

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

ABSTRACT

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 of the line figures 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, and it 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.

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.

2. OUTLINE OF THE STUDY AND THE RELATED REFERENCES

The procedure of this study first extract contours from an original binary line figure, which is supposed to constitute Eulerian Circuit [4]. 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. So, if two 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 two contours, the contours can be bi-partitioned. The Bi-partitioning algorithm [3] separates them into X-partition and Y-partition. This algorithm works without or before organizing the adjacency graph, or region adjacency graph (RAG). 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.

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 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. It is known that the region adjacency graph (RAG), the dual [5], 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. The algorithm introduced 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:

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

METHOD:

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

92 X, Y: partition sets of the bipartite graph, both sets are empty at the beginning; they will have
 93 the resulted vertices of bi-partition.
 94 1.Remove an arbitrary $cycle_j$ from *CONTOURS* and put it in X;
 95 2.Repeat begin
 96 3. For each j in X do begin
 97 4. For each i in *CONTOURS* do begin
 98 5. If j and i are adjacent, do begin
 99 6. Remove i from *CONTOURS* and put it in Y;
 100 7. If no cycle remains in *CONTOURS*, do begin
 101 8. If the outer cycle is in Y, exchange X and Y;
 102 8. Paint inside all CONTOURS with different colors for X and Y and finish;
 103 10. end;
 104 11. end;
 105 12. end;
 106 13. end;
 107 14. Exchange X and Y;
 108 15.end;

109 110 **CONTOUR ADJACENCY:**

111 The adjacency checking can be done by distance computation between two cycles. Let two
 112 $cycle_j$ and $cycle_i$ are defined; they are adjacent if there are more than a certain number, β , of
 113 points on $cycle_j$ such that the shortest distance from a point on $cycle_j$ to $cycle_i$ is less than
 114 the overall average of the shortest distance (d_{AS} defined below) multiplied by a certain number,
 115 α . Let $d_s(j, p_l, i)$ be the shortest distance from the point p_l on $cycle_j$ to $cycle_i$. $d_s(j, p_l, i)$ and
 116 d_{AS} are expressed as:

$$117 \quad d_s(j, p_l, i) = \min_{1 \leq k \leq n_i} d(j, l, i, k); \quad (4)$$

$$118 \quad d_{AS} = \frac{1}{\sum_{j=1}^c n_j} \sum_{j=1}^c \sum_{l=1}^{n_j} \min_{1 \leq i \leq c} \min_{1 \leq k \leq n_i} d(j, l, i, k). \quad (5)$$

121
122 And $d(j, l, i, k)$ is defined as

$$123 \quad d(j, l, i, k) = \sqrt{(x(i, k) - x(j, l))^2 + (y(i, k) - (y(j, l)))^2}, \text{ where} \quad (6)$$

124
125
126 $j \neq i$ and n_j is the number of points on $cycle_j$. So if more than β such different points on j
 127 exist, the two cycles j and i are considered adjacent. That is, cycles j and i are adjacent if the
 128 size of S is greater than β in the following expression:

$$129 \quad S = \{p_l \mid d_s(j, p_l, i) \leq \alpha * d_{AS} \text{ and } 1 \leq l \leq n_j \text{ where } \alpha \text{ is a small positive value}\}. \quad (7)$$

130 131 132 **2.2 EXPERIMENT WITH BI-PARTITIONING AND REGION FILLING**

133
134 By applying the above algorithm to Fig. 2, the contour cycles are bi-partitioned to X-partition
 135 (Fig. 4) and Y-partition (Fig. 5). Although they are interchangeable, the partition with the most
 136 outer contour is designated as X-partition in this study. X-partition includes the contours of 1,
 137 and 4. While Y-partition holds 2, 3, 5, and 6. Fig. 6 is obtained by examining all the adjacencies
 138 between the contours. The graph (yellow) in Fig. 3 and the one in Fig. 6 represent the same
 139 graph. Filling the regions of X-partition with yellow, Fig. 7 is obtained, so called checkered
 140 patterns.

141
142

143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195

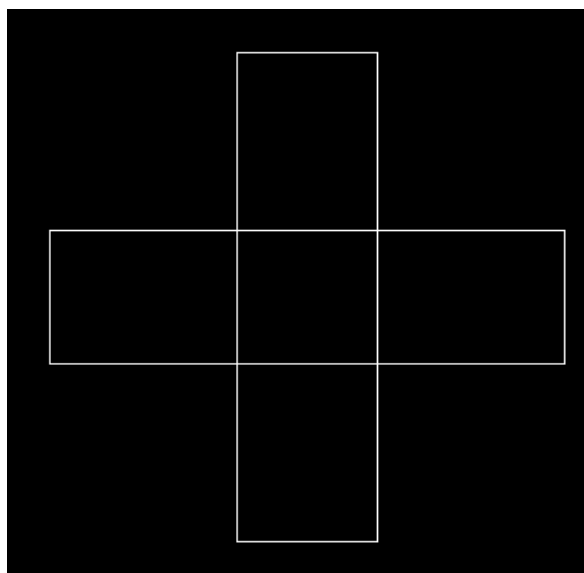


Fig. 1 Original drawing of two rectangles overlapped with line width of 1

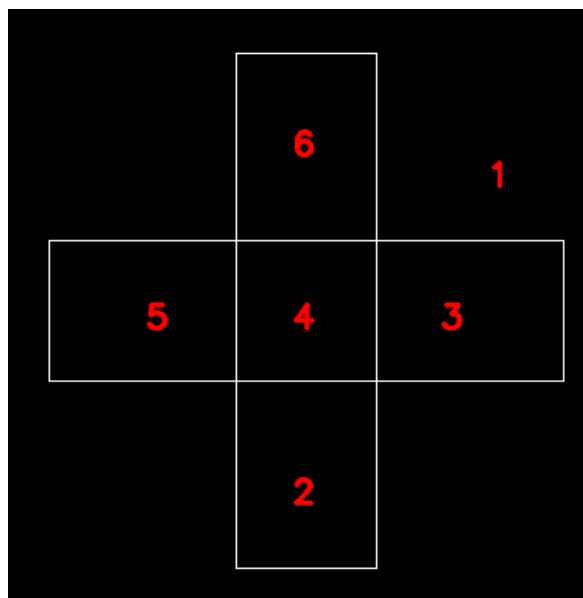


Fig. 2 Extracted contours of two rectangles overlapped, 6 contours

196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248

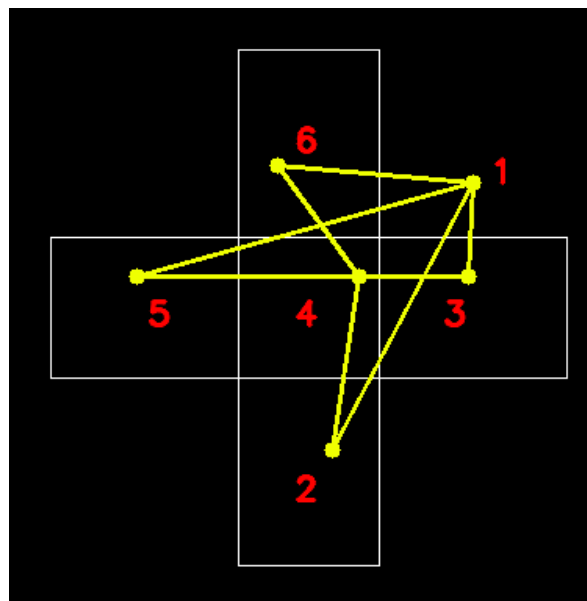


Fig. 3 Extracted contours constitute a bipartite graph; the regions are numbered

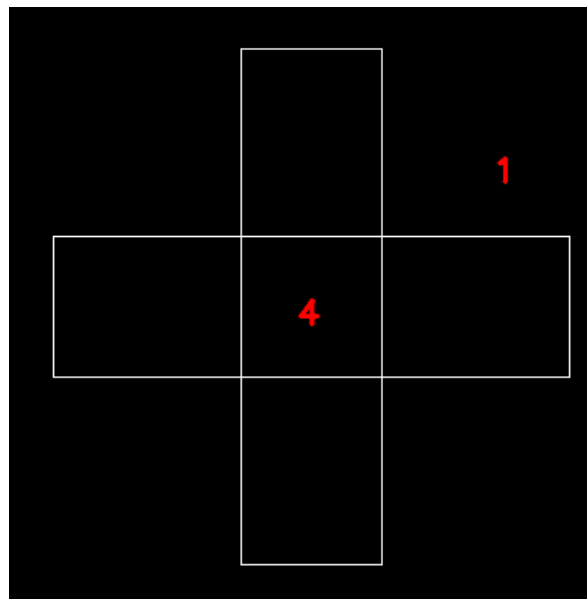


Fig. 4 X-partition, 2 contours

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
300
301

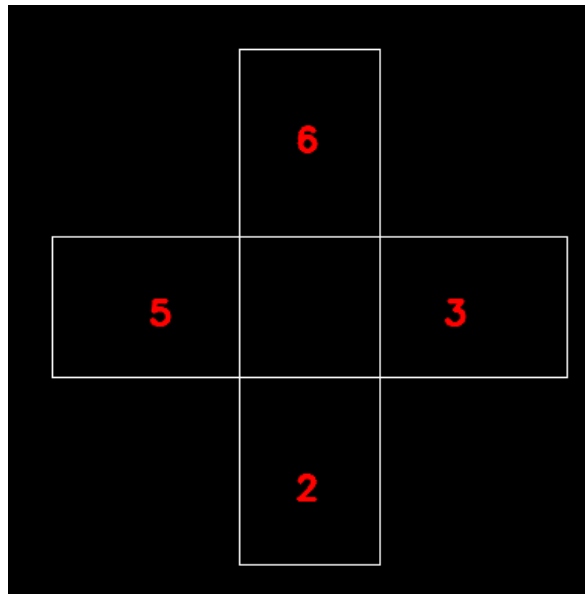


Fig. 5 Y-partition, 4 contours

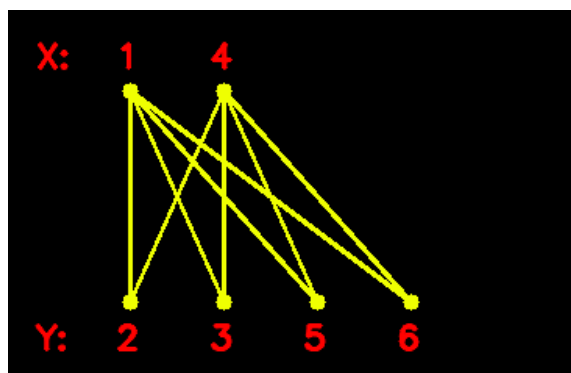


Fig. 6 bi-partitioned graph

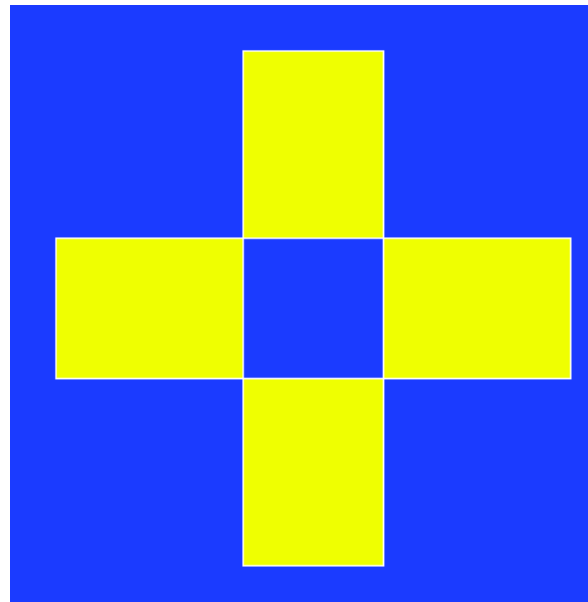


Fig. 7 Checkered patterns: X-partition is filled with blue; Y-partition and boundaries with yellow

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; 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 RAG of Fig. 3. And the graph of Fig. 13 is the same as that of Fig. 6. 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 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.

355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404

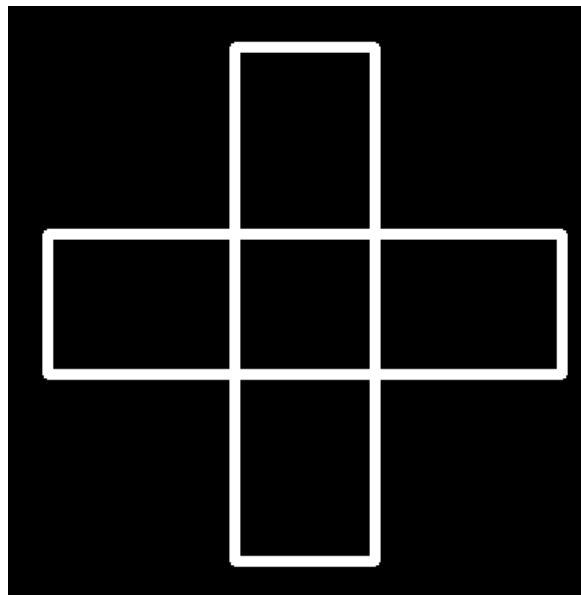


Fig. 8 Original drawing of two rectangles overlapped with line width of 5

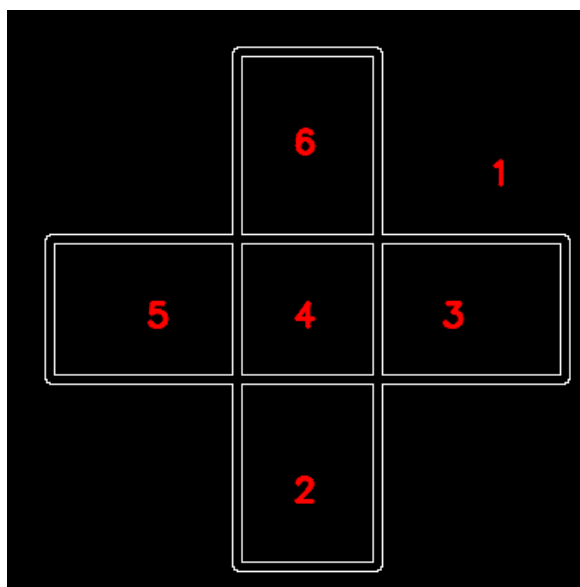


Fig. 9 Extracted contours, 6 contours; regions are numbered

405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448

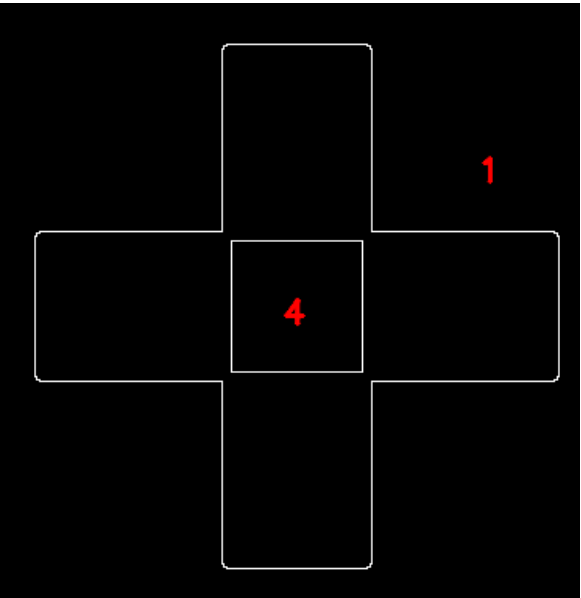


Fig. 10 X-partition, 2 contours

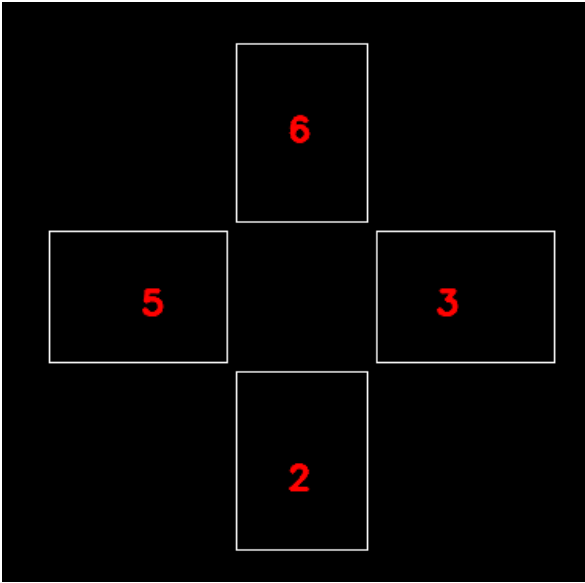


Fig. 11 Y-partition, 4 contours

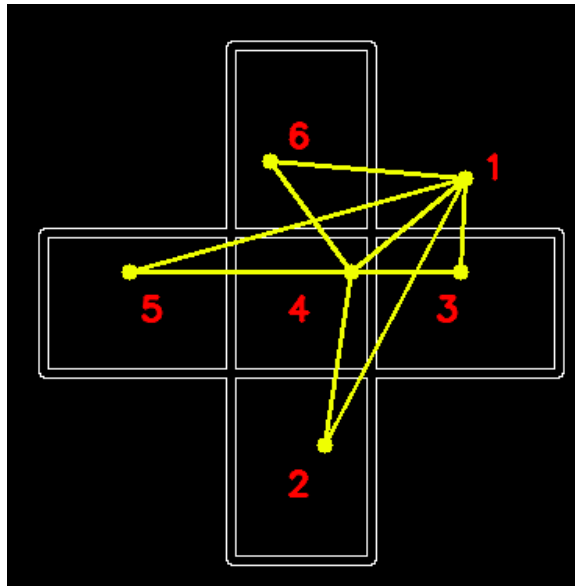


Fig. 12 Extracted contours constitute a region adjacency graph (RAG)

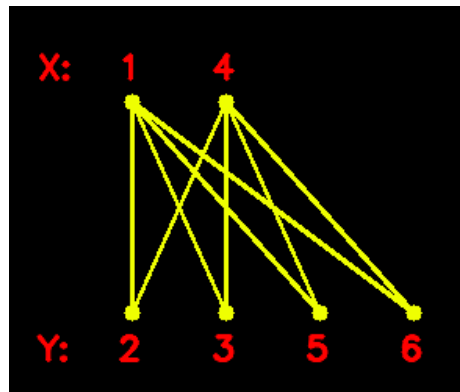


Fig. 13 Bi-partitioned graph

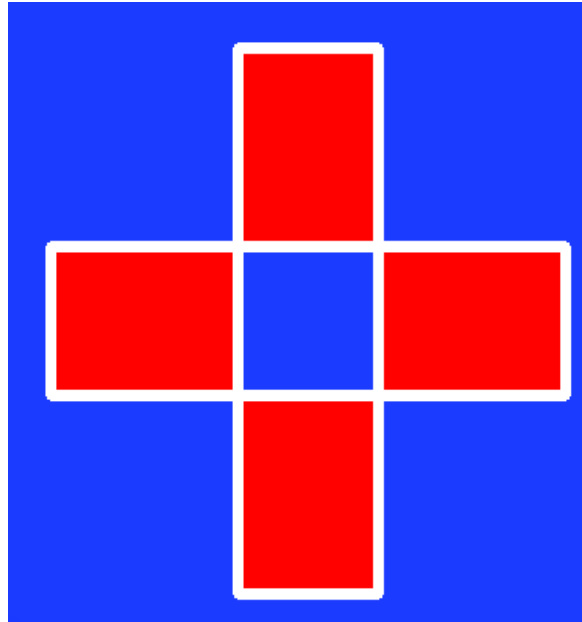


Fig. 14 Checkered patterns: X-partition blue;
Y-partition red; boundaries white

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.

541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585

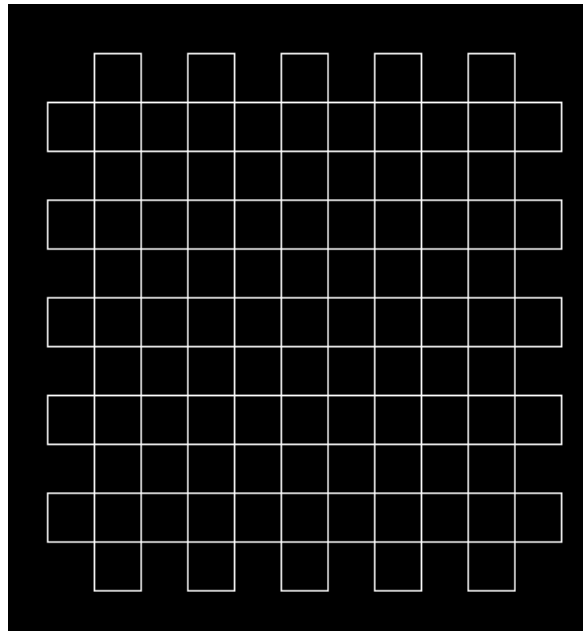


Fig. 15 Original drawing of grids: 10 overlapped rectangles with line width of 1

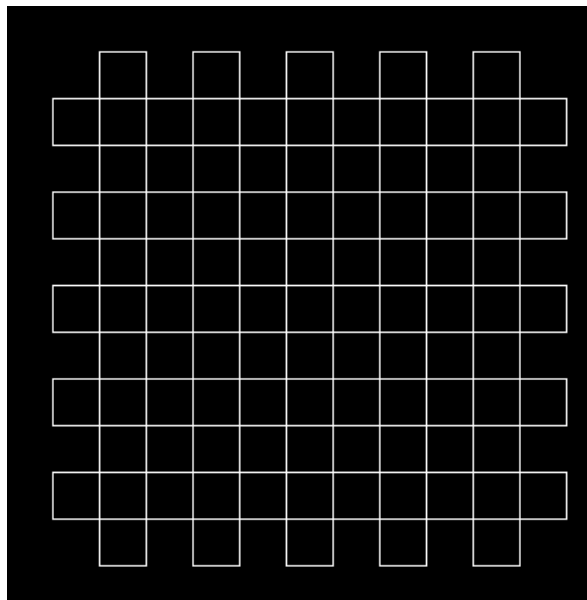


Fig. 16 Contours extracted, 102 contours

586
587
588
589
590
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
619
620
621
622
623
624
625
626
627
628
629

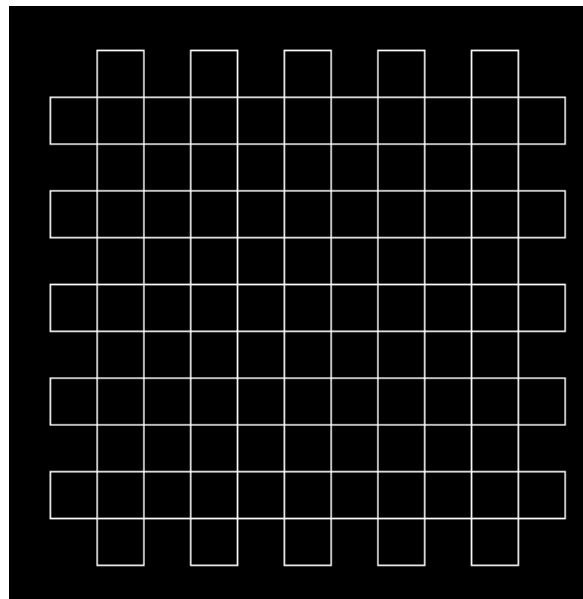


Fig. 17 Y-partition, 60 contours

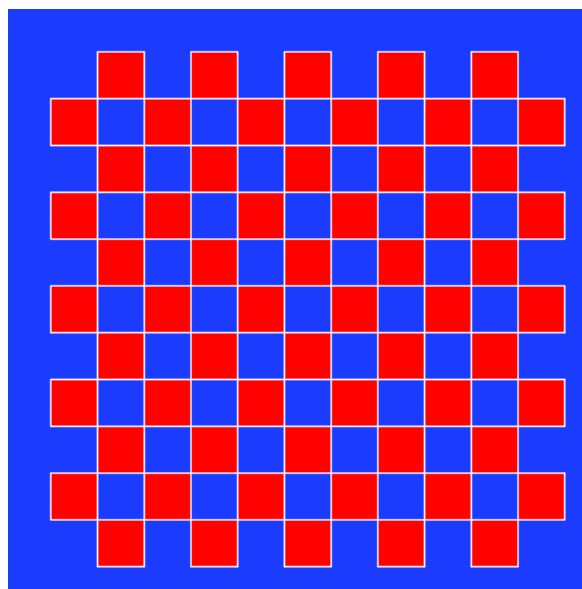


Fig. 18 Checkered patterns: X-partition with blue; Y-partition with red; boundary white

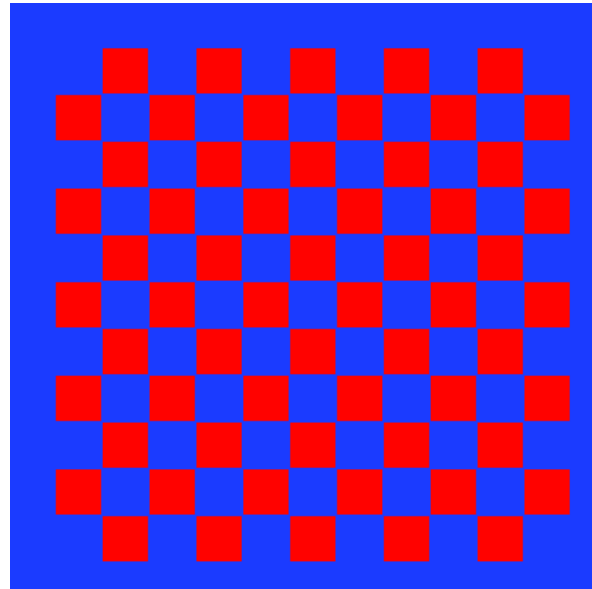


Fig. 19 Checkered patterns: X-partition and boundary blue; Y-partition red

4.3 A LARGE NUMBER OF GRIDS WITH LINE WIDTH 5

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

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
706
707
708
709
710
711
712
713
714
715
716
717
718
719

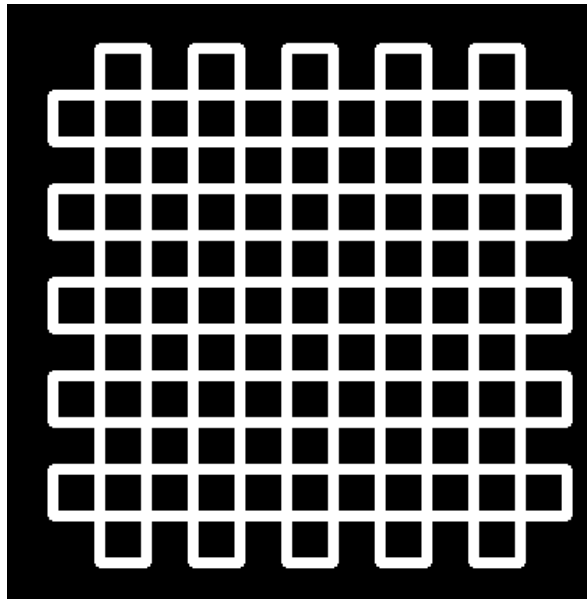


Fig. 20 Original drawing of grids: 10 overlapped rectangles with line width of 5

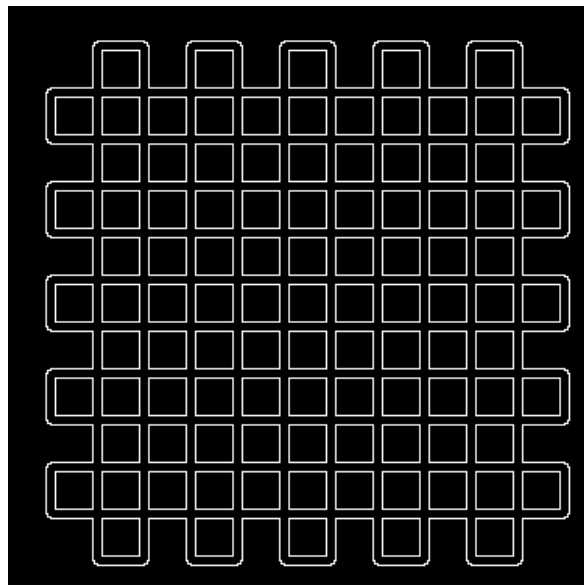


Fig. 21 Contours extracted from Fig. 20, 102 contours

720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763

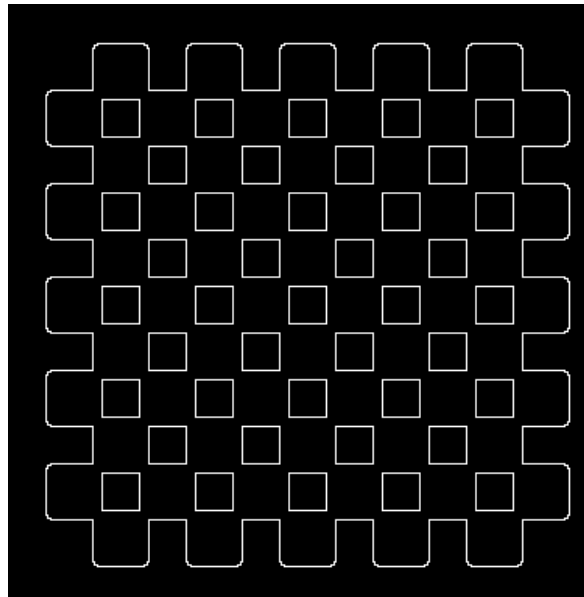


Fig. 22 X-partition, 42 contours

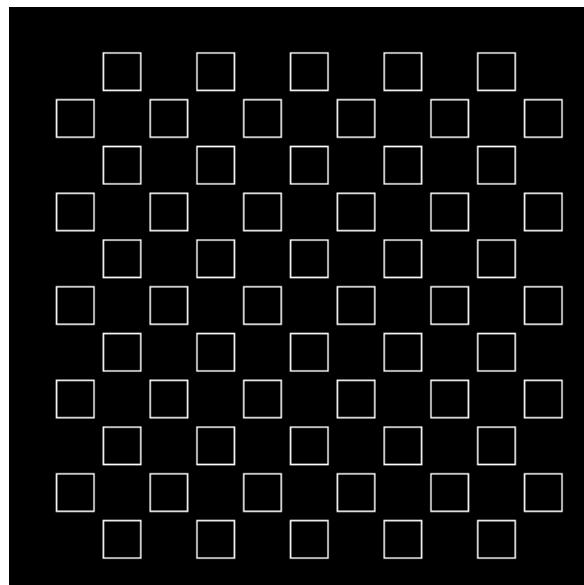


Fig. 23 Y-partition, 60 contours

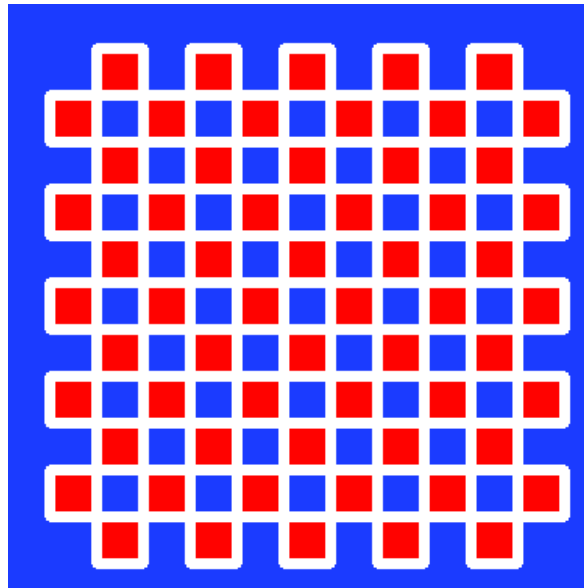


Fig. 24 Checkered patterns: X-partition with blue; Y-partition with red; boundary white

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.

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
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854

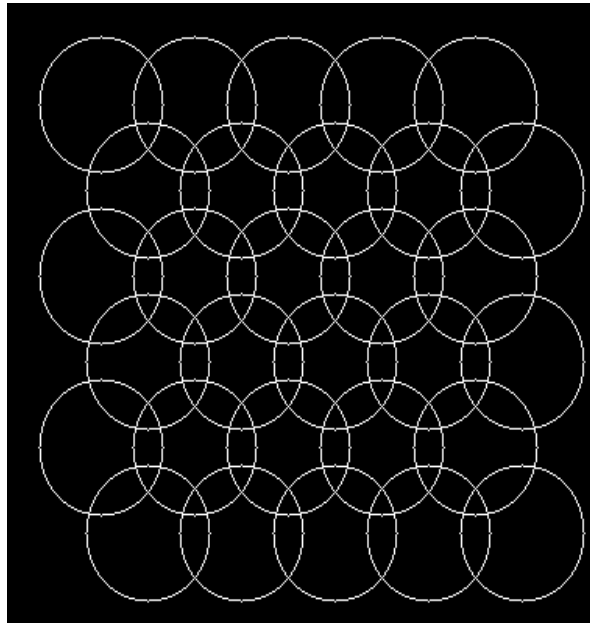


Fig. 25 Original drawing of 30 circles with line width of 1

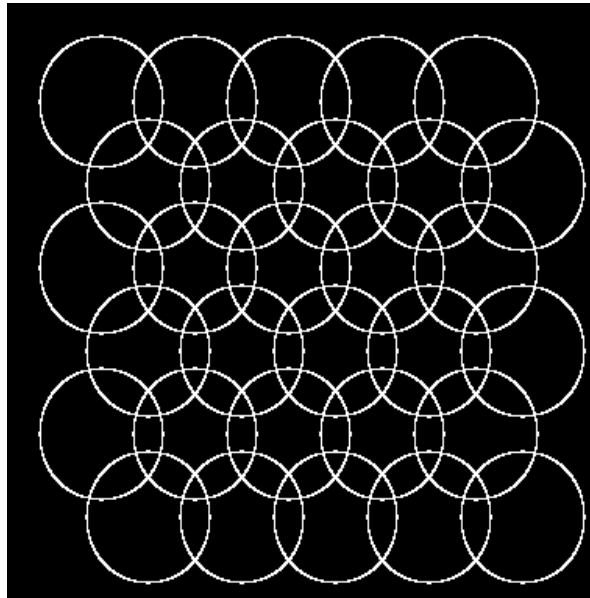


Fig. 26 Contours extracted, 140 contours

855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898

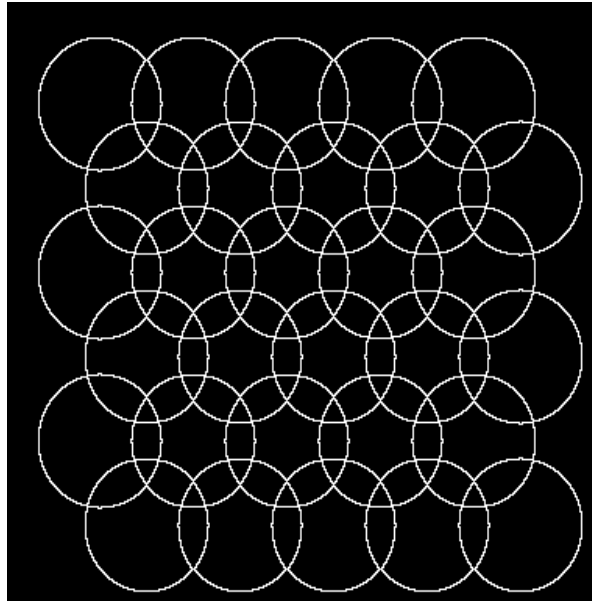


Fig. 27 X-partition, 70 contours

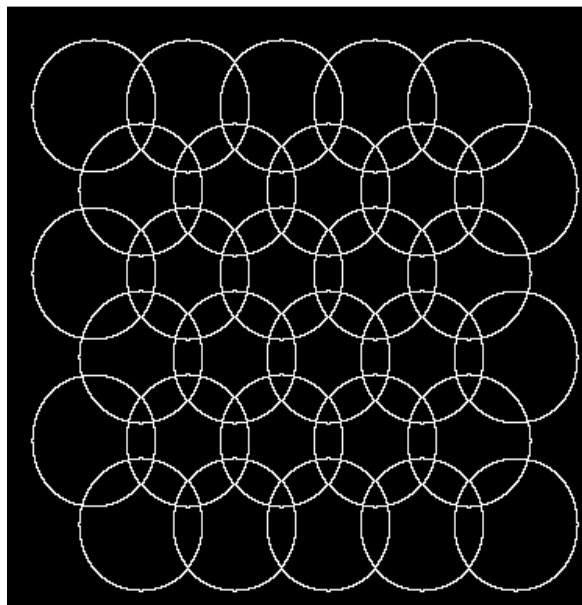


Fig. 28 Y-partition, 70 contours

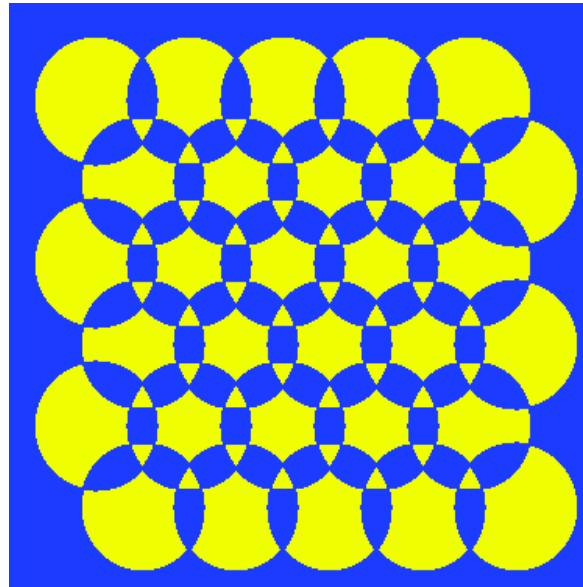


Fig. 29 Checkered-like patterns: with X-partition and boundaries blue, Y-partition yellow

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.

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
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997

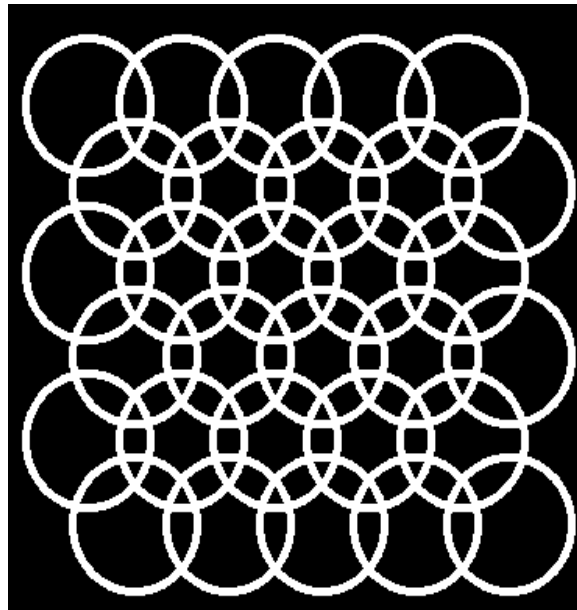


Fig. 30 Original drawing of 30 circles with line width of 3

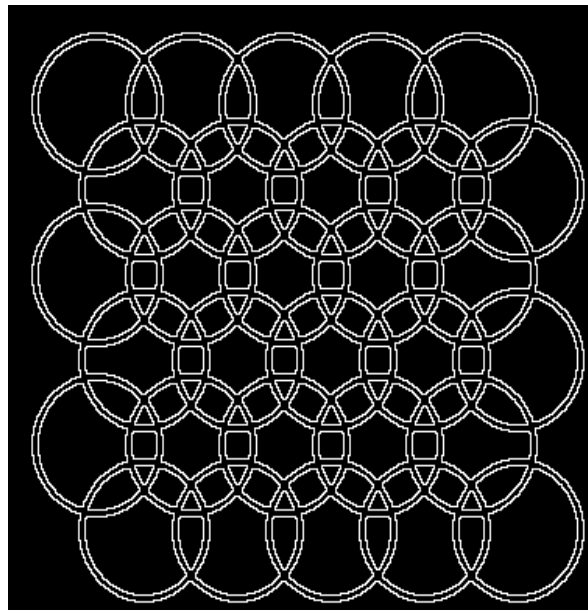


Fig. 31 Contours extracted, 140 contours

998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
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

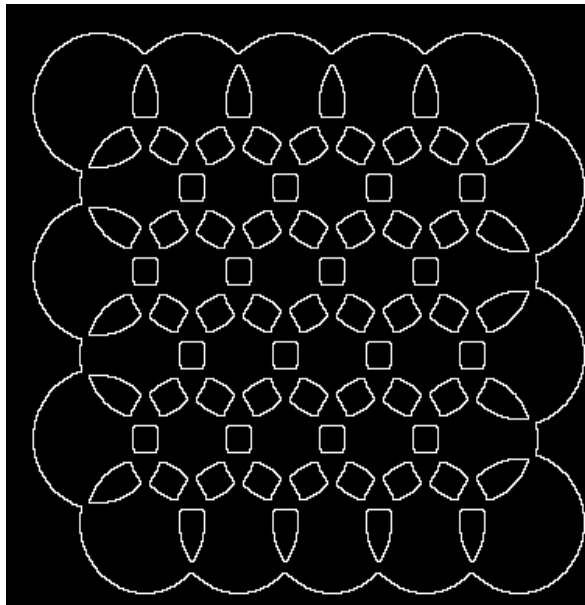


Fig. 32 X-partition, 70 contours

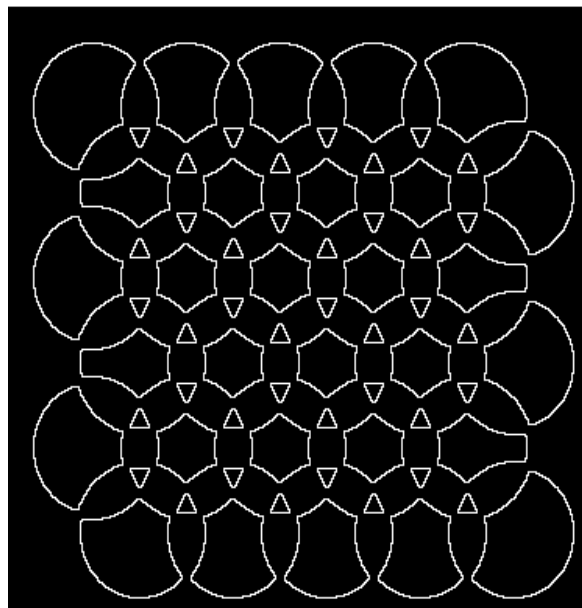


Fig. 33 Y-partition, 70 contours

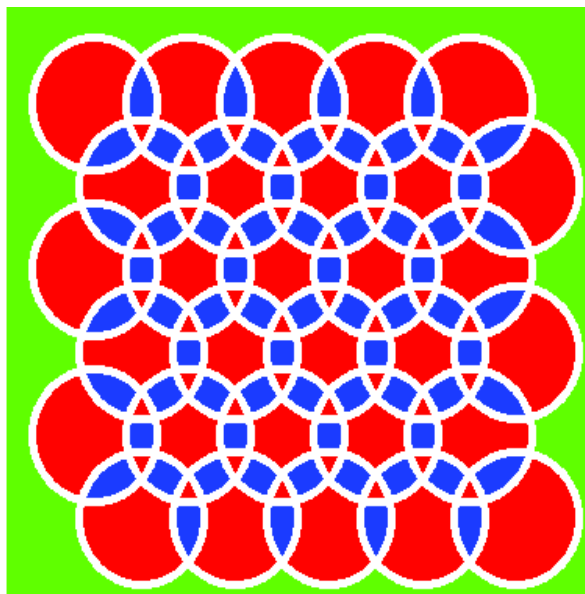


Fig. 34 Checkered-like patterns: with X-partition blue, most outer region green, Y-partition red and boundaries white

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 (5) 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

1104 freehand ovals and rectangle frame lines, the adjacent parts between the two regions which
 1105 are supposed to be in a same partition set are inclined to be longer. This length, sometimes
 1106 causes the confusion. But this was solved by setting β to 10, longer adjacent point count, of
 1107 (7). The final checkered patterns are shown in Fig. 44. Notice that colors of blue and yellow
 1108 are interchanged between Figs. 39 and 44. It is by the interchangeability of X-partition and Y-
 1109 partition.
 1110
 1111
 1112
 1113

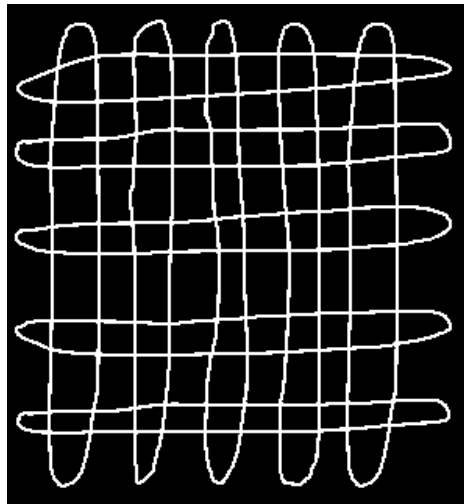


Fig. 35 Original drawing of 10
freehand ovals with line width of 1

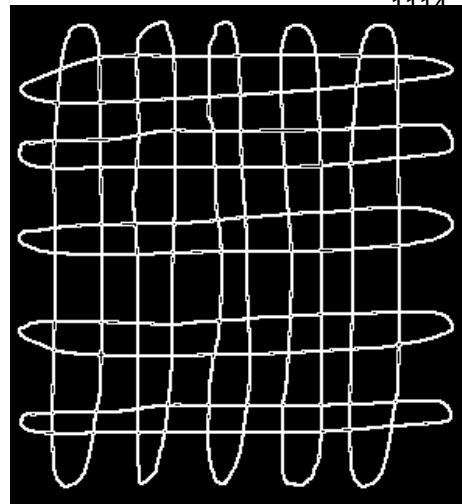


Fig. 36 102 contours extracted

1132
1133

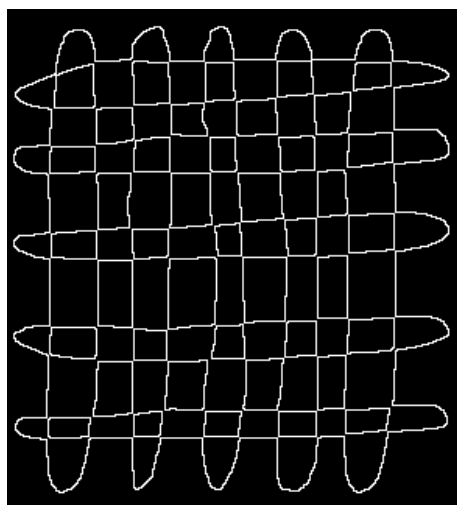


Fig. 37 X-partition, 42 contours

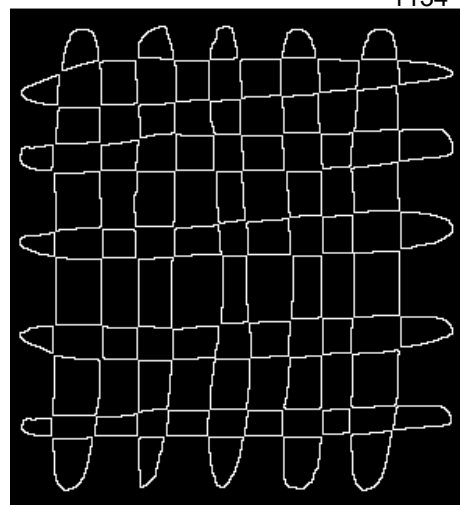


Fig. 38 Y-partition, 60 contours

1150
1151
1152
1153
1154
1155
1156

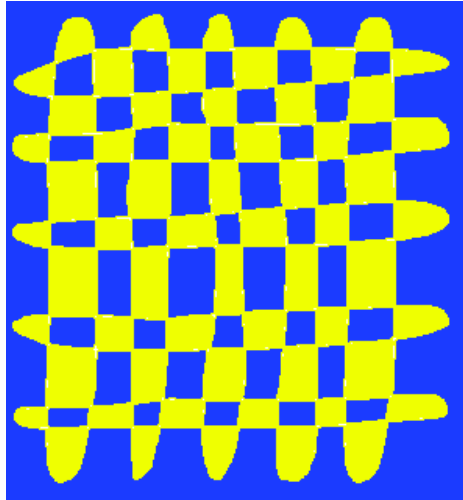


Fig. 39 Checkered-like patterns: X-partition blue; Y-partition and boundaries yellow

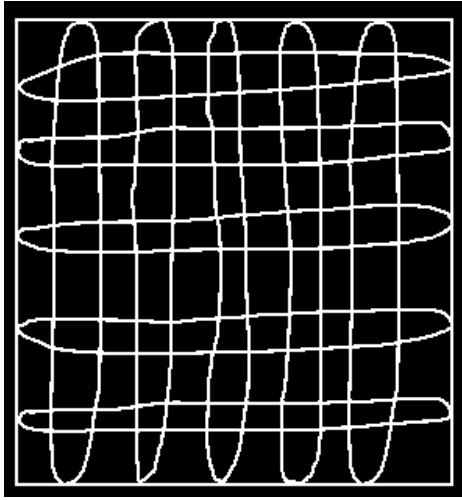


Fig. 40 Original drawing of 10 freehand ovals with line width of 1 and the outer frame

1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176

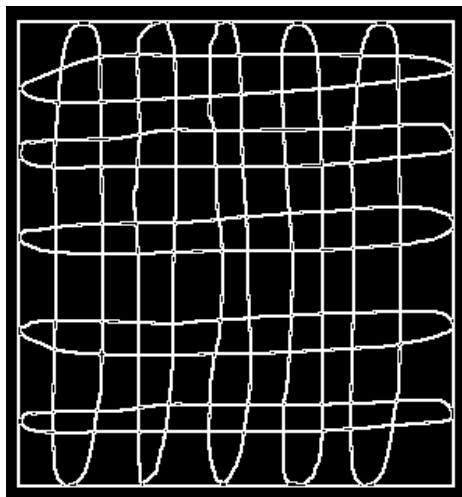


Fig. 41 122 contours extracted

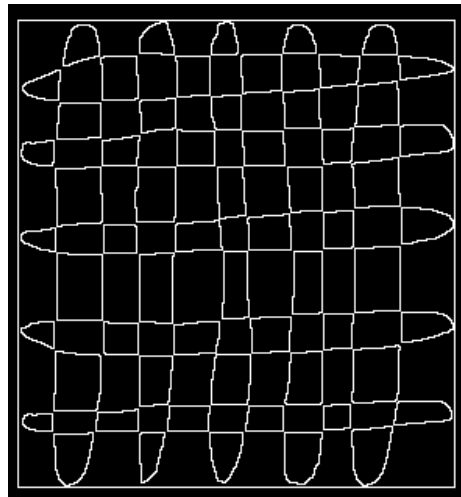


Fig. 42 X-partition, 61 contours

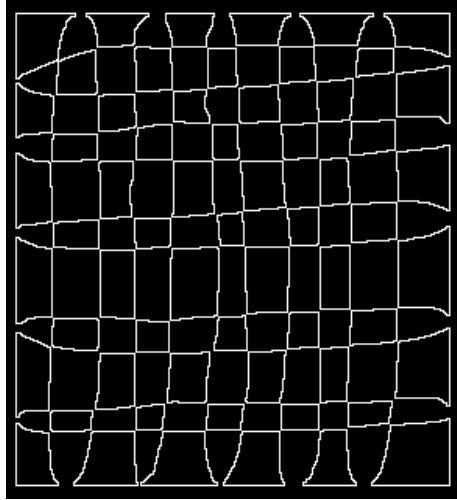


Fig. 43 Y-partition, 61 contours

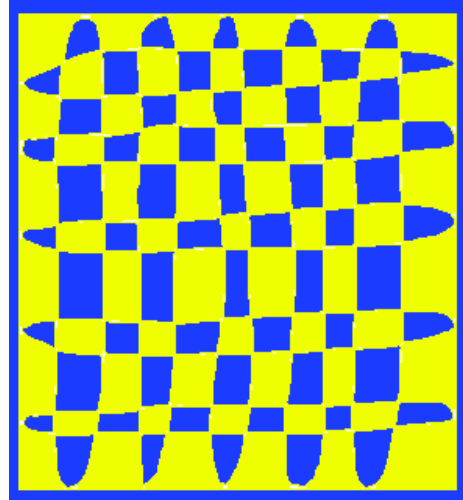


Fig. 44 Checkered-like patterns: X-partition blue, Y-partition and boundary yellow.

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.

1225
1226
1227
1228
1229
1230
1231
1232

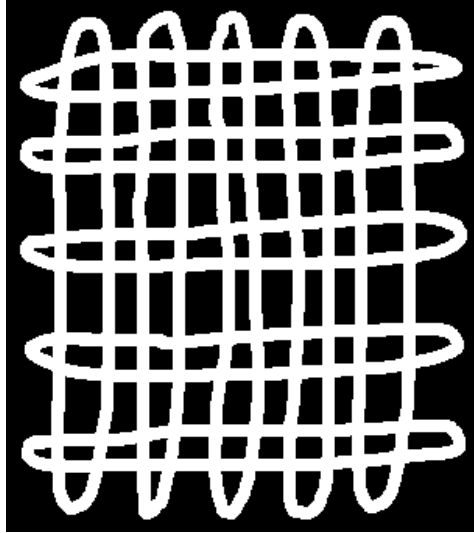


Fig. 45 Original drawing of 10
freehand ovals with line width of 5

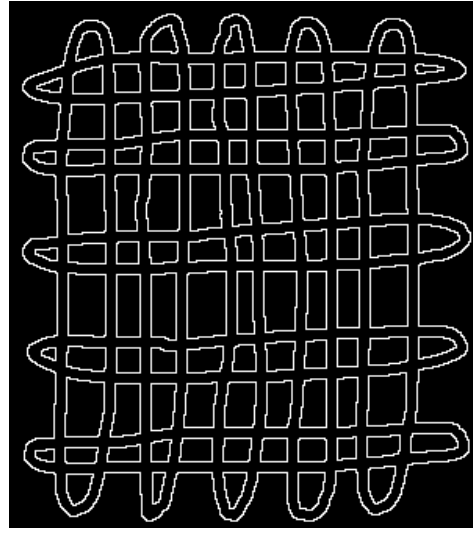


Fig. 46 102 contours extracted

1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256

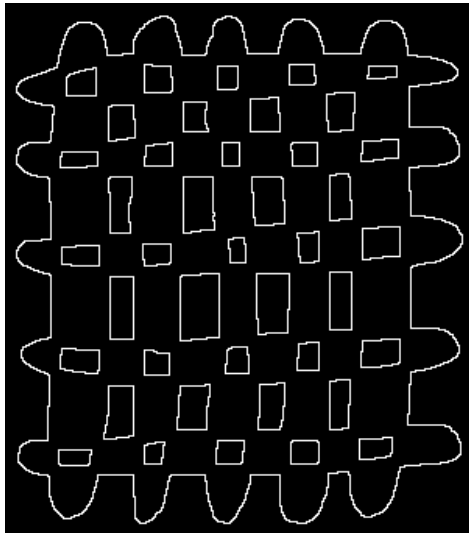


Fig. 47 X-partition, 42 contours

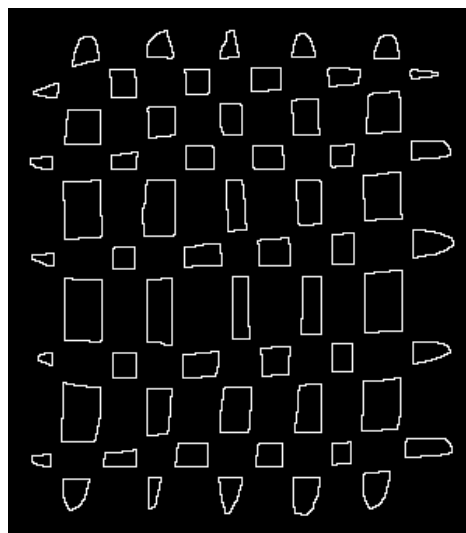


Fig. 48 Y-partition, 60 contours

1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283

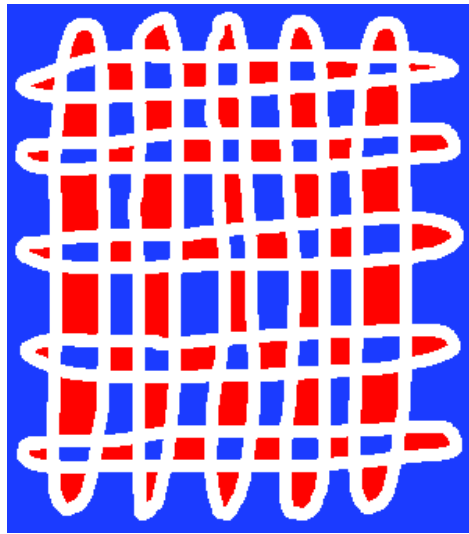


Fig. 49 Checkered-like patterns: X-partition blue, Y-partition red, boundary white

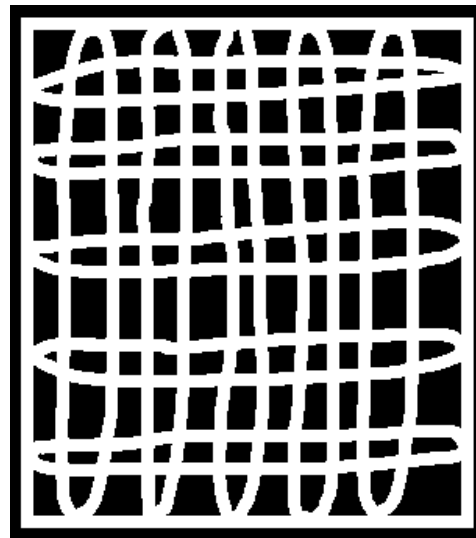


Fig. 50 Original drawing of 10 freehand ovals with line width of 5 and the outer rectangle frame

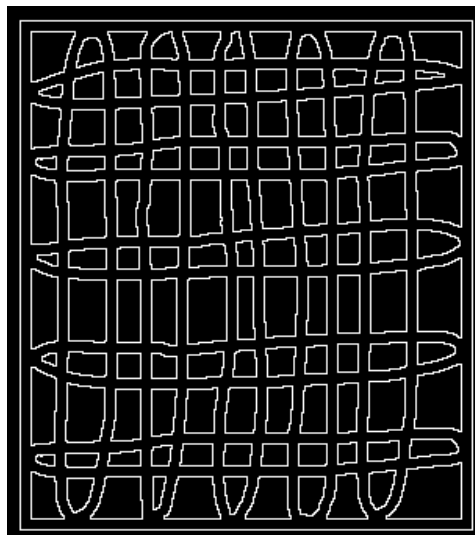


Fig. 51 122 contours extracted

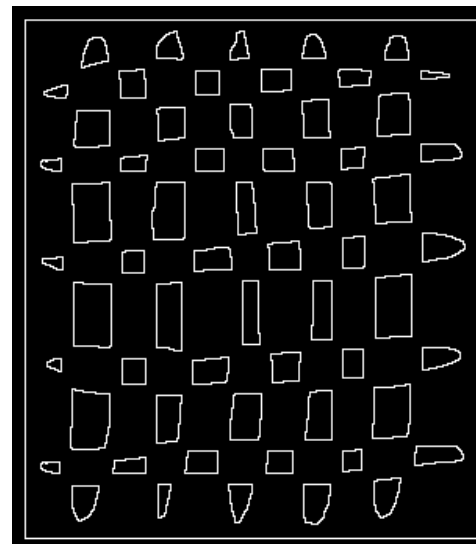


Fig. 52 X-partition, 61 contours

1284
1285
1286
1287
1288

1289
1290
1291
1292

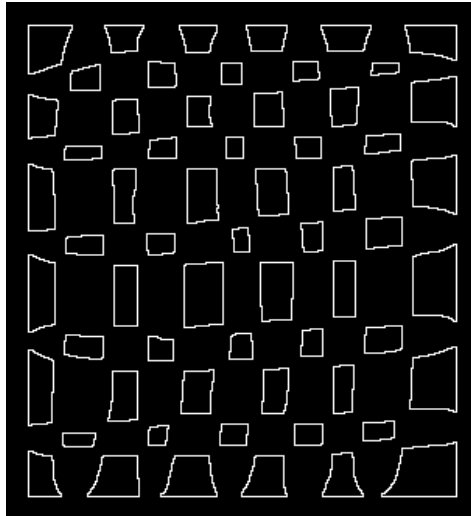


Fig. 53 Y-partition, 61 contours

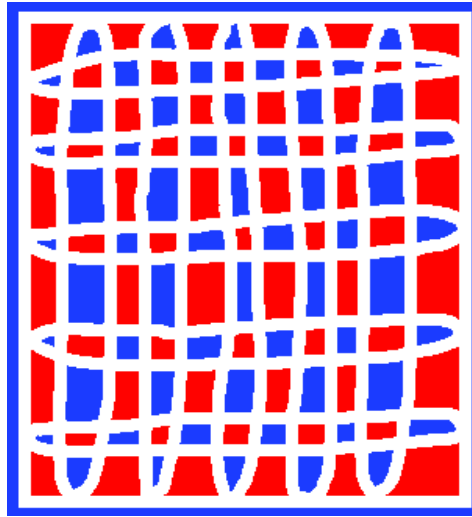


Fig. 54 Checkered-like patterns: X-partition blue, Y-partition red, boundary white

1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320

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.

1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
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

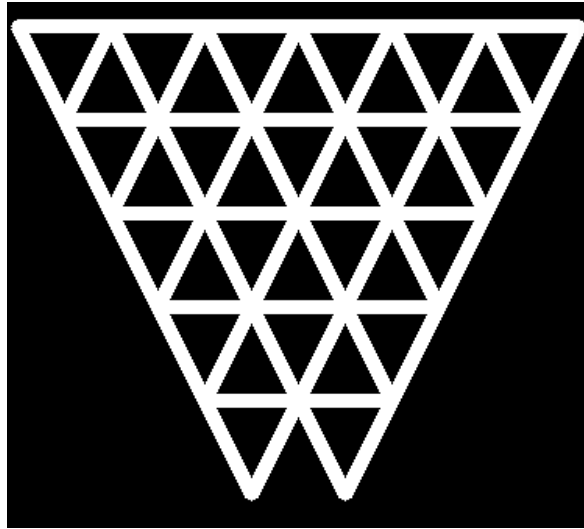


Fig. 55 Original drawing of triangles with 6 degree vertices and line width of 8

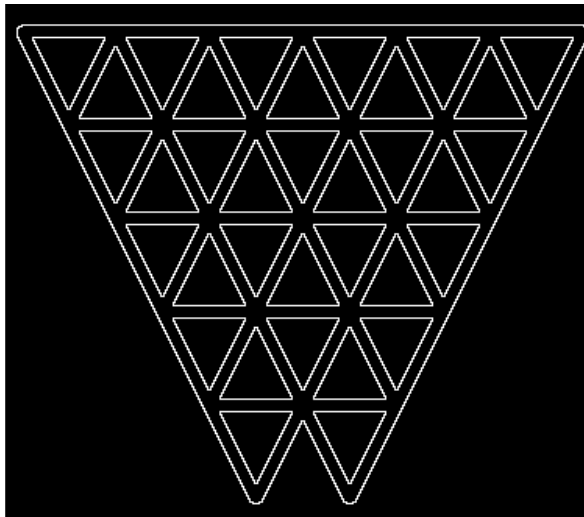


Fig. 56 35 contours extracted

1374
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

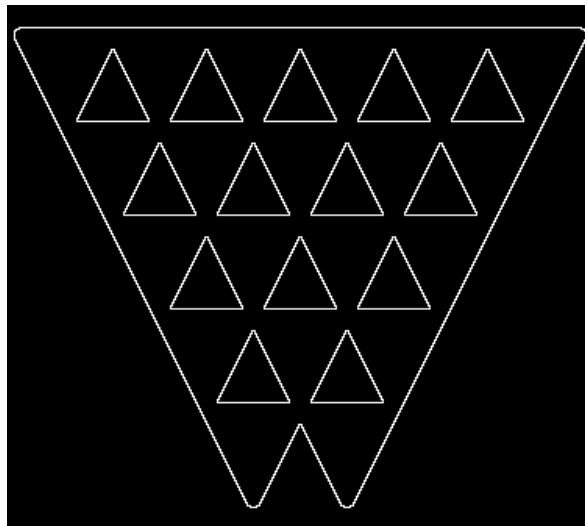


Fig. 57 X-partition, 15 contours

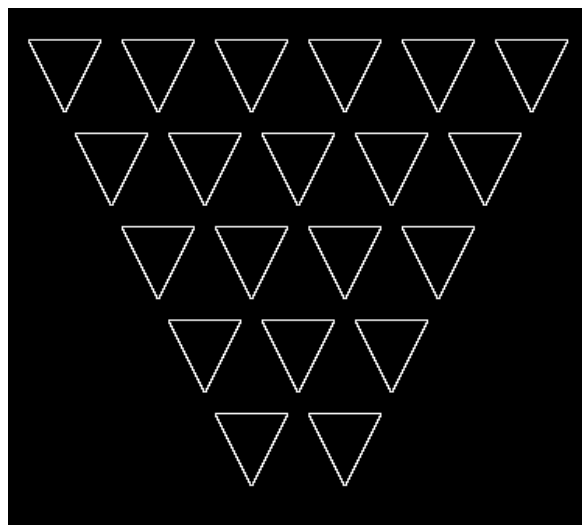


Fig. 58 Y-partition, 20 contours

1427
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

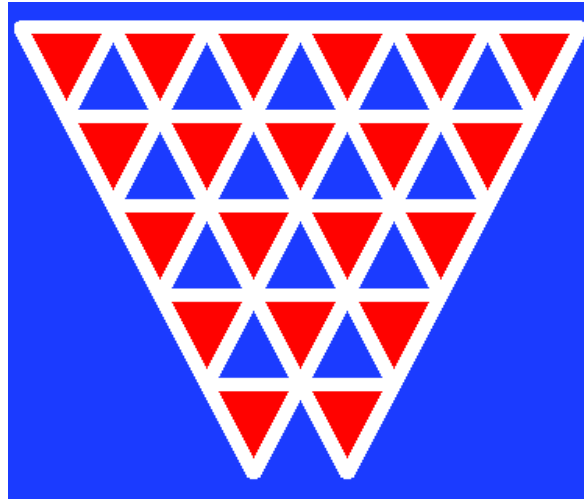


Fig. 59 Checkered-like patterns: X-partition blue; Y-partition red; boundary white

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.

1480
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

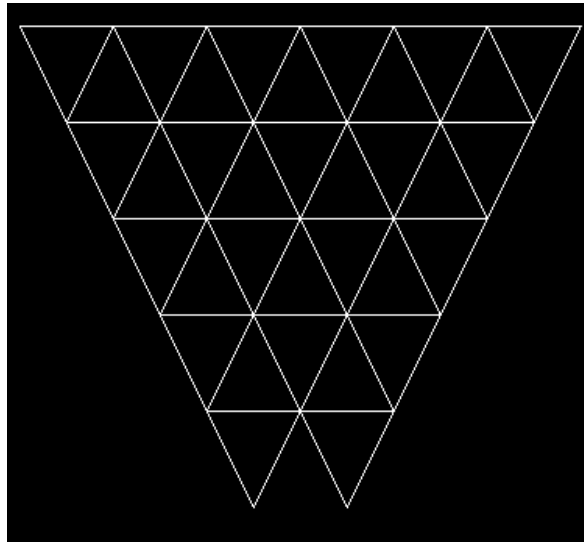


Fig. 60 Original drawing of triangles with 6 degree vertices and line width of 1

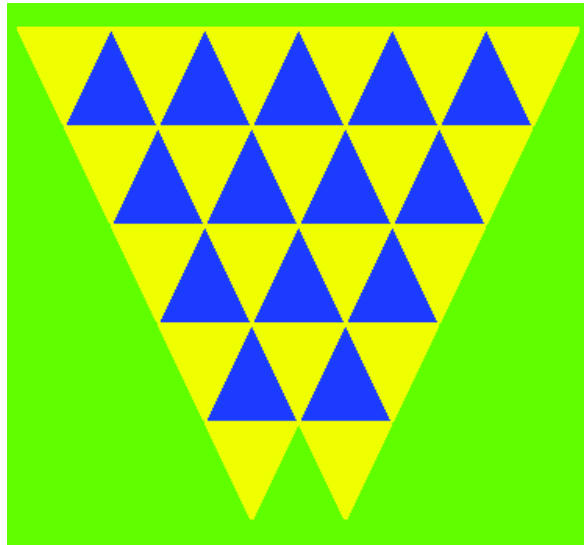


Fig. 61 Checkered-like patterns: X-partition blue, the most outer region green; Y-partition yellow

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 (5) 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 these parameters given, the program produces automatically the checkered-like patterns at once. All the experiments successfully demonstrated that the bi-partitioning algorithm is very robust.

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; 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

1571
1572
1573
1574

6. CONCLUSION

1575 The method of this study demonstrated its ability to generate varieties of checkered patterns
1576 and their variations with various features including:
1577 1) Wide range of patterns can be produced.
1578 2) It can be applied to drawings of various closed line shapes as --- circles, rectangle,
1579 triangles, freehand drawing, etc.
1580 3) Width of line or curve can vary.
1581 4) It is automatic and very robust as demonstrated.
1582

1583 REFERENCES

- 1584 [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-72Jjb7PAhVFhJQKHZNJCJAQsAQIKA>
1585 0&bih=1060&tbm=isch&tbo=u&source=univ&sa=X&ved=0ahUKEwiG-72Jjb7PAhVFhJQKHZNJCJAQsAQIKA
1586 [2] <https://www.google.co.jp/search?q=checkered+pattern&espv=2&biw=1920&bih=1060&tbm=isch&tbo=u&source=univ&sa=X&ved=0ahUKEwj7vPCQ9b3PAhVCVZQKHT8KAUYQsAQIHg&dpr=1>, accessed 2016/10/3
1587 [3] T. Kurokawa, "Maze Construction by Using Characteristics of Eulerian Graphs," British Journal
1588 of Mathematics & Computer Science, Vol.9, Issue.6, pp.453-472, 2015.6, SCIENCE DOMAIN
1589 international, UK, ISSN: 2231-0851, <http://sciencedomain.org/issue/1147>
1590 [4] K. K. Meng, D. Fengming and T. E. Guan, Introduction to Graph Theory, World Scientific, 2007
1591 [5] J. Matousek and J. Nešetřil, Invitation to Discrete Mathematics, Second Edition, Oxford
1592 University Press, New York, 2009.
1593
1594
1595