPUT:

A set, *CONTOURS*, of all the extracted n contour cycles as in Fig. 2 and is expressed as

$$CONTOURS = \{ cycle_j \mid 1 \leq j \leq n \}. \quad (1)$$

With n_j the length of $cycle_j$, each $cycle_j$ is expressed as

$$cycle_j = \{ (x(j, i), y(j, i)) \mid 1 \leq i \leq n_j \}. \quad (2)$$

OUTPUT:

Two partitions called *X-partition* and *Y-partition*, obtained from the bipartite graph, are disjoint and the union of the two is *CONTOURS* of (1).

DEFINITIONS:

The distance from $cycle_j$ to $cycle_k$ is defined as:

* Corresponding author: kurokawa@aitech.ac.jp

$$d(j, k) = \min_{1 \leq p \leq n_j \text{ \& } 1 \leq q \leq n_k} \sqrt{(x(j, p) - x(k, q))^2 + (y(j, p) - y(k, q))^2}. \quad (3)$$

146

147

METHOD:

148

Assumption: n : initial number of contour cycles in *CONTOURS*

149

X, Y : partition sets of the bipartite graph, both sets are empty at the beginning; they will have the resulted vertices of bi-partition.

150

151

1. Remove an arbitrary $cycle_j$ from *CONTOURS* and put it in X ;

152

2. Repeat begin

153

3. For each j in X do begin

154

4. For each i in *CONTOURS* do begin

155

5. If j and i are adjacent, do begin

156

6. Remove i from *CONTOURS* and put it in Y ;

157

7. If no cycle remains in *CONTOURS*, do begin

158

8. If the outer cycle is in Y , exchange X and Y ;

159

8. Paint inside all *CONTOURS* with different colors for X and Y and finish;

160

10. end;

161

11. end;

162

12. end;

163

13. end;

164

14. Exchange X and Y ;

165

15. end;

166

167

CONTOUR ADJACENCY:

168

The adjacency checking is based on distance computation between two cycles. The basic criteria for the adjacency is based on the overall average of the shortest distance (d_{AS}) between two cycles $cycle_j$ and $cycle_i$, where d_{AS} is defined as:

171

172

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

173

174

where $d(j, l, i, k)$ is defined as

175

176

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

177

178

where $j \neq i$ and n_j is the number of points on $cycle_j$.

179

Let's define two cycles $cycle_j$ and $cycle_i$ are adjacent if there are more than a certain number, β , of points on $cycle_j$ such that the shortest distance, $d_s(j, p_l, i)$, from a point p_l on $cycle_j$ to $cycle_i$ is less than d_{AS} multiplied by a certain number, α , where $d_s(j, p_l, i)$ is expressed as:

182

183

$$d_s(j, p_l, i) = \min_{1 \leq k \leq n_i} d(j, l, i, k). \quad (6)$$

184

185

186

So if more than β such different points on $cycle_j$ exist, $cycle_j$ and $cycle_i$ are considered adjacent. That is, $cycle_j$ and $cycle_i$ are adjacent if the size of S is greater than β in the following expression:

188

189

190

$$S = \{p_l \mid d_s(j, p_l, i) \leq \alpha * d_{AS} \text{ and } 1 \leq l \leq n_j \text{ where } \alpha \text{ is a small positive value}\}. \quad (7)$$

191

192

193

3.2 EXPERIMENT WITH BI-PARTITIONING AND REGION FILLING

* Corresponding author: kurokawa@aitech.ac.jp

194 By applying the above algorithm to Fig. 2, the contour cycles are bi-partitioned to X-partition
 195 (Fig. 4) and Y-partition (Fig. 5). Although they are interchangeable, the partition with the most
 196 outer contour is designated as X-partition in this study. X-partition includes the contours of 1,
 197 and 4. While Y-partition holds 2, 3, 5, and 6. Fig. 6 is obtained by examining all the adjacencies
 198 between the contours. The graph (yellow) in Fig. 3 and the one in Fig. 6 represent the same
 199 graph. Filling the regions of X-partition with yellow, Fig. 7 is obtained, so called checkered
 200 patterns.

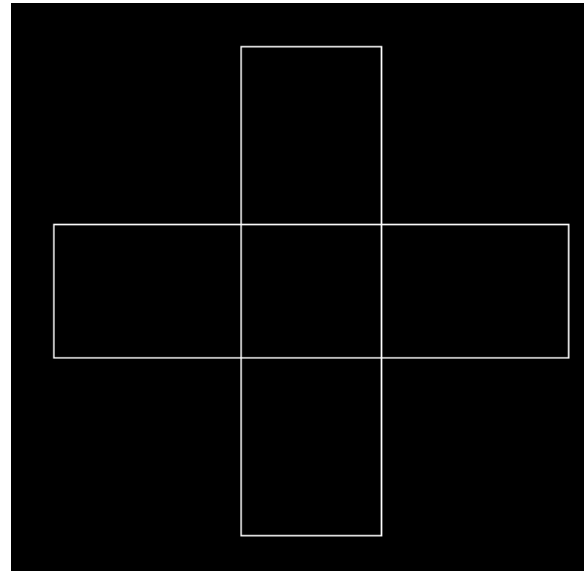


Fig. 1 Original drawing of two rectangles overlapped with line width of 1

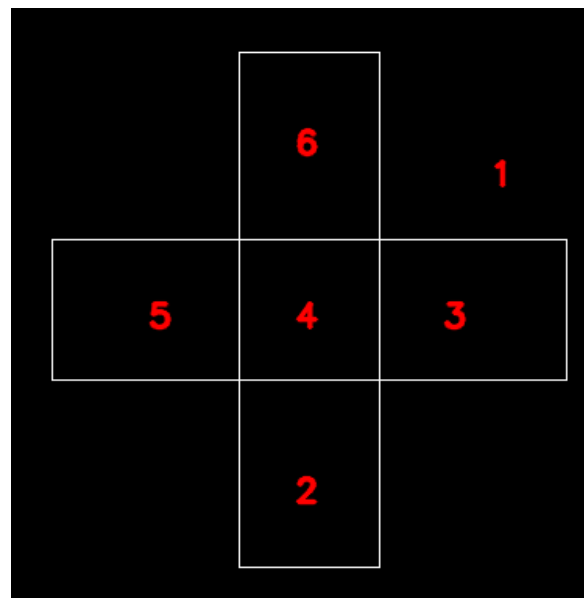


Fig. 2 Extracted contours of two rectangles overlapped, 6 contours

247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299

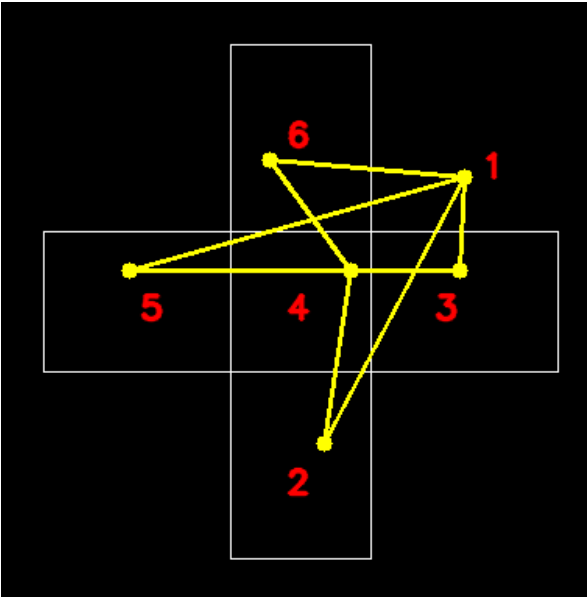


Fig. 3 Extracted contours constitute a bipartite graph and the regions are numbered

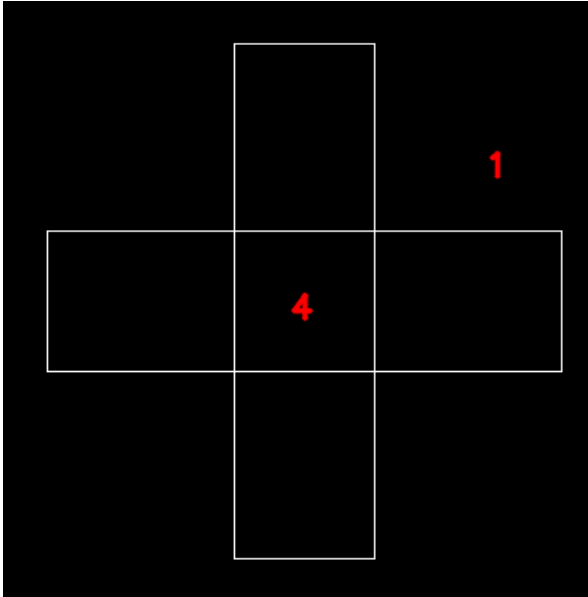


Fig. 4 X-partition, 2 contours

* Corresponding author: kurokawa@aitech.ac.jp

300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352

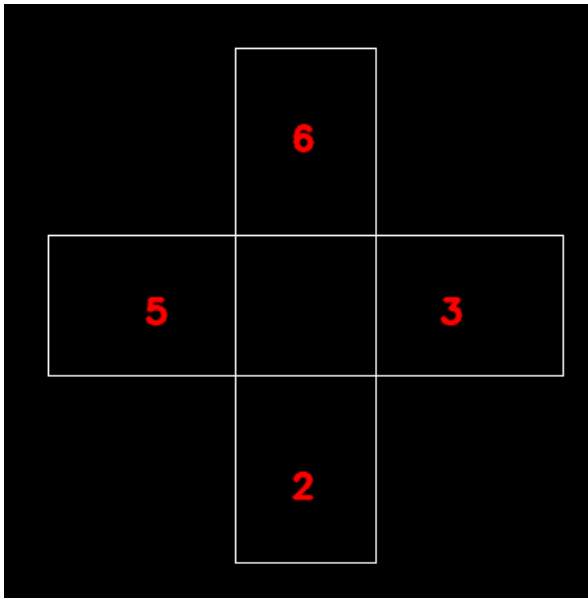


Fig. 5 Y-partition, 4 contours

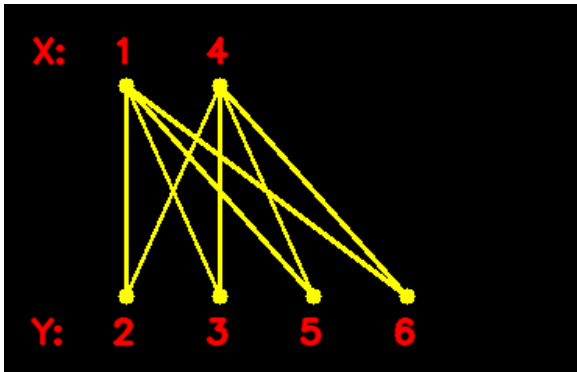


Fig. 6 bi-partitioned graph

* Corresponding author: kurokawa@aitech.ac.jp

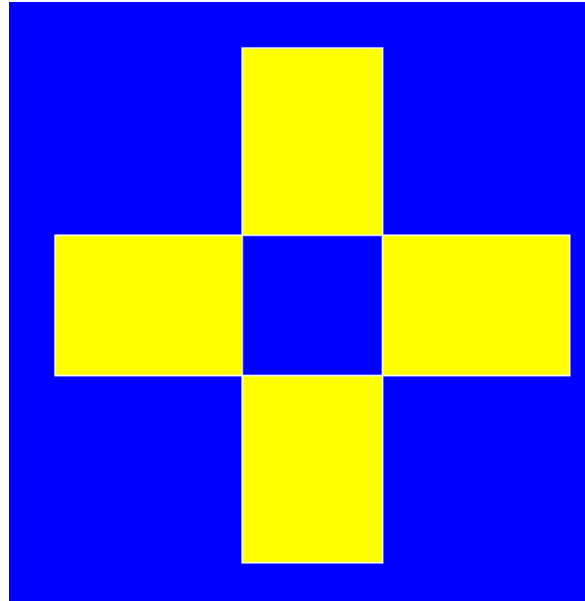


Fig. 7 Checkered patterns: X-partition is filled with blue, and Y-partition and boundaries with yellow

4. MORE EXPERIMENTS WITH COMPLEX FIGURES AND DISCUSSIONS

4.1 TWO RECTANGLES OVERLAPPED BUT WITH LINE WIDTH 5

In these experiments, the algorithm was applied to the diagram of Fig. 8, which was drawn using OpenCV library with line thickness of 5, the same shape figure as Fig. 1, but with different line width. Contours were extracted as in Fig. 9. All the contours clearly appeared and no overlap occurred this time. By comparing Figs. 8 and 9, we notice that contours are for the drawn lines as well as for the regions surrounded by the drawn lines. And the figure clearly shows how each of the contours is related to its corresponding region. The contours were bi-partitioned into X-partition (Fig. 10) and Y-partition (Fig. 11). These figures clearly illustrate which contour belongs to which partition. Although the two partitions are interchangeable, the partition with the most outer contour is designated as X-partition in this study. Hence, X-partition includes the contours of 1, and 4, and Y-partition holds 2, 3, 5, and 6. Fig.12 (RAG) and Fig. 13 were obtained by checking all the adjacencies between the contours. The graph (yellow) in Fig. 12 is exactly the same as the bipartite graph (yellow) of Fig. 3. And the graph of Fig. 13 is the same as that of Fig. 6. This means the extracted contours constitute the same bipartite graphs as well as the same RAGs. This suggests that the width difference of the original line drawings causes no significant difference for the RAG. Filling the regions of X-partition with blue color and Y-partition with red color and the boundaries with white color, Fig. 14 was obtained, the checkered patterns with wider boundaries. The patterns give different

* Corresponding author: kurokawa@aitech.ac.jp

impressions with wider boundaries. The successful experiments in section 3 and section 4.1 suggest that the algorithm is effective even for different line widths.

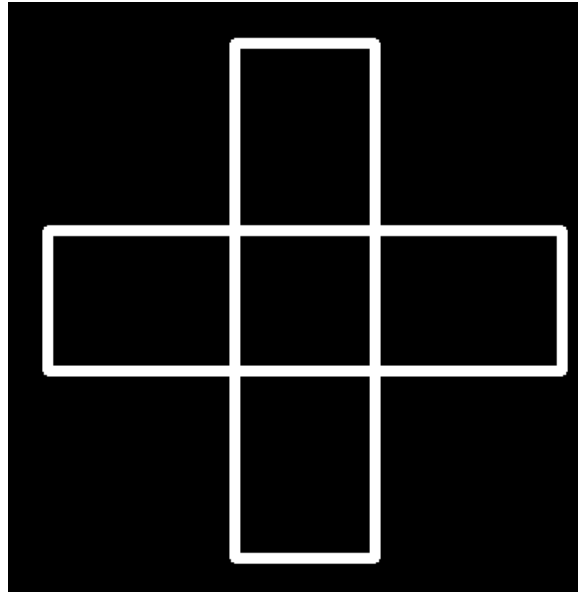


Fig. 8 Original drawing of two rectangles overlapped with line width of 5

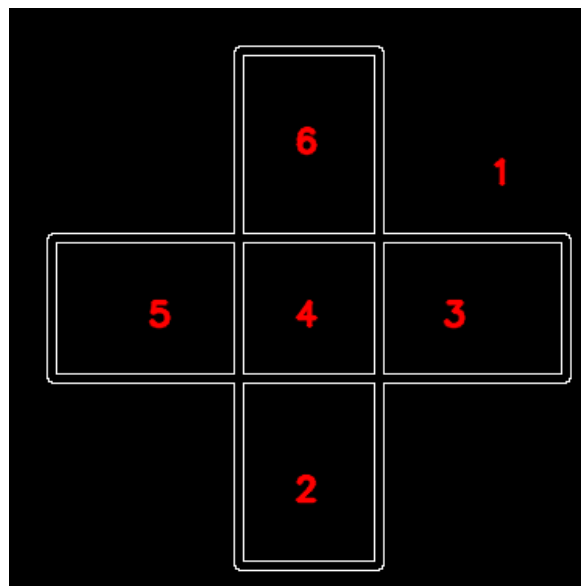


Fig. 9 Extracted contours, 6 contours and regions are numbered

455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498

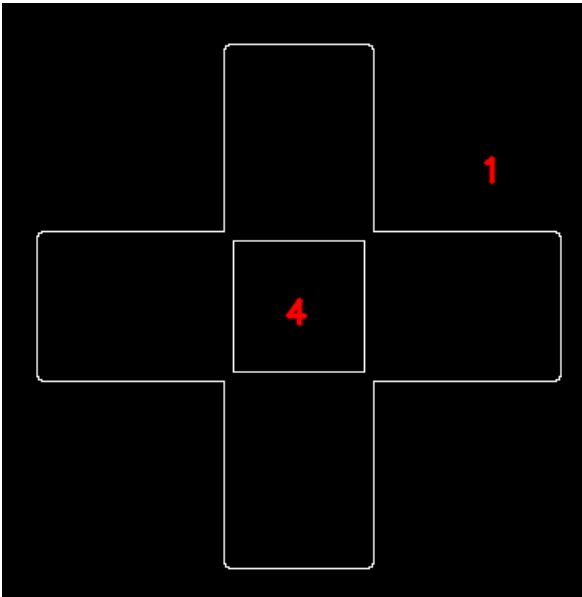


Fig. 10 X-partition, 2 contours

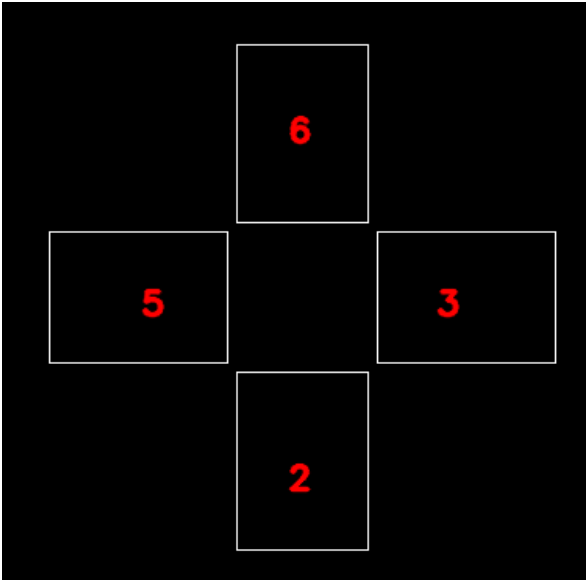


Fig. 11 Y-partition, 4 contours

* Corresponding author: kurokawa@aitech.ac.jp

499
500
501
502
503
504
505
506
507
508
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
541
542

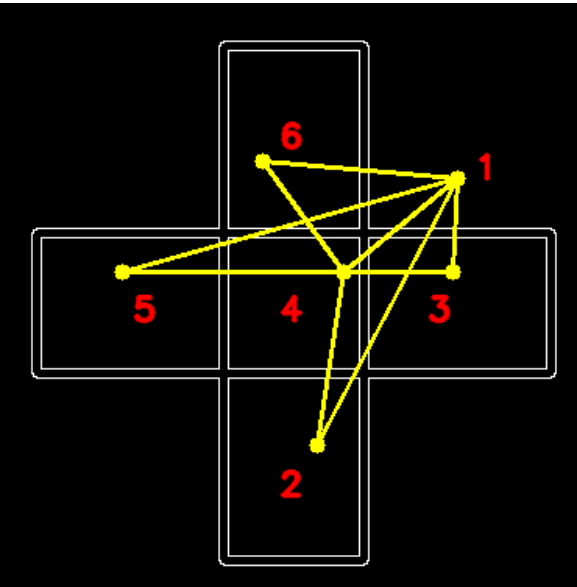


Fig. 12 Extracted contours constitute a region adjacency graph (RAG) as well as a bipartite graph

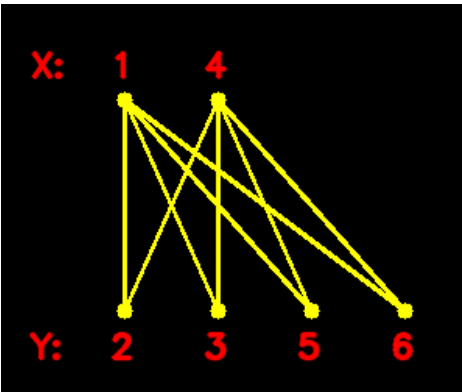


Fig. 13 Bi-partitioned graph

* Corresponding author: kurokawa@aitech.ac.jp

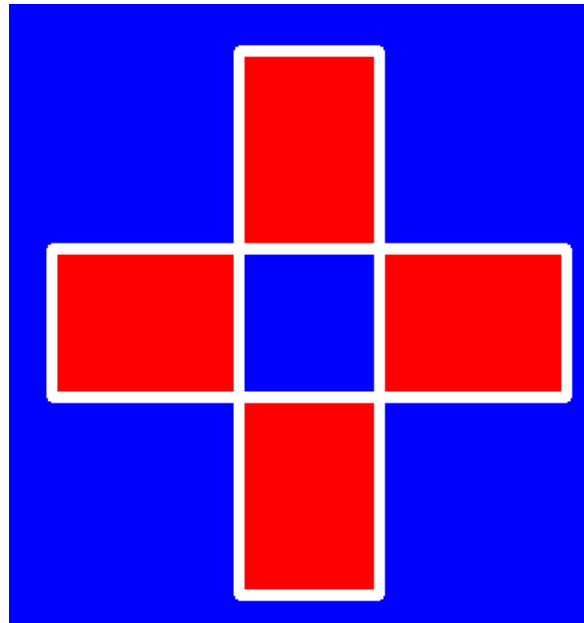


Fig. 14 Checkered patterns: X-partition blue, Y-partition red, and boundaries white

4.2 A LARGE NUMBER OF GRIDS WITH LINE WIDTH 1

Fig. 15 is the figure of 10 rectangles crossed drawn using OpenCV library with line thickness of 1. Contours were extracted as in Fig. 16. There are 102 contours. Figs. 15 and 16 look exactly the same. The lines in Fig. 16 look single everywhere on the lines. Hence, it means that the contours should be overlapped everywhere.

The bi-partitioning algorithm was applied to Fig. 16. Fig. 17 is the resulted Y-partition with 60 contours; X-partition is not shown because it's supposed to look the same as Y-partition. Fig. 18 is the checkered patterns obtained by filling the inside of all the contours of X-partition blue, Y-partition red, and the boundary white. Fig. 19 is with X-partition and boundary blue and Y-partition red, literally 2-colored. Some subtle difference seems to exist between Figs. 18 and 19.

591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618

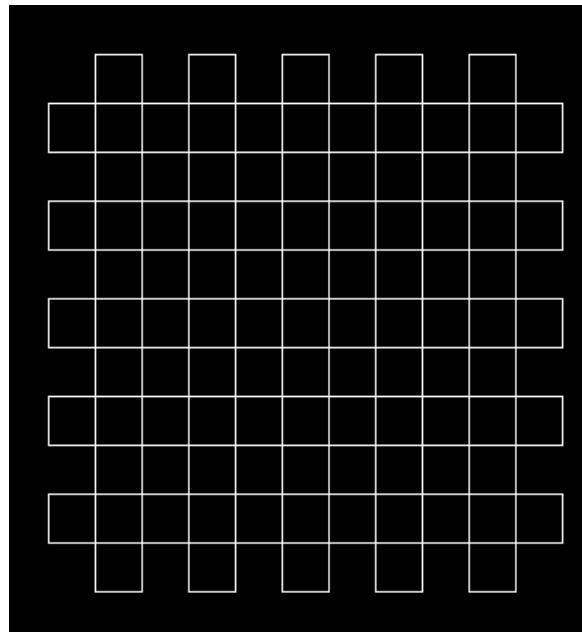


Fig. 15 Original drawing of grids: 10 overlapped rectangles with line width of 1

619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661

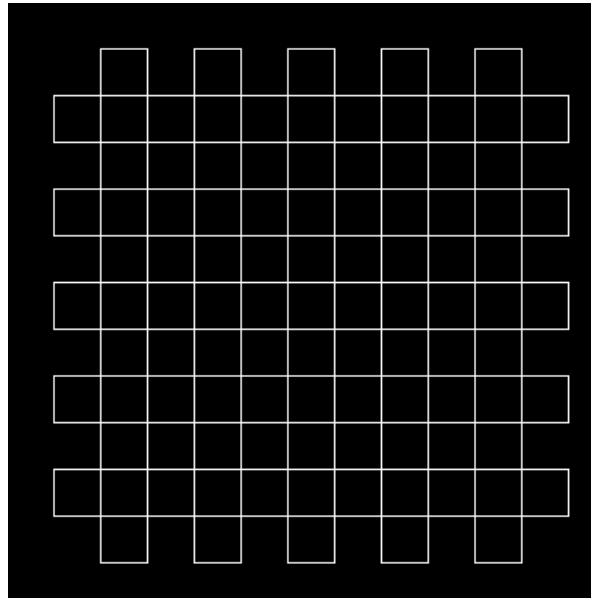


Fig. 16 Contours extracted, 102 contours

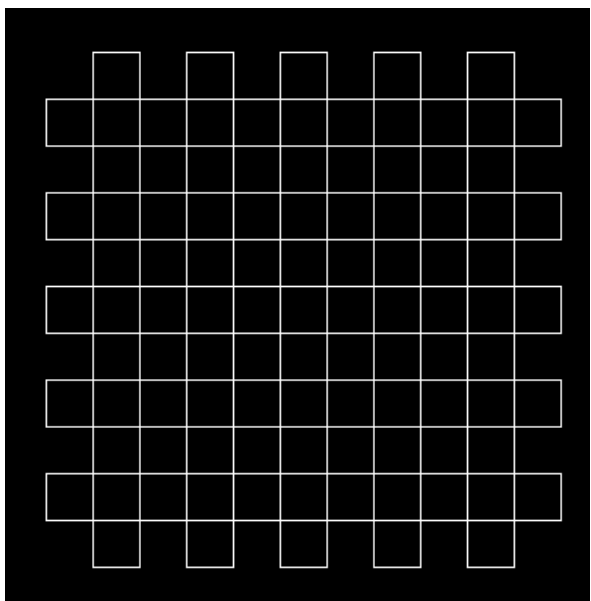


Fig. 17 Y-partition, 60 contours

662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705

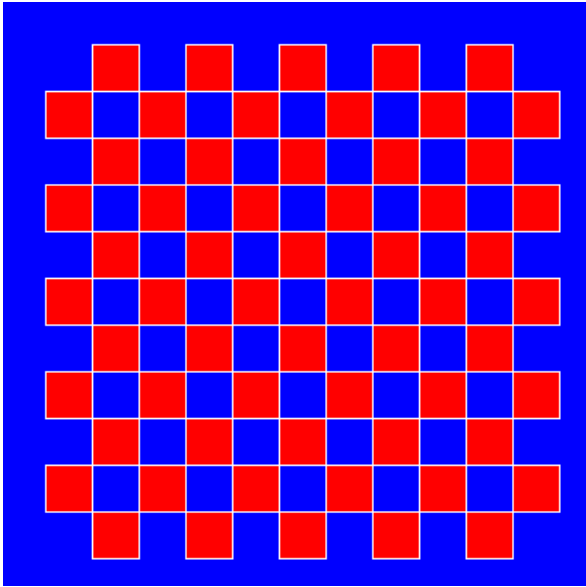


Fig. 18 Checkered patterns: X-partition with blue, Y-partition with red, and boundary white

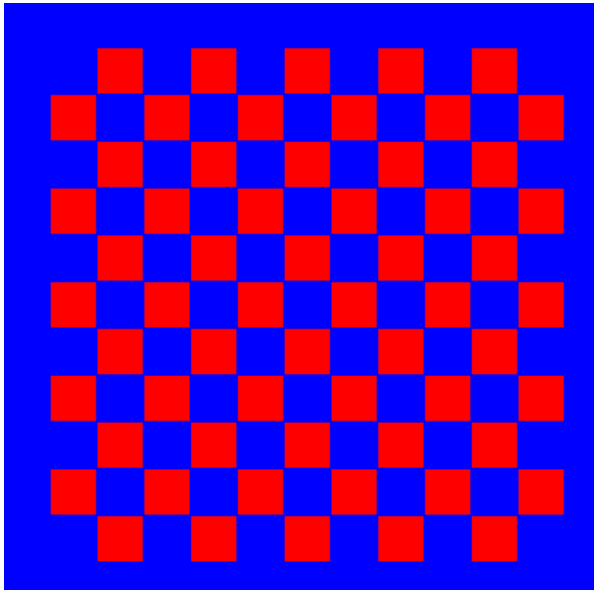
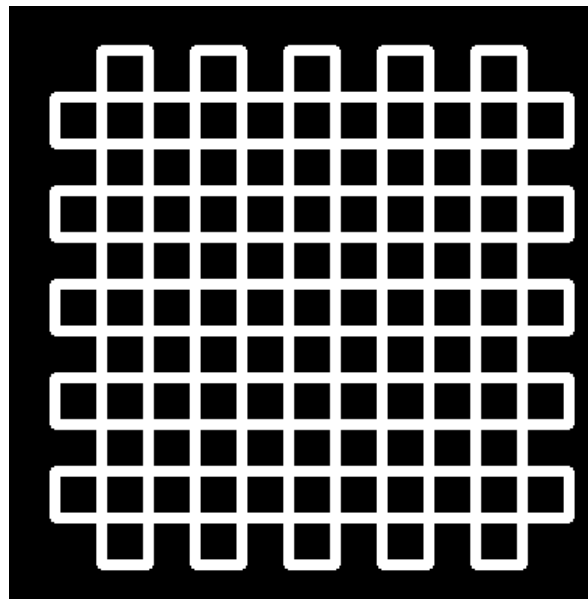


Fig. 19 Checkered patterns: X-partition, boundary blue, and Y-partition red

4.3 A LARGE NUMBER OF GRIDS WITH LINE WIDTH 5

* Corresponding author: kurokawa@aitech.ac.jp

706 Fig. 20 is the figure of 10 rectangles crossed drawn using OpenCV library, but with line
707 thickness of 5. Contours were extracted **and** the result is in Fig. 21. There are 102 contours
708 shown as in Fig. 16. It seems that no contours overlap. The contours in Fig. 21 are quite
709 different from those in Fig.16. The bi-partitioning algorithm was applied to Fig. 21. Fig. 22 is
710 X-partition with 42 contours, one of which is the outer contour, very long, and Fig. 23 is Y-
711 partition with 60 contours. Fig. 24 shows the checkered patterns with X-partition filled with blue
712 color and Y-partition with red and the boundaries with white. It seems that the algorithm worked
713 quite well, even with the different line width, this time too. The original picture of Fig. 21 is
714 different from that Fig. 15 just for the line width. But the resulted patterns give different
715 impressions.



746 Fig. 20 Original drawing of grids: 10
747 overlapped rectangles with line width of 5
748

749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792

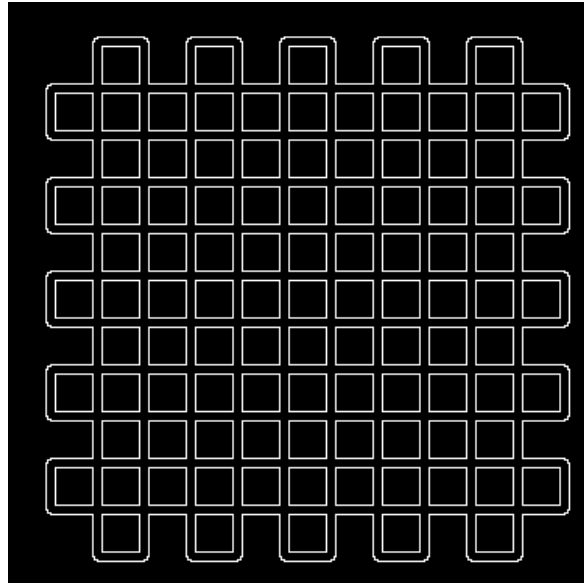


Fig. 21 Contours extracted from Fig. 20, 102 contours

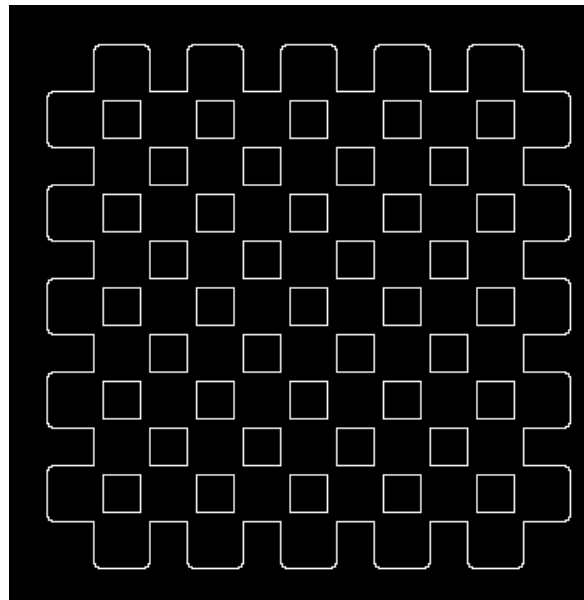


Fig. 22 X-partition, 42 contours

793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837

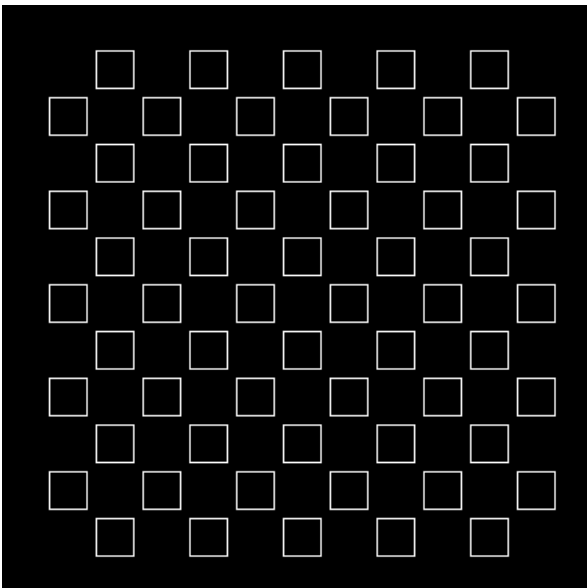


Fig. 23 Y-partition, 60 contours

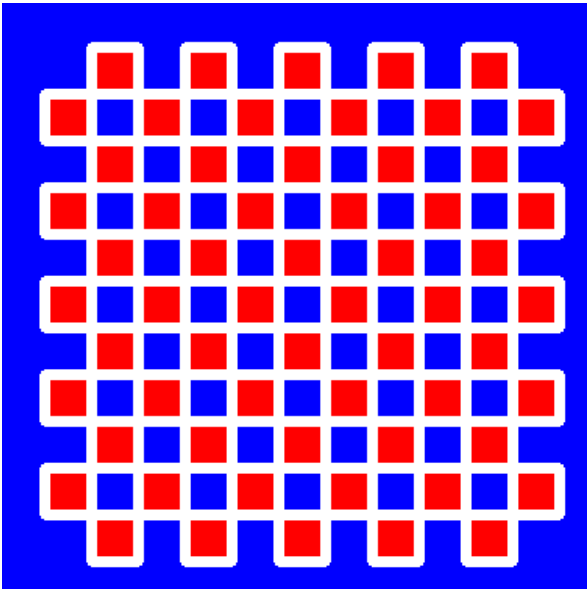


Fig. 24 Checkered patterns: X-partition with blue, Y-partition with red and boundary white

* Corresponding author: kurokawa@aitech.ac.jp

4.4 A LARGE NUMBER OF CIRCLES OVERLAPPED WITH LINE WIDTH 1

Fig. 25 is the figure of 30 circles crossed. These circles were drawn using OpenCV library with line thickness of 1. Since the drawing of circle curves is different from that of straight lines like rectangles, contours could be different from those of rectangle figure. Fig. 26 shows the contours extracted. But the lines look somewhat bolder. There are 140 contours counted by the program. The bi-partitioning algorithm was applied to Fig. 26. Fig. 27 is the resulted X-partition and Fig. 28 is Y-partition, both holding 70 contours. Figs. 26, 27 and 28 look alike. Lines of Fig. 26 look a little bolder. But those of Figs. 27 and 28 look thin. This suggests that the contour lines in Fig. 26 overlap partially, but those in Figs. 27 and 28 are not overlapped and just look single. As a matter of facts, the regions surrounded by the lines of Fig. 27 and those of Fig. 28 are completely different, exclusive each other because they are bi-partitioned regions. Fig. 29 is the output patterns (checkered-like patterns) with X-partition and boundaries painted with blue and Y-partition by red.

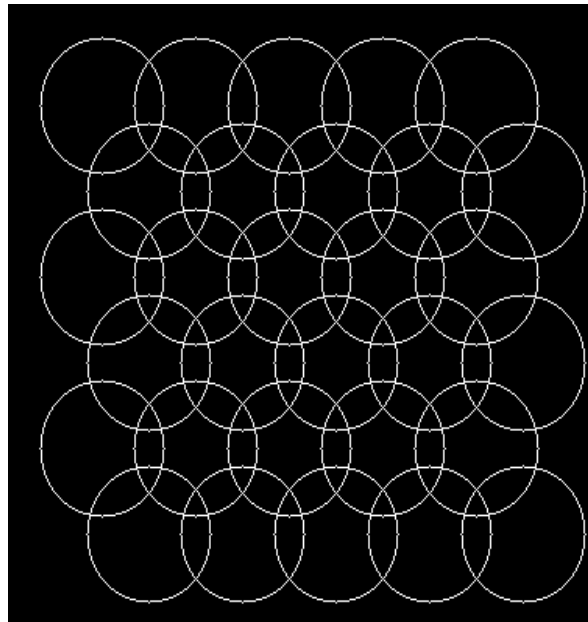


Fig. 25 Original drawing of 30 circles with line width of 1

885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928

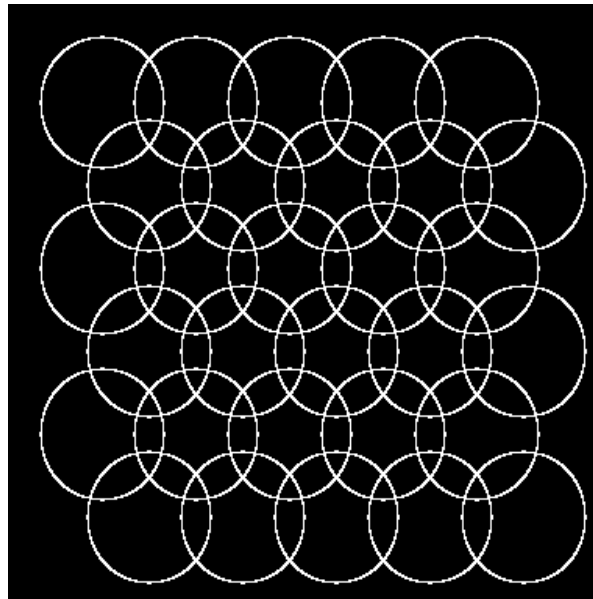


Fig. 26 Contours extracted, 140 contours

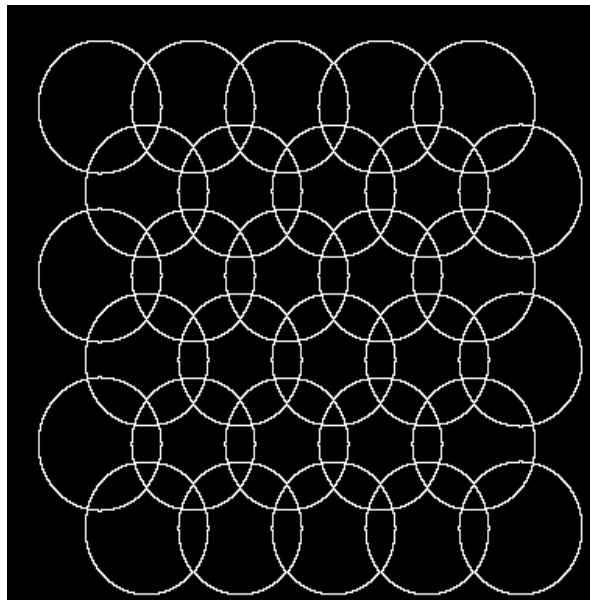


Fig. 27 X-partition, 70 contours

929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972

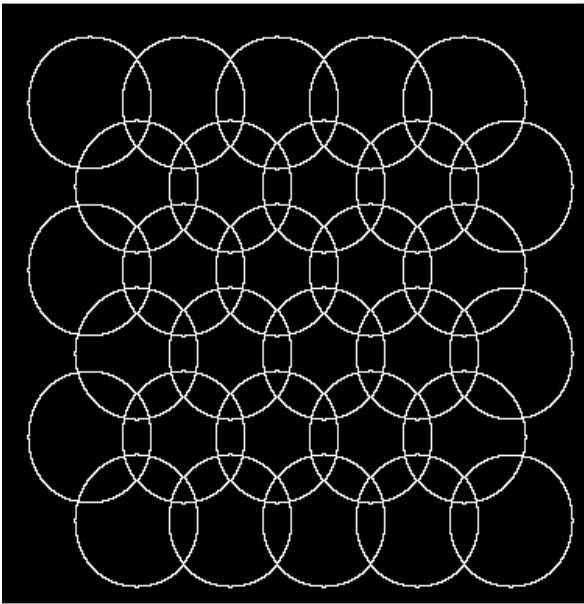


Fig. 28 Y-partition, 70 contours

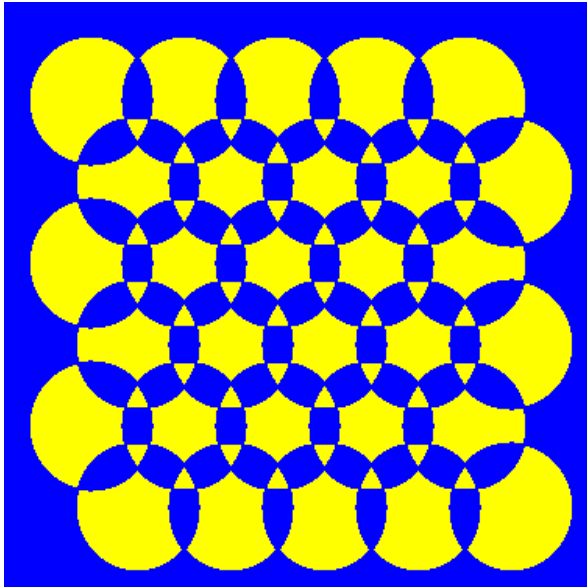


Fig. 29 Checkered-like patterns: with X-partition and boundaries blue, Y-partition yellow

* Corresponding author: kurokawa@aitech.ac.jp

4.5 A LARGE NUMBER OF CIRCLES OVERLAPPED WITH LINE WIDTH 3

Fig. 30 is the figure of 30 circles crossed, but drawn using OpenCV library with line thickness of 3. Fig. 31 shows the contours extracted. It seems that all the contour lines are completely separate and look thin. There are 140 contours. The bi-partitioning algorithm was applied to Fig. 31. Fig. 32 is the resulted X-partition and Fig. 33 is Y-partition, both holding 70 contours. Figs. 32 and 33 look complexly different and all the contours are completely separate. This means that regions of the partitions are completely separate and exclusive, but they occupy all the regions. Fig. 34 is the output patterns (checkered-like patterns) with X-partition painted by blue color, but the most outer region with green, Y-partition by red, and boundaries white.

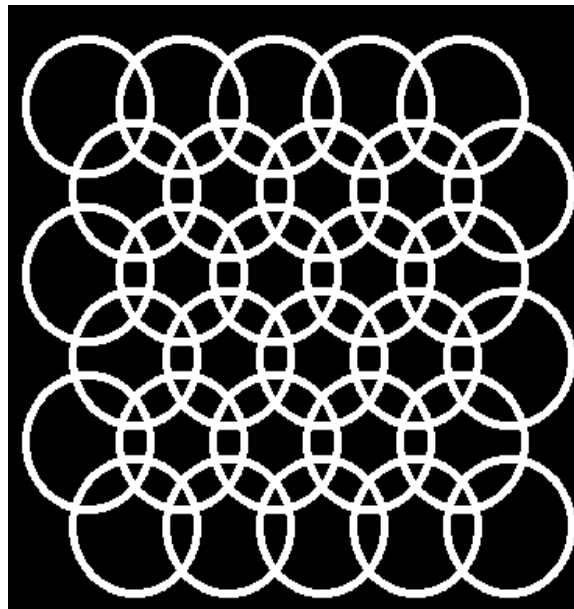


Fig. 30 Original drawing of 30 circles with line width of 3

1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075

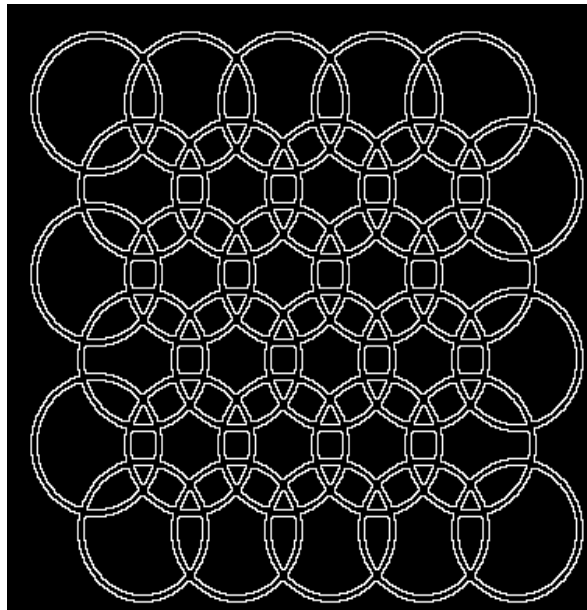


Fig. 31 Contours extracted, 140 contours

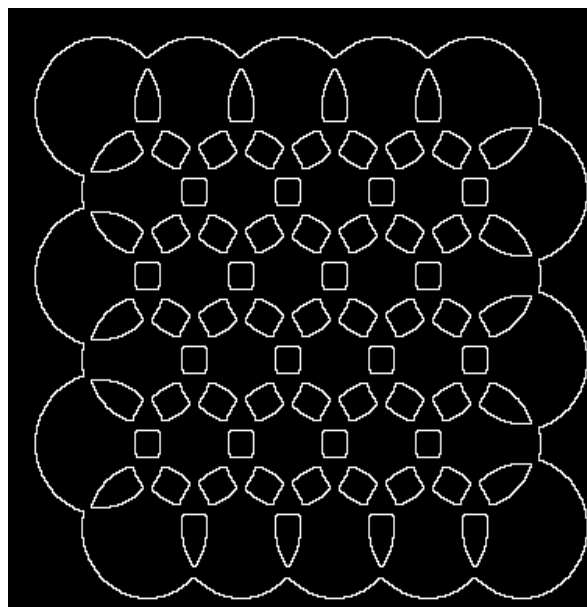


Fig. 32 X-partition, 70 contours

1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105

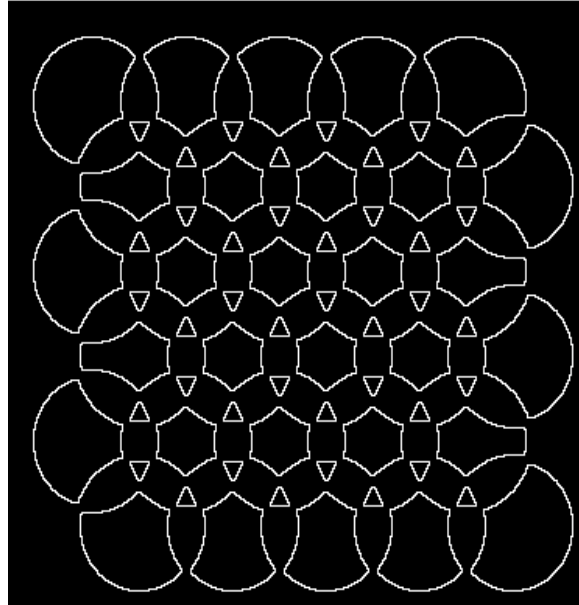


Fig. 33 Y-partition, 70 contours

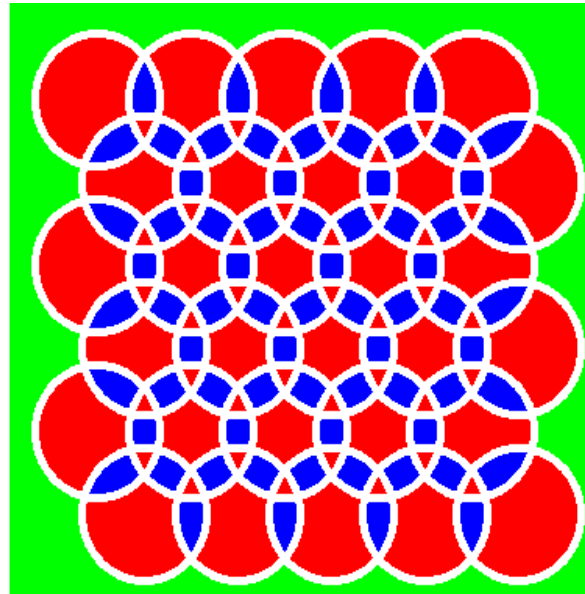


Fig. 34 Checkered-like patterns: with X-partition blue, most outer region green, Y-partition red and boundaries white

4.6 FREEHAND DRAWING OF 10 OVALS OVERLAPPED WITH LINE WIDTH 1

Fig. 35 is the diagram of 10 freehand drawn ovals with line width of 1 overlapped. Somehow line widths vary at places  originally drawn by MS WORD. Fig. 36 shows 102 contours extracted. Notice that some curve parts are double lines. The double lines seem 2 pixels apart. That is, the distance between the two are two pixels. Accordingly, the distance of the rests should be one pixel. As will be shown in the subsequent section, the average shortest distance (4) for Fig. 35 is 1.01. This suggests that only a very limited line parts are double. The bi-partitioning algorithm was applied to Fig. 36. Fig. 37 is the resulted X-partition with 42 contours and Fig. 38 is Y-partition with 60 contours. As seen, those contour curves are very thin, one-pixel width, thinner than those of Fig. 35. Fig. 39 is the checkered patterns obtained from Figs. 37 and 38  X-partition is blue, Y-partition and boundary are yellow.

Fig. 40 is the same diagram as Fig. 35, but with the outer rectangle frame line. The oval curves and the outer frame line cross for longer or wider area at the two touching places. Since these curves are not mathematically ideal curves, but realized in an actual image plane, occupying fussy crossing areas, it is expected to be more difficult to recognize the adjacency between contours. Actually, the parameter $\beta = 10$, adjacent point count, was used instead of $\beta = 4$ which were applied to most of the cases. This means that more severe criterion for adjacency was applied in this case. Fig. 41 shows 122 contours extracted. Those seem successfully extracted  no significant difference from the Fig. 36. The bi-partitioning algorithm was applied and the results are as in Fig. 42 (X-partition) and Fig. 43 (Y-partition). All contours were successfully partitioned into two parts. Since the intersecting parts are longer between freehand ovals and rectangle frame lines, the adjacent parts between the two regions which are supposed to be in a same partition set are inclined to be longer. This length, sometimes causes the confusion. But this was solved by setting β to 10, longer adjacent point count, of (7). The final checkered patterns are shown in Fig. 44. Notice that colors of blue and yellow

* Corresponding author: kurokawa@aitech.ac.jp

1159 are interchanged between Figs. 39 and 44. It is by the interchangeability of X-partition and Y-
1160 partition.
1161
1162

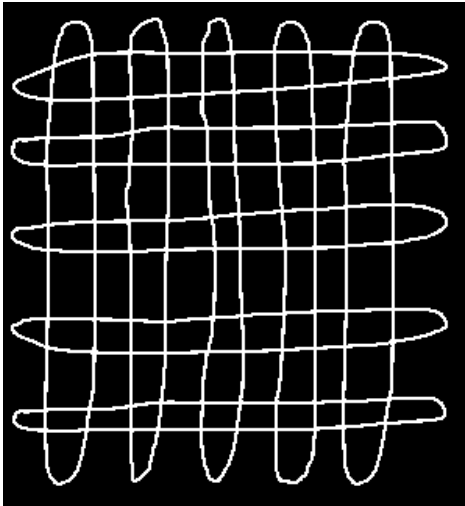


Fig. 35 Original drawing of 10
freehand ovals with line width of 1

1180

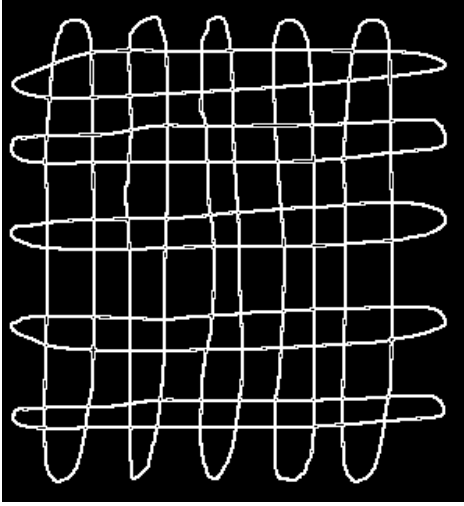


Fig. 36 102 contours extracted

1179

1181
1182

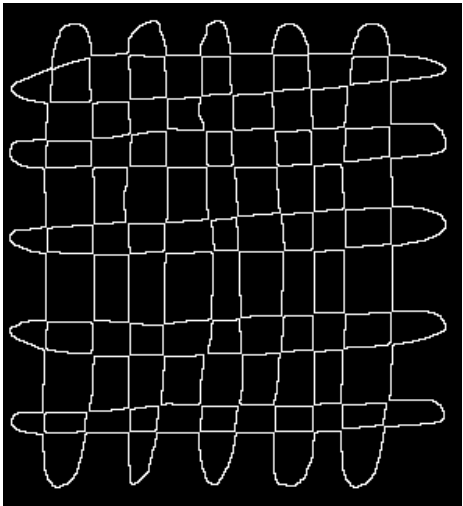


Fig. 37 X-partition, 42 contours

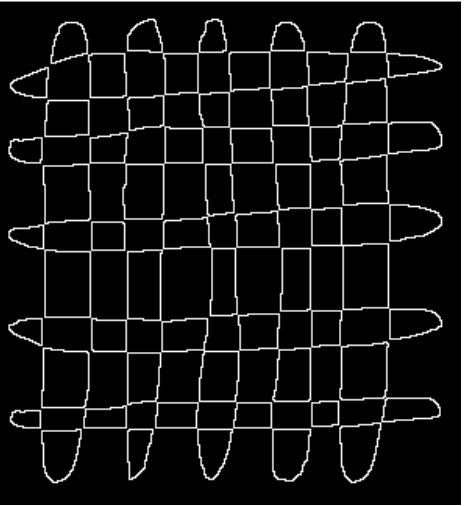


Fig. 38 Y-partition, 60 contours

1183

1200
1201
1202
1203

* Corresponding author: kurokawa@aitech.ac.jp

1204
1205

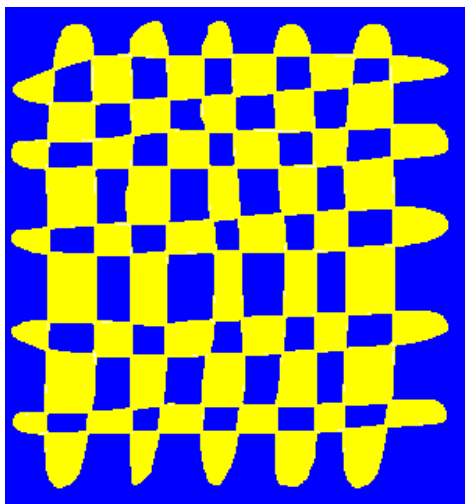


Fig. 39 Checkered-like patterns: X-partition blue, Y-partition and boundaries yellow

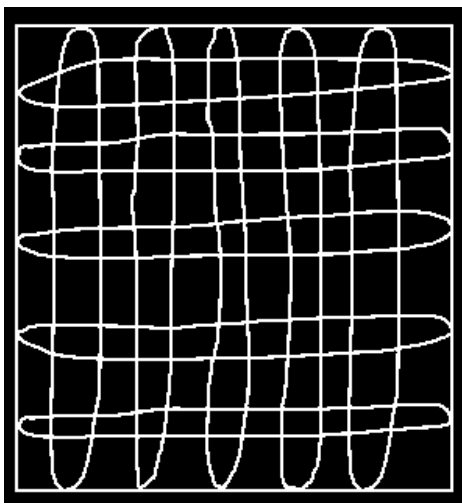


Fig. 40 Original drawing of 10 freehand ovals with line width of 1 and the outer frame

1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229

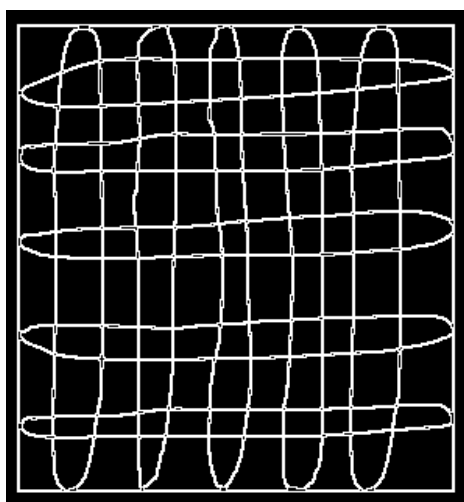


Fig. 41 122 contours extracted

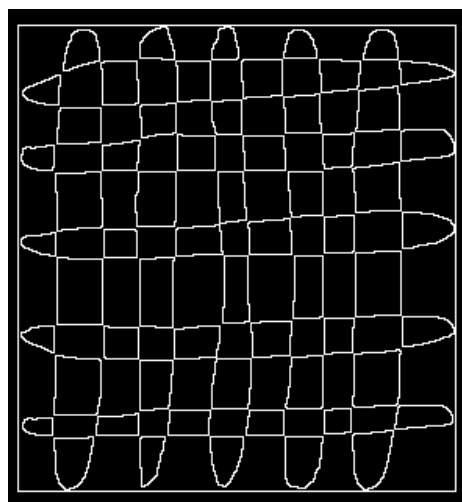


Fig. 42 X-partition, 61 contours

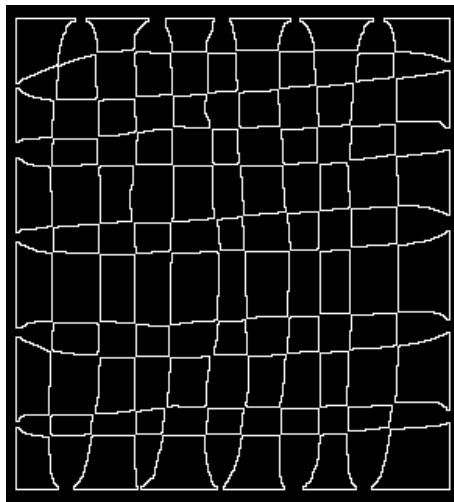


Fig. 43 Y-partition, 61 contours

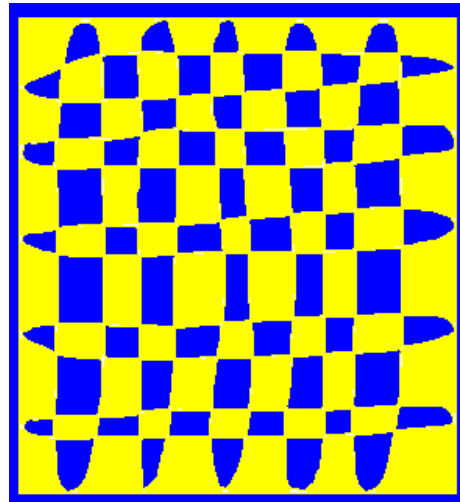


Fig. 44 Checkered-like patterns: X-partition blue, Y-partition and boundary yellow.

4.7 FREEHAND DRAWING OF 10 OVALS CROSSED WITH CURVE WIDTH 5

Fig. 45 is the figure of 10 freehand drawn ovals overlapped, but with curve width of 5. Fig. 46 shows 102 extracted contours. All of them are successfully extracted. Notice that the most outer contour is very long. All the contours are completely separate. The bi-partitioning algorithm was applied to Fig. 46. Fig. 47 is the resulted X-partition with 42 contours and Fig. 48 is Y-partition with 60 contours. Despite the wide and fussy crossing areas of curves, the bi-partitioning was successfully performed for all the regions using the parameters $\alpha = 1.1$ and $\beta = 4$. Fig. 49 is the final checkered-like patterns with X-partition blue and Y-partition red and boundaries white. These give another different impression from the ordinary checkered-like patterns of Fig. 39 or Fig. 44.

Fig. 50 is supposedly the same diagram as Fig. 45, but with the outer rectangle frame line of width 5. Notice that curves and lines cross at wider and fussy areas. Fig. 51 shows successfully extracted 122 contours. The bi-partitioning algorithm was applied. The results are as in Fig. 52 (X-partition with 61 contours) and Fig. 53 (Y-partition with 61 contours). The parameters $\alpha = 1.1$ and $\beta = 4$ were used as before. The final checkered-like patterns are shown in Fig. 54. Notice that colors of blue and red are seemingly interchanged between Figs. 49 and 54 despite regions of X-partition were painted with blue and those of Y-partition red. It is due to the interchangeable feature of X-partition and Y-partition.

1279
1280
1281

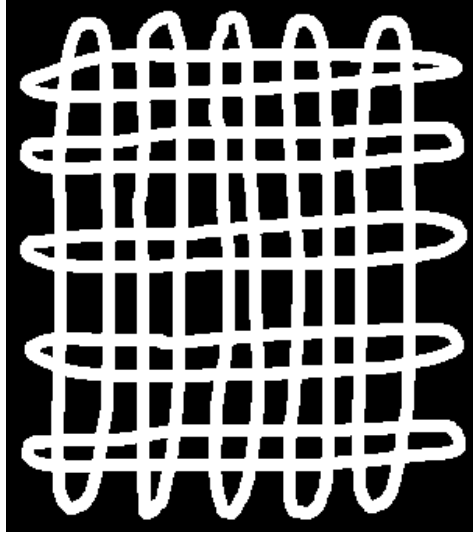


Fig. 45 Original drawing of 10
freehand ovals with line width of 5

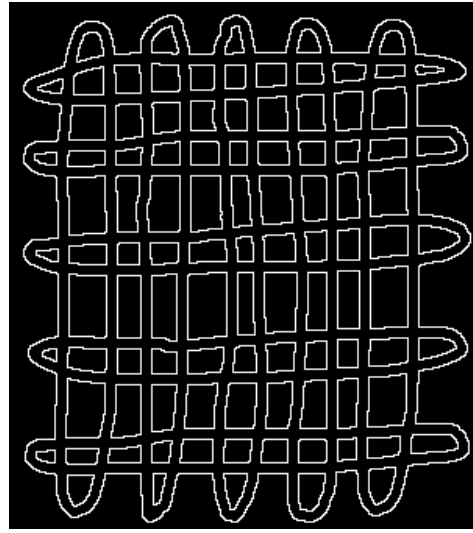


Fig. 46 102 contours extracted

1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310

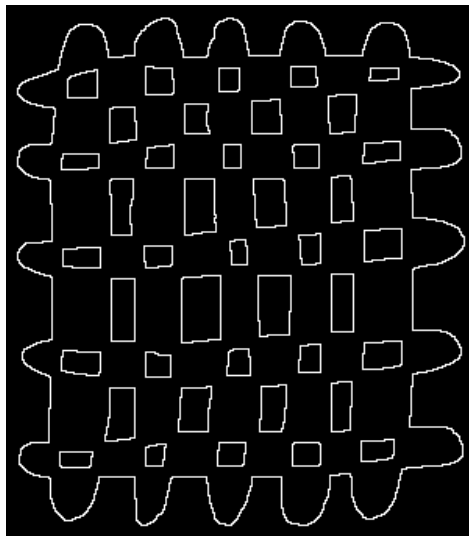


Fig. 47 X-partition, 42 contours

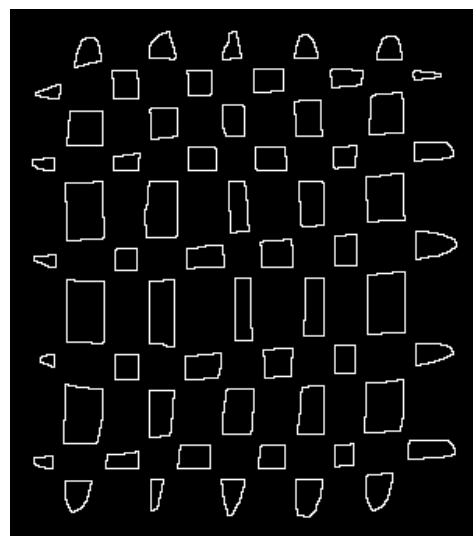


Fig. 48 Y-partition, 60 contours

1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334

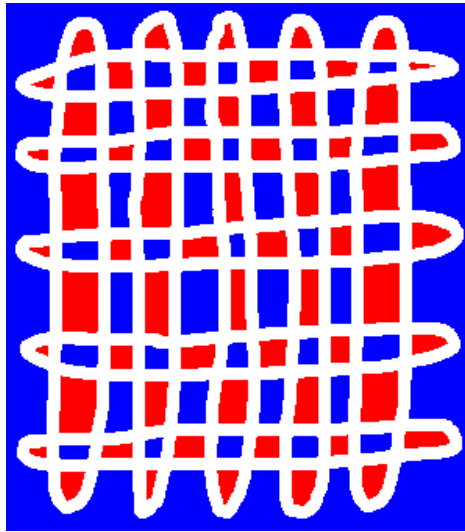


Fig. 49 Checkered-like patterns: X-partition blue, Y-partition red, boundary white

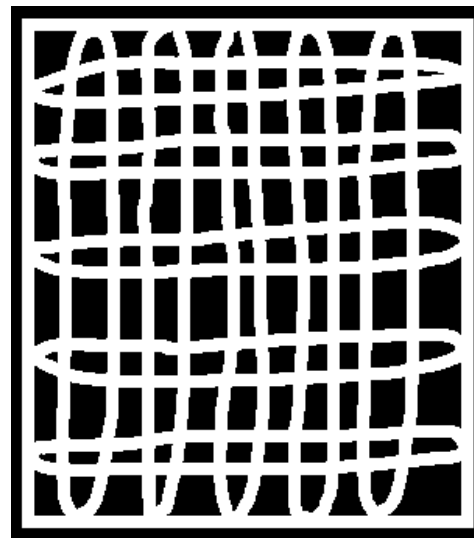


Fig. 50 Original drawing of 10 freehand ovals with line width of 5 and the outer rectangle frame

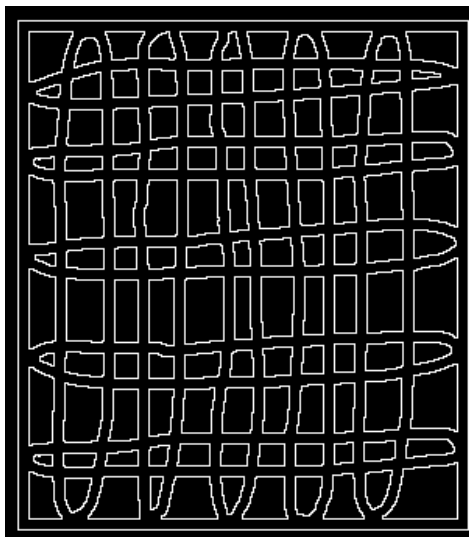


Fig. 51 122 contours extracted

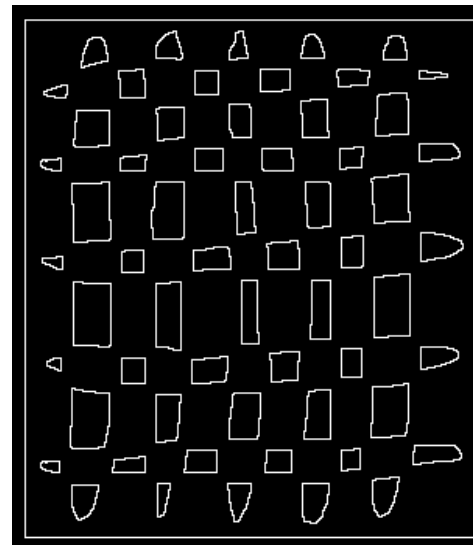


Fig. 52 X-partition, 61 contours

1335
1336
1337
1338
1339
1340
1341
1342

1343

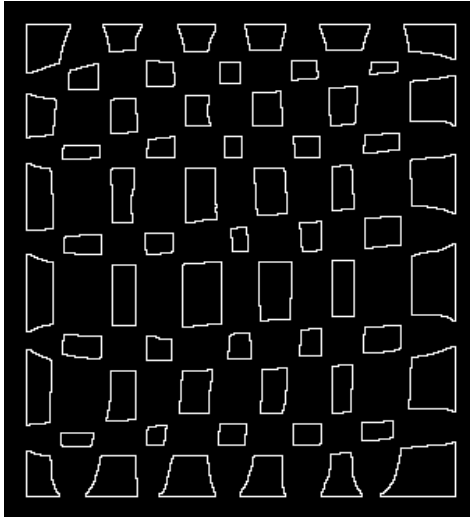


Fig. 53 Y-partition, 61 contours

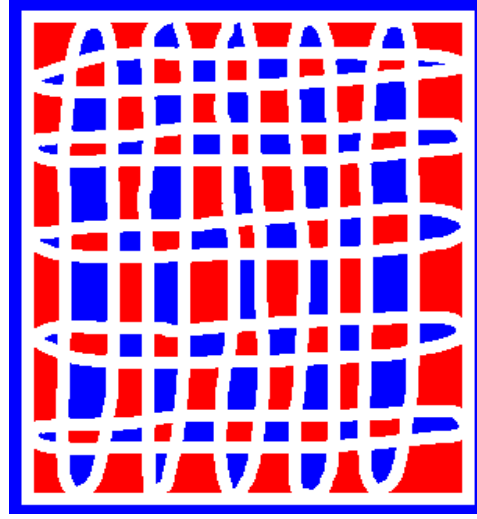


Fig. 54 Checkered-like patterns: X-partition blue, Y-partition red, boundary white

1344

1345

1346

1347

1348

1349

1350

1351

1352

1353

1354

1355

1356

1357

1358

1359

1360

1361

1362

1363

1364

1365

1366

1367

1368

1369

1370

1371

1372

1373

1374

4.8 TRIANGLES WITH SIX DEGREE VERTICES WITH LINE WIDTH 8

Fig. 55 is the figure of triangles with 6 degree vertices drawn using OpenCV library with line thickness of 8. Contours were extracted as in Fig. 56. There are 35 contours, all separate. The bi-partitioning algorithm was applied to Fig. 56 and contours were partitioned into two parts. Fig. 57 is the resulted X-partition with 15 contours and Fig. 58 is Y-partition with 20 contours. The vertices of 4 degrees as well as 6 were handled successfully. By the way, the parameters, $\alpha = 1.1$ and $\beta = 4$, were used as before. Fig. 59 is the colored checkered-like patterns with X-partition blue including the most outer region, Y-partition red and boundaries white.

1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427

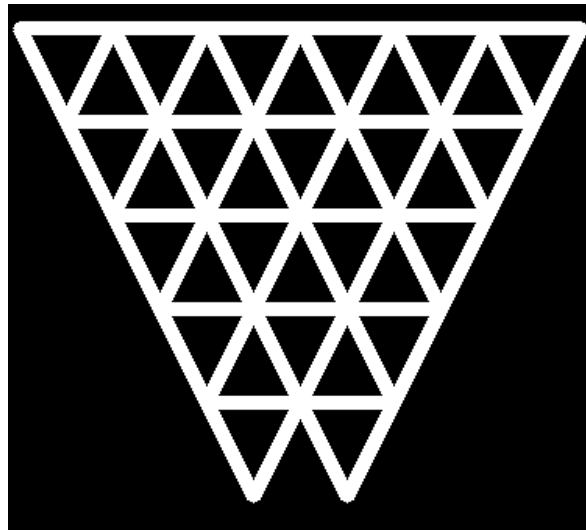


Fig. 55 Original drawing of triangles with 6 degree vertices and line width of 8

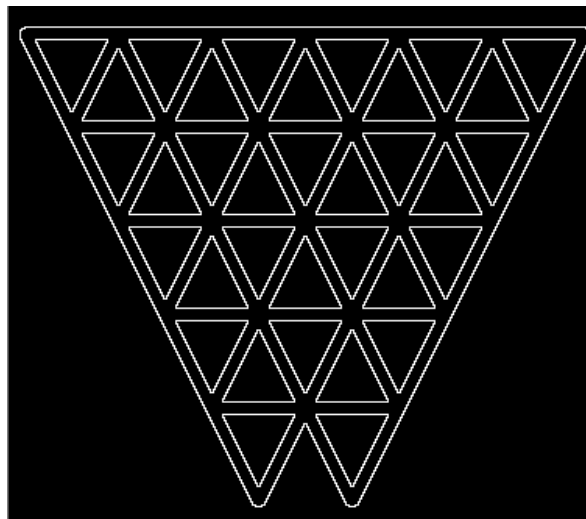


Fig. 56 35 contours extracted

1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480

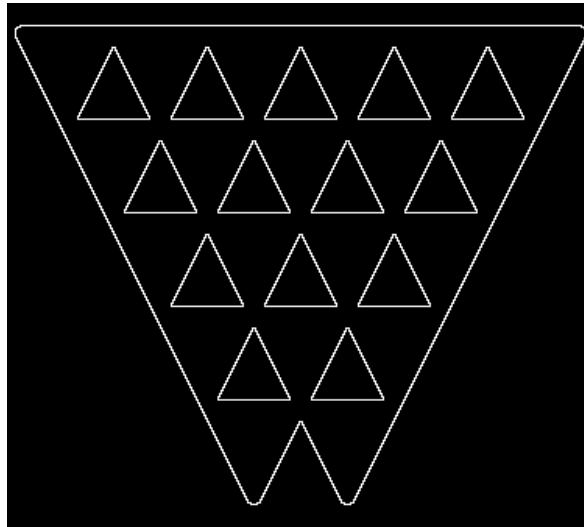


Fig. 57 X-partition, 15 contours

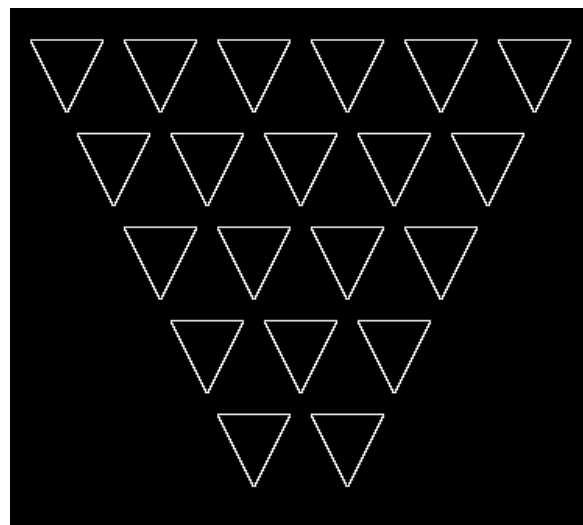


Fig. 58 Y-partition, 20 contours

1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533

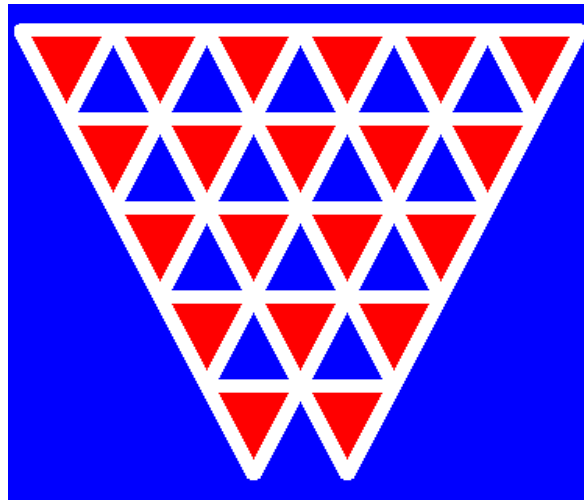


Fig. 59 Checkered-like patterns: X-partition blue, Y-partition red and boundary white

4.9 TRIANGLES WITH SIX DEGREE VERTICES AND LINE WIDTH 1

Fig. 60 is the figure of triangles with 6 degree vertices. It was drawn using OpenCV library with line thickness of 1. Fig. 61 is the colored checkered-like patterns with X-partition blue, but the most outer region green and Y-partition and boundary yellow. This example also successfully demonstrates the bi-partitioning algorithm worked well with 6 degree vertices. The parameters, $\alpha = 1.1$ and $\beta = 4$, were used as before.

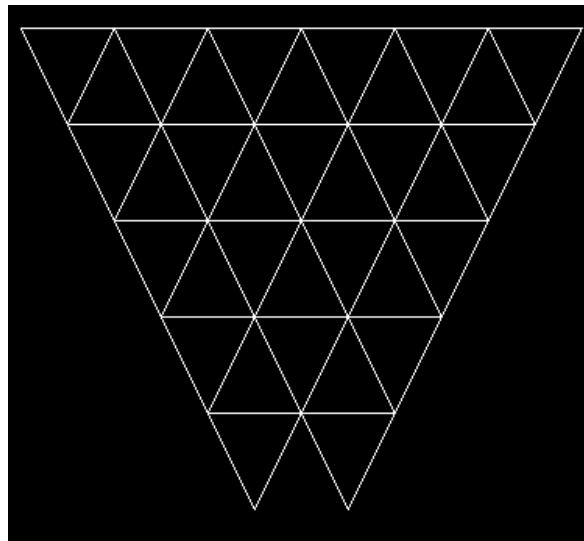


Fig. 60 Original drawing of triangles with 6 degree vertices and line width of 1

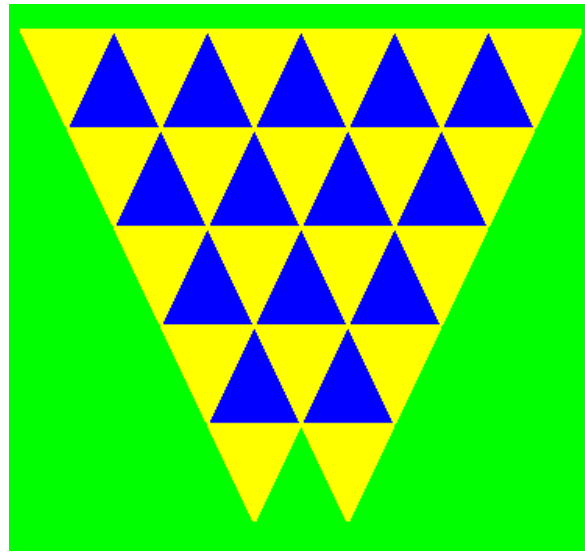


Fig. 61 Checkered-like patterns: X-partition blue, the most outer region green and Y-partition yellow

5. LINE WIDTH AND ADJACENCY BETWEEN CONTOURS

A number of figures with different drawing line widths were tried for generating checkered patterns and their likes in the experiments. This is to see the effects of the different line widths. It turned out that different line widths cause no problem in algorithm execution and give successful outputs with different impressions. Figs. 7 and 14, Figs. 18 and 24, Figs. 29 and 34, Figs. 39 and 49, and Figs. 44 and 54, and Figs. 59 and 61 are the examples. These different line widths were specified by the parameters, but once the figures were drawn by hand or by the program, all the checkered-like patterns were automatically generated by the program. Specifications such as line width for different original figures are given in Table 1.

The bi-partitioning was done based on the adjacency between two contours. The adjacency criterion is the distance between two contours. In the experiments, the overall average of the shortest distance from a point on a contour to other contours as (4) was first computed. The values vary depending on the drawn lines, especially on the line widths. This distance was used as the criteria for deciding if two contours are adjacent. The averages (d_{As}) for Figs. 15, 20, 25, 30, 35, 45, 40, 50, 55, and 60, respectively, are shown in Table 1. This table also includes the shape type of figures and the line width originally given.

The adjacency is based on how many adjacent points exist between two contours. The minimal count was given by β of the expression (7) with the parameter $\alpha = 1.1$. $\beta = 4$ was used for all the experiments except for Fig. 40 ($\beta = 10$ was used). This figure of Fig.40 is with the outer rectangle frame. Many of the line crossing parts are relatively long for Fig. 40 than other figures. The adjacency checking had to be more severe for this kind of figures. That is, at those places where ovals touch the outer frame, the crossing parts are inclined to be longer and misleading. With proper parameters manually given at the program start, the program produces automatically the checkered-like patterns at once. All the experiments successfully demonstrated that the bi-partitioning algorithm is very robust.

* Corresponding author: kurokawa@aitech.ac.jp

The suitable parameters (α and β) are dependent on the figure drawn situations. If the line width changes while the figure is drawn, the overall average distance (d_{AS}) might not work properly. But, as the experiments show, $\alpha = 1.1$ seems to be proper under the condition of no line width change. But smaller α might be better, more severely distinguishing the real adjacency and the false one. The parameter β , together with α has to work to distinguish between the real adjacency and the fake one. Fake adjacency comes often from the long intersecting or touching part between two contours which are not adjacent. This fake adjacency was caused, when $\beta = 4$, in the experiment of Fig.40, the free hand drawn ovals touch the straight frame line, which could be considered as an unusual type of figure. For most cases, β can be arbitrary within a roughly estimated reasonably range. But β cannot be too large for the case where contours are not long enough. If proper parameters are computed automatically, it's better.

The contour cycles of Fig. 2 were all on the lines of rectangles with line thickness of 1 (as of OpenCV). This means that the distance between any two adjacent regions was zero. The computed average distance was also zero. Rectangles of Fig. 15, Circles of Fig. 25 and triangles of Fig. 60, all with the line width 1, give the distance of 0.0, 0.28 and 0.25, respectively. It seems that horizontal or vertical lines give the shorter distance. Rectangles of Figs. 20 and triangles of Fig. 55 are with line width of 5 and 8, respectively, and the computed average distance was 6.0 and 7.8, respectively. The distance does not seem to be proportional to the width. Freehand ovals of Figs. 35 and 40 were drawn with line width 1. But the d_{AS} is slightly smaller for Fig. 35 than Fig. 40. It should be due to the crossing parts between ovals and outer frame.

Table 1 Figure specifications and average of shortest distance (d_{AS})
Unit shape symbol explanation: r means rectangle, c circle, f freehand drawing, t triangle. Line thickness of Figs. 35 and 44 are by MSWORD (point), others are by OpenCV library

Fig. NO	15	20	25	30	35	45	40	50	55	60
Unit shape	r	r	c	c	f	f	f	f	t	t
Line thickness	1	5	1	3	1pt	5pt	1pt	5pt	8	1
d_{AS}	0.0	6.0	0.28	3.4	1.01	7.0	1.03	7.03	7.8	0.25

6. CONCLUSION

The method of this study demonstrated its ability to generate varieties of checkered patterns and their variations with various features including:

- 1) Wide range of patterns can be produced.
- 2) It can be applied to drawings of various closed line shapes as circles, rectangle, triangles, freehand drawing, etc.
- 3) Width of the line or curve can vary.
- 4) It is automatic and very robust as demonstrated.

REFERENCES

- [1] <https://www.google.co.jp/search?q=tokyo+olympic+games+2020+emblem&espv=2&biw=1920&bih=1060&tbm=isch&tbo=u&source=univ&sa=X&ved=0ahUKEwiG-72Jjb7PAhVFfJQKHZnJCJAQsAQIKA>, accessed 2016/10/3

- [2] <https://www.google.co.jp/search?q=checkered+pattern&espv=2&biw=1920&bih=1060&tbm=sch&tbo=u&source=univ&sa=X&ved=0ahUKEwj7vPCQ9b3PAhVCVZQKHT8KAUYQsAQIHg&dpr=1>, accessed 2016/10/3
- [3] T. Kurokawa, "Maze Construction by Using Characteristics of Eulerian Graphs," British Journal of Mathematics & Computer Science, Vol.9, Issue.6, pp.453-472, 2015.6, SCIENCE DOMAIN international, UK, ISSN: 2231-0851, <http://sciencedomain.org/issue/1147>
- [4] Open CV Library, <https://msdn.microsoft.com/ja-jp/library/dn188695.aspx> Accessed 2016/12/30
- [5] S. Suzuki and K. Abe, "Topological Structural Analysis of Digitized Binary Images by Border Following," CVGIP 30 1, pp 32-46, 1985
- [6] A. Rosenfeld and A. C. Kak, Digital Picture Processing, 2nd ed., Vol. 2, Academic Press, New York, 1982.
- [7] S. Yokoi, J. Toriwaki, and T. Fukumura, "An analysis of topological properties of digitized binary pictures using local features," Comput. Graphics Image Process., Vol.4, pp.63-73, 1975
- [8] M. K. Agoston, Computer Graphics and Geometric Modeling, Springer, 2005
- [9] T. Kurokawa, Kazumasa Mori (Shizuoka Information Processing Center Co., Ltd.), and Teruaki Mizuno (Daiichi System Engineering Co., Ltd.), "Automatic Construction of Picture Maze by Repeated Contour Connection," Thailand-Japan Joint Conference on Computational Geometry and Graphics 2012, pp.5-6, Bangkok, Thailand, 2012.12
- [10] 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>
- [11] T. Kurokawa, "Construction of Picture Maze along Set of Image Dot Vertices," British Journal of Mathematics & Computer Science, Vol.17, Issue.1, pp.1-14, 2016.5, SCIENCE DOMAIN international, UK, ISSN: 2231-0851, <http://sciencedomain.org/issue/1786>
- [12] K. K. Meng, D. Fengming and T. E. Guan, Introduction to Graph Theory, World Scientific, 2007
- [13] J. Matousek and J. Nešetřil, Invitation to Discrete Mathematics, Second Edition, Oxford University Press, New York, 2009.