AN ALGORITHM FOR MATCHING PERSPECTIVE VIEWS OF 3-D OBJECT BY USING COMPOSITE CIRCUITS

W. K. GU, J. Y. YANG AND T. S. HUANG

JANUARY 3, 1984

ACKNOWLEDGEMENT

This work was supported by the National Science Foundation Grant MCS 82-06926.

AN ALGORITHM FOR MATCHING TWO GRAPHS OF A 3-D MOVING OBJECT

W.K. Gu, J.Y. Yang, and T.S. Huang

Contents | Page

REFERENCES

[1] R. Y. Tsai, T. S. Huang, and W. L. Zhu, "Estimating three-dimentional motion parameters of a rigid planar patch," Part 1, IEEE Trans. ASSP, Dec. 1981; Part 2, Aug. 1982.

[2] B. L. Yen and T. S. Huang, "Determining three-dimentional motion and structure of a rigid body using the spherical projection," CVGIP, V-21, 21-32, Jan. 1983.

[3] J. Q. Fang and T. S. Huang, "Solving three dimentional small rotation motion equations," Proc. IEEE Conf. CVPR, June 19-23, 1983, Washington, D. C.

[4] S. Yam and L. S. Davis, "Image registration using generalized Hough Transform," Proc. IEEE Conf. PRIP, pp. 526-533, Aug. 1981, Dallas, TX.

[5] S. Ranade and A. Rosenfeld, "Point pattern matching by relaxation," PR-12, pp. 269-272, 1980.

[6] J. Q. Fang and T. S. Huang, "A corner finding algorithm for image analysis and registration," Proc. AAAI Natinal Conf., Aug. 18-20, 1982, Pittsburgh, PA.

[7] D. J. Burr, "Elastic matching of line drawing," Proc. of 5th ICPR, pp. 223-228, Dec. 1980, Miami, Florida.

[8] R. Bajcsy and C. Broit, "Matching of deformed images," Proc. of 6th IJCPR, pp. 351-353.

[9] J. J. Hwang and E. L. Hall, "Matching of featured objects using relatinal tables from stereo images," CGIP., V-20, 1982, pp. 22-24.

Results of experiment 6.

Number of circuits of G6: 230.

Number of circuit categories of G6: 192.

Number of circuits of $G'6$: 221.

Number of circuit categories of $G'6$: 191.

Number of category pairs between two graphs: 51.

Number of main matching cycles: 6.

CPU time: 20 seconds.

Matching result:

V of G6 (coresponding to)	V' of $G'6$	*
1	10	
2	5	
3	1	
6	15	
7	4	
8	2	
9	17	
11	3	
12	7	
13	16	
14	14	
15	9	
16	8	T--j
17	11	
18	19	
19	6	
20	18	

CL of G6

V	X	Y
1	140	270
2	147	214
3	182	201
4	191	143
5	226	134
6	225	170
7	261	160
8	220	229
9	251	216
10	261	226
11	264	200
12	301	185
13	303	227
14	337	213
15	218	271
16	256	258
17	210	329
18	246	313
19	282	346
20	316	329

VCL of G6

V	Connected with			
1	2	17		
2	1	3	15	
3	2	4	8	
4	3	5	6	7
5	4	7		
6	4	7	8	
7	5	6	9	4
8	3	6	9	
9	7	8	10	
10	9	11	16	
11	10	12	13	
12	11	14		
13	11	14	19	
14	12	13	20	
15	2	16	17	
16	10	15	18	
17	1	15	18	
18	16	17	19	
19	13	18	20	
20	14	19		

Experiment 6. Input data of G6

Experiment 6.

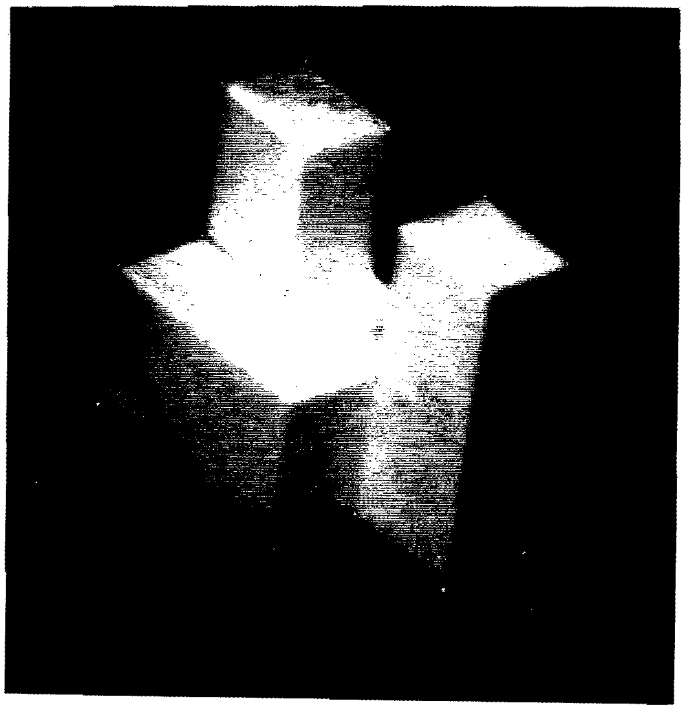


Image 6

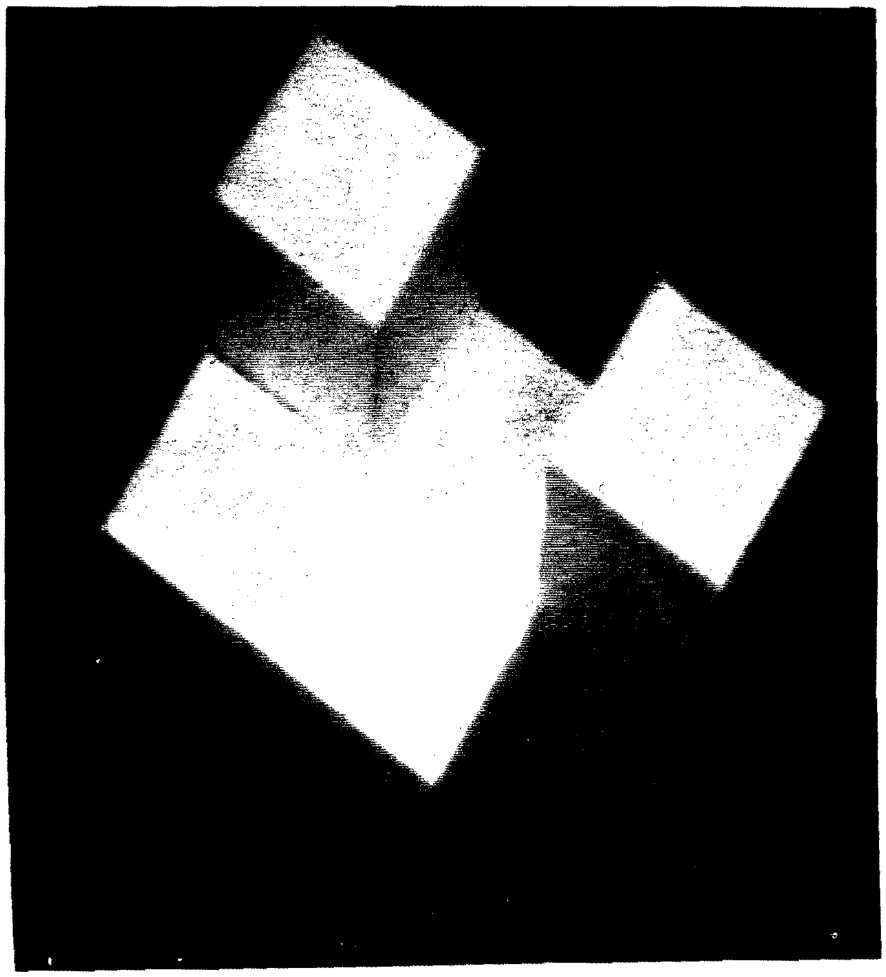


Image 6'

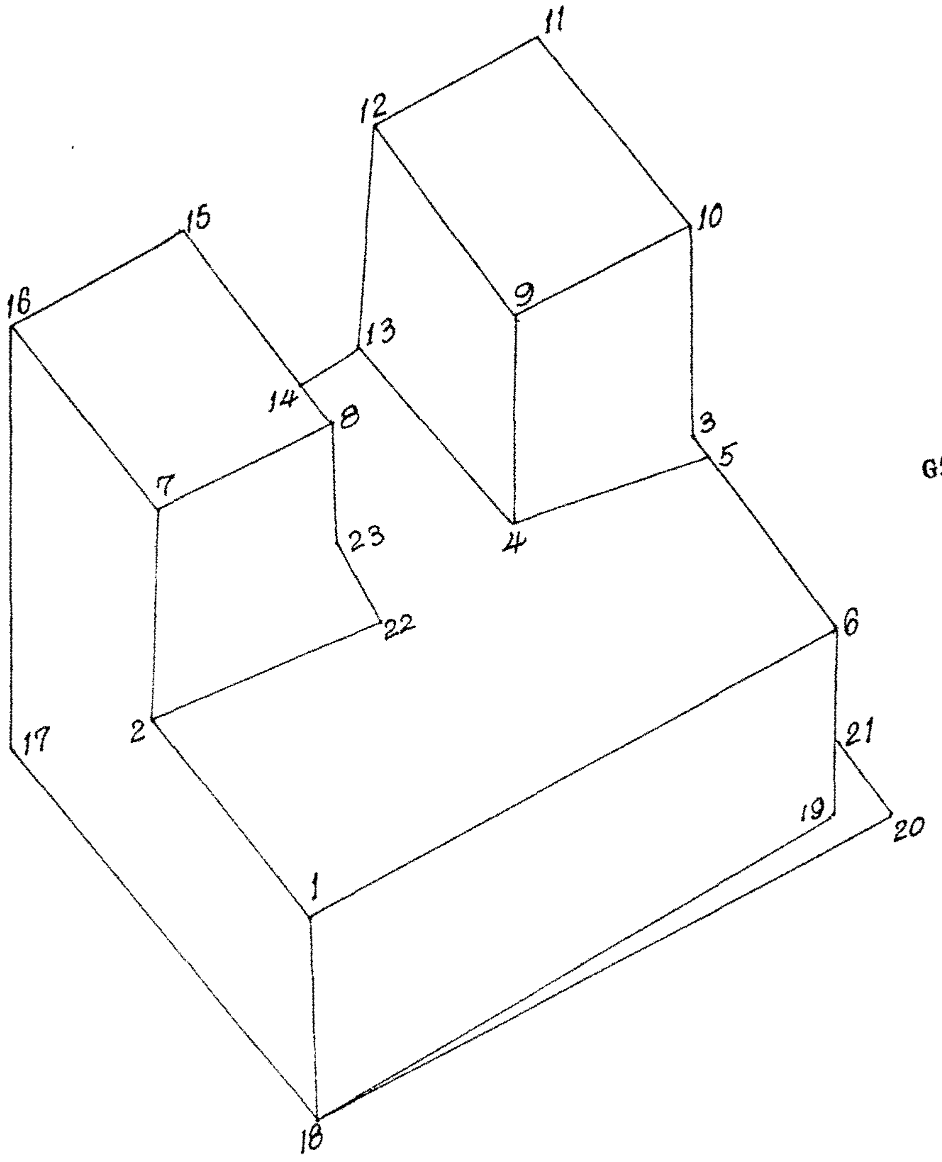
CL of $G'5$

V'	X	Y
1	338	302
2	350	333
3	393	192
4	382	157
5	324	195
6	340	149
7	314	163
8	327	116
9	286	109
10	272	154
11	282	191
12	277	208
13	257	205
14	299	213
15	241	251
16	282	261
17	312	247
18	295	294
19	266	319
20	293	251
21	372	263
22	342	348
23	312	340
24	302	353
25	263	340
26	185	282
27	165	253
28	221	261
29	215	230
30	160	202
31	165	126
32	279	200
33	338	192
34	335	208
35	364	215

VCL of $G'5$

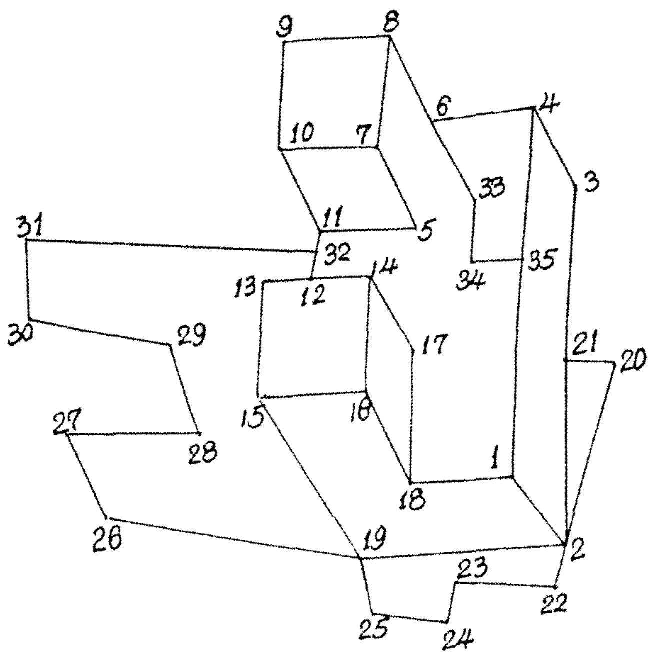
V'	Connected with				
1	2	35	18		
2	1	21	19	20	22
3	21	4			
4	35	3	6		
5	7	11			
6	8	4	33		
7	5	8	10		
8	6	7	9		
9	8	10			
10	7	9	11		
11	5	10	32		
12	32	13	14		
13	12	15			
14	12	16	17		
15	13	16	19		
16	14	15	18		
17	14	18			
18	1	16	17		
19	2	15	25	26	
20	2	21			
21	2	3	20		
22	2	23			
23	22	24			
24	23	25			
25	24	19			
26	19	27			
27	28	26			
28	27	29			
29	28	30			
30	29	31			
31	32	30			
32	11	12	31		
33	6	34			
34	33	35			
35	1	4	34		

Experiment 5. Input data o $G'5$



11
12
10
15
16
9
13
14
8
3
5
7
23
4
22
6
17
2
21
19
20
1
18

G5



9
8
6
4
10
7
3
33
11
5
31
32
35
34
13
14
12
17
30
29
21
20
27
15
16
28
1
18
26
19
2
23
22
25
24

G'5

Results of experiment 4--3.

Number of circuits of G4--3: 12.

Number of circuit categories of G4--3: 7.

Number of circuits of G'4--1: 747.

Number of circuit categories of G'4--1: 560.

Number of category pairs between two graphs: 6.

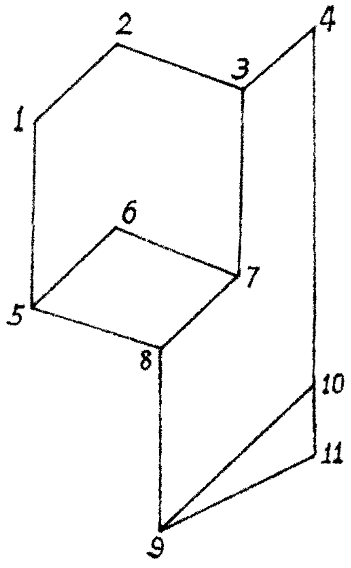
Number of main matching cycles: 3.

CPU time: 18 seconds.

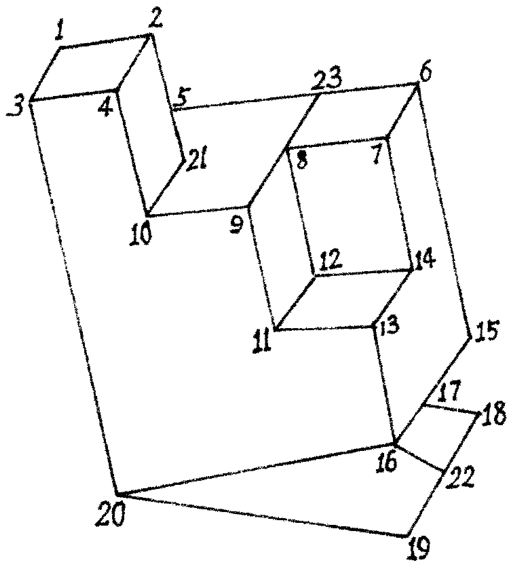
Matching result:

V of G4--3 (corresponding to)	V' of G'4--1
1	9
2	8
3	7
4	6
5	11
6	12
7	14
8	13
9	16
10	15

Experiment 4--3.



G4--3



G'4--1

CL of G4--2

V	X	Y
1	182	143
2	265	119
3	234	152
4	245	225
5	275	191
6	280	230
7	301	234
8	310	307
9	341	273
10	366	319
11	397	282
12	376	389
13	208	359
14	432	320
15	438	357

VCL of G4--2

V	Connected with			
1	3	13		
2	3	5		
3	1	2	4	
4	3	5	6	
5	2	4	6	
6	4	5	7	
7	6	8		
8	7	9	10	
9	8	11		
10	8	11	12	
11	9	10		
12	10	14	15	13
13	1	12		
14	12	15		
15	12	14		

Experiment 4--2. Input data of G4--2

<u>Results</u> <u>of</u> <u>experiment</u> <u>4</u>--<u>1</u>.

Number of circuits of G4--1: 55.

Number of circuit categories of G4--1: 28.

Number of circuits of G'4--1: 747.

Number of circuit categories of G'4--1: 560.

Number of category pairs between two graphs: 24.

Number of main matching cycles: 3.

CPU time: 25 seconds.

Matching result:

V of G4--1 (corresponding to)	V' of G'4--1 *
1	3
2	2
3	4
4	10
5	21
7	9
8	8
9	7
10	6
11	11
12	12
13	14
14	13
15	20
16	16
17	15

CL of G4--1

V	X	Y
1	182	143
2	265	119
3	234	152
4	245	225
5	275	191
6	280	230
7	301	234
8	332	198
9	390	210
10	416	174
11	310	307
12	341	273
13	397	282
14	366	319
15	208	359
16	376	389
17	432	320
18	438	357

VCL of G4--1

V	Connected with			
1	3	15		
2	3	5		
3	1	2	4	
4	5	3	6	
5	2	4	6	
6	5	4	7	
7	8	6	11	
8	9	7		
9	10	8	13	
10	17	9		
11	12	7	14	
12	11	13		
13	9	12	14	
14	13	11	16	
15	1	16		
16	17	14	15	18
17	10	16	18	
18	17	16		

Experiment 4--1. Input data of G4--1

<u>Experiment 4--1.</u>

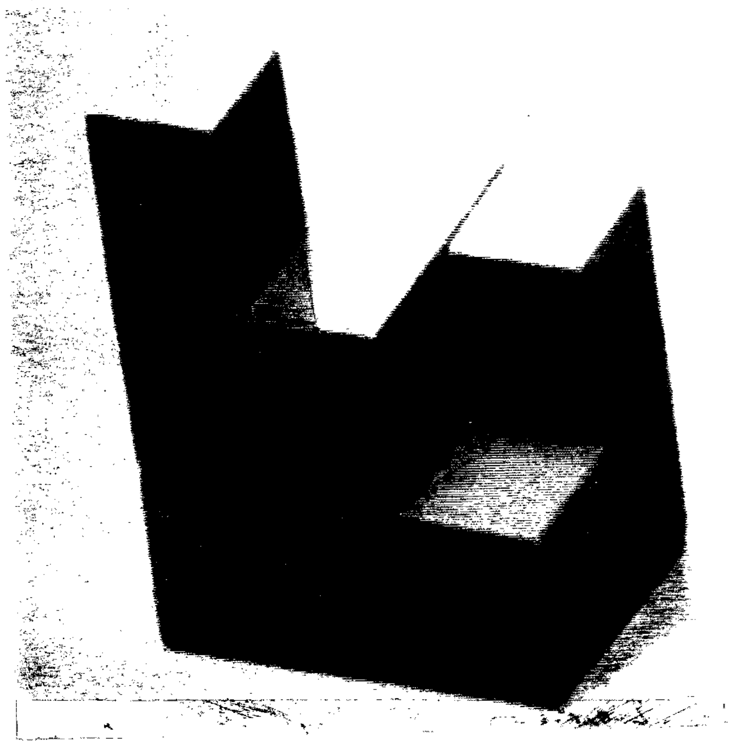


Image 4

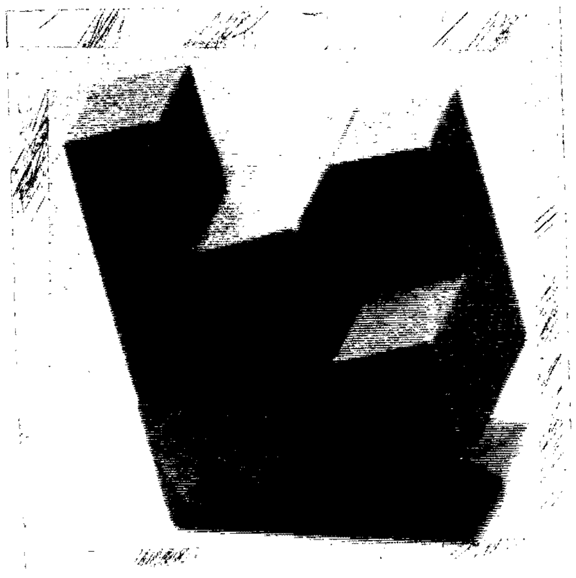


Image 4'

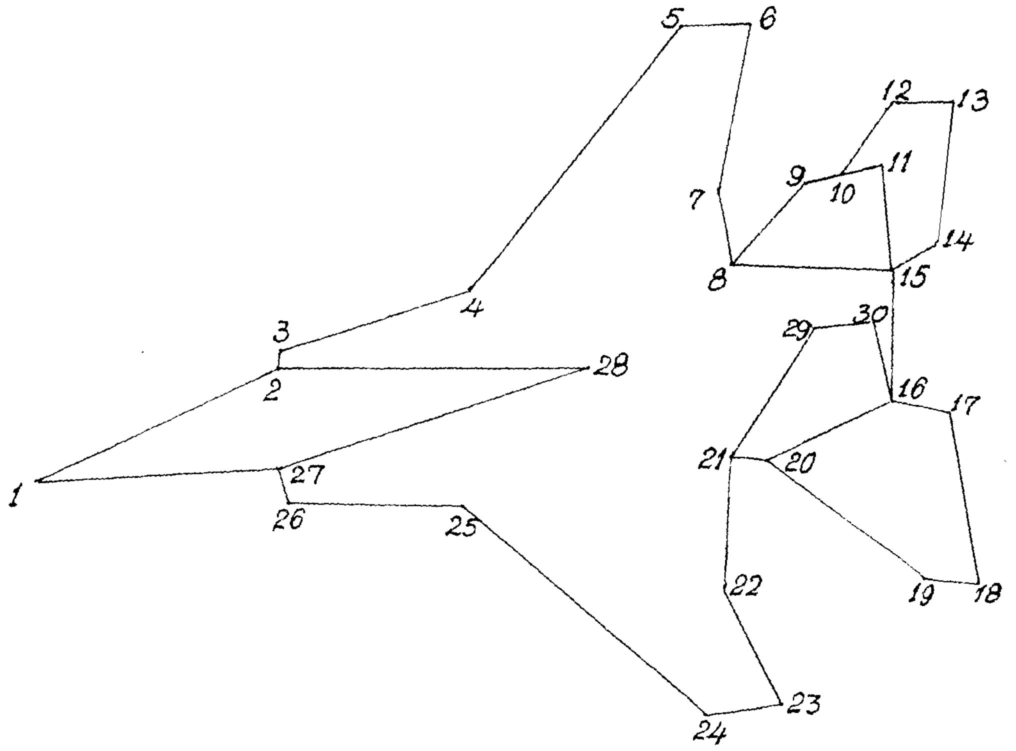
CL of $G'3$

V'	X	Y
1	304	255
2	351	263
3	321	316
4	330	322
5	283	203
6	248	255
7	346	391
8	361	403
9	417	264
10	428	278
11	206	207
12	213	198
13	223	176
14	230	169
15	377	189
16	393	205
17	364	241
18	305	345
19	382	275
20	371	266
21	392	283
22	389	303
23	405	309
24	377	354
25	368	341
26	352	303
27	369	318
28	271	373
29	293	386
30	154	132

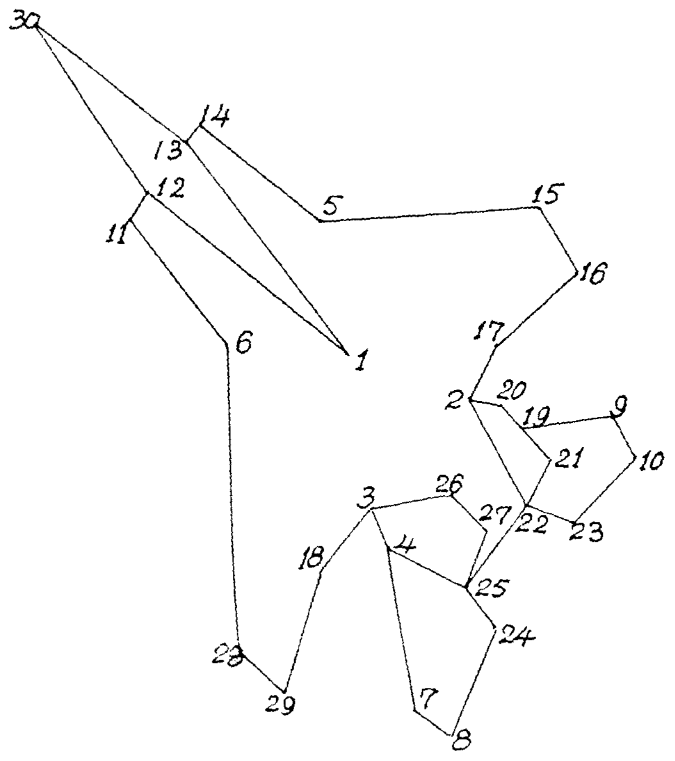
VCL of $G'3$

V'	Connected with			
1	12	13		
2	17	20	22	
3	4	18	26	
4	3	7	25	
5	14	15		
6	11	28		
7	4	8		
8	7	24		
9	10	19		
10	9	23		
11	6	12		
12	1	11	30	
13	1	14	30	
14	5	13		
15	5	16		
16	15	17		
17	2	16		
18	3	29		
19	9	20	21	
20	2	19		
21	19	22		
22	2	21	23	25
23	10	22		
24	25	8		
25	22	27	4	24
26	3	27		
27	26	25		
28	6	29		
29	18	28		
30	13	12		

Experiment 3. Input data of $G'3$


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

G3


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

G'3

<u>Results of experiment 2</u>

Number of circuits of G2: 162.

Number of circuit categories of G2: 124.

Number of circuits of $G'2$: 39.

Number of circuit categories of $G'2$: 34.

Number of category pairs between two graphs: 34.

Number of main matching cycles: 3.

CPU time: 7.1 seconds.

Matching result:

V of G2	(corresponding to)	V' of $G'2$	*
1		13	
2		14	
3		15	
4		16	
5		17	
6		1	
7		12	
8		2	
9		3	
10		10	
11		9	T--j
12		4	T--j
13		5	
14		6	
15		8	
16		7	
17		11	

CL of G2

V	X	Y
1	138	372
2	179	405
3	292	327
4	337	359
5	393	321
6	126	297
7	167	327
8	184	260
9	173	180
10	215	209
11	271	168
12	266	121
13	263	90
14	320	52
15	327	119
16	366	76
17	226	289
18	217	225
19	313	150

VCL of G2

V	Connected with			
1	2	6		
2	1	7	3	
3	4	11	2	
4	5	15	3	
5	4	16		
6	1	7	8	
7	2	6	17	
8	6	9	18	
9	8	10	12	
10	9	11	18	
11	10	3	19	12
12	9	13	11	19
13	12	14	15	
14	13	16		
15	4	16	13	
16	15	14	5	
17	7	18		
18	8	10	17	
19	11	12		

Experiment 2. Input data of G2

Experiment 2.

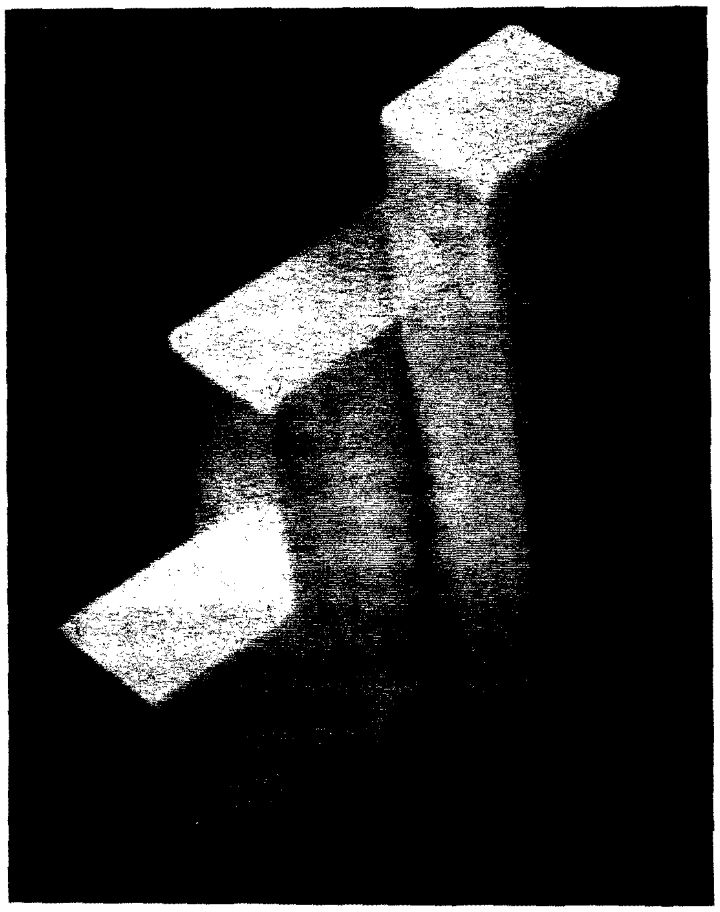


Image 2

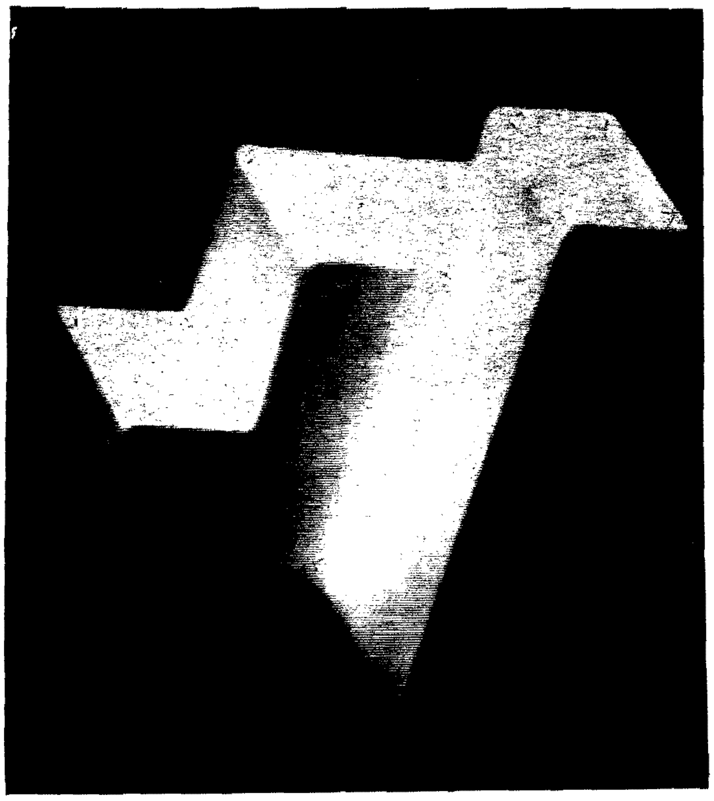


Image 2'

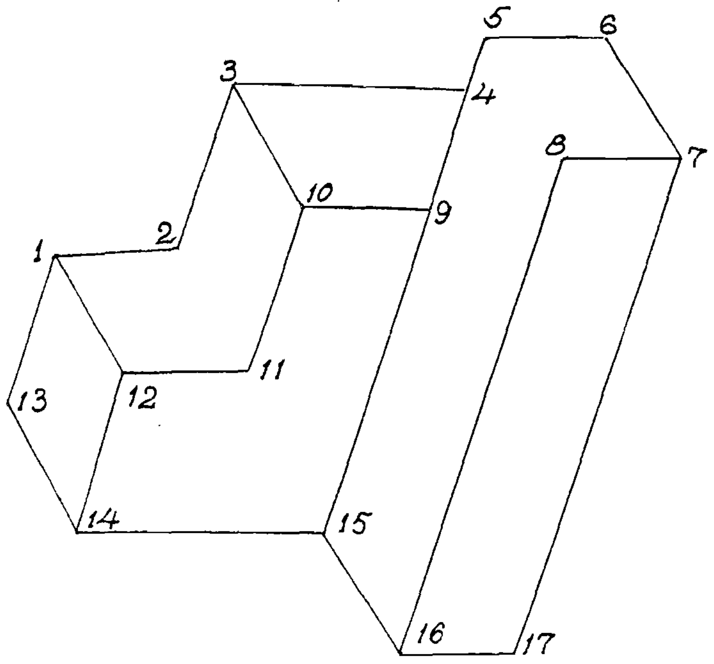
CL of $G'1$

V'	X	Y
1	206	265
2	342	228
3	410	211
4	403	264
5	196	323
6	317	156
7	327	105
8	257	124
9	249	177
10	223	98
11	232	48
12	165	66
13	154	119
14	222	393
15	428	337
16	435	285
17	364	223

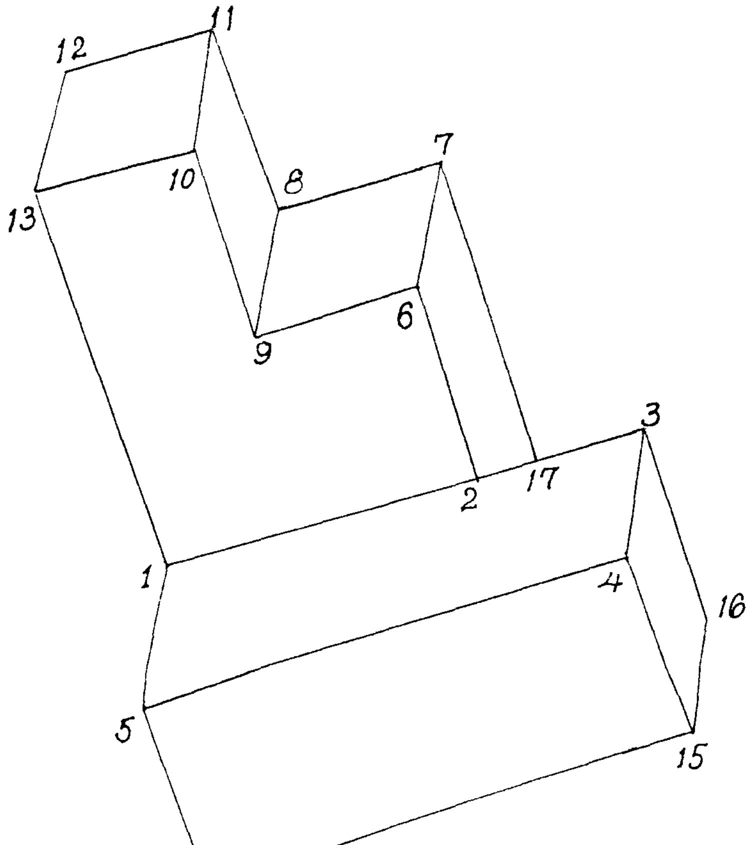
VCL of $G'1$

V'	Connected with		
1	2	5	13
2	1	6	17
3	4	16	17
4	3	5	15
5	1	4	14
6	2	7	9
7	6	8	17
8	7	9	11
9	6	8	10
10	9	11	13
11	8	10	12
12	11	13	
13	1	10	12
14	5	15	
15	4	14	16
16	3	15	
17	2	3	7

Experiment 1. Input data of $G'1$


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

G1


11
12
10
13
8
7
9
6
3
2
17
1
4
16
5
15
14

G'1

the edge 11--16 which contains T--joint vertex 10 in G6 does not correspond to the edge 3--7 which contains T--joint vertex 20 in $G'6$. This is caused by the perspective projection. The results of experiment 6 illustrate that the algorithm can reject such T--joints. And notice that the number of the main matching cycles in this experiment is six which is the largest number within the six experiments.

We can see, from above experimental results, that our approach of matching two consecutive graphs of a moving object is quite powerful. It works well in the presence of shape distortions, scaling, rotation and dissimilarities as well as shadows, and T--joint vertices. However, as it is, our algorithm can be used only for objects whose faces are or can be closely approximated by planar polygons. We are currently working on modification of our algorithm which can be applied to other types of objects.

graph, see point A in Fig. 6.2, or it can be an intersection point between a shadow line and an edge of an object, see point B in Fig. 6.2, or it can be an intersection point between an illusory line and an edge of an object, see point C in Fig. 6.2. The T--joint pairs are denoted by the symbol "T--j" in the matching results. In experiment 1, vertex 4 and vertex 9 in G1 and vertex 2 vertex 17 in $G'1$ are T--joints.

Experiment 2

In G2 and G'2 (see pp. 91), besides three-dimensional motion, scaling and dissimilarites, there are two vertices 19 and 18 in G2 which are

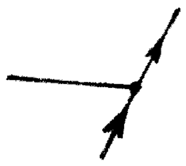
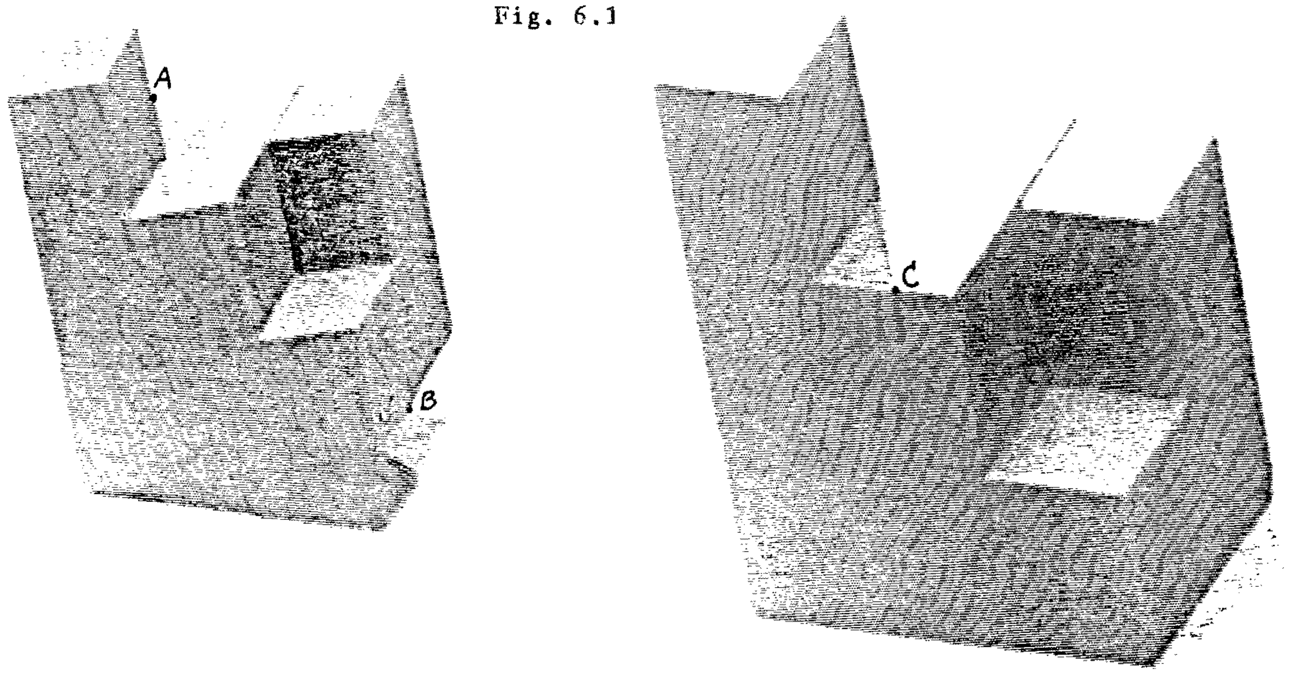


Fig. 6.1



(a) (b)

Fig. 6.2

a special counter, say C--counter, has to be used.

After the MMC has been stoped, the DL is checked. If the DL and its buffer have the same content and all the relative clustering degree RCD in the third column are all greater than a certain constant, i.e., before matching decision process in each row of MT the value is concentrated sharply in one accumulator, then the DL is printed out as the final matching result. If the DL and its buffer have different contents, i.e., after N_c cycles the result of MMC does not converge, then the matching process is considered to have failed. And if the result of MMS has converged but not all the RCD in the third column are greater than a certain constant (it means that the value in each row cannot be sharply concentrated in one accumulator), then the matching process is considered to have failed too. In these cases, a string of character, "The matching process is failed", is printed out to point out the failure of the matching process.

The Structure of the Algorithm

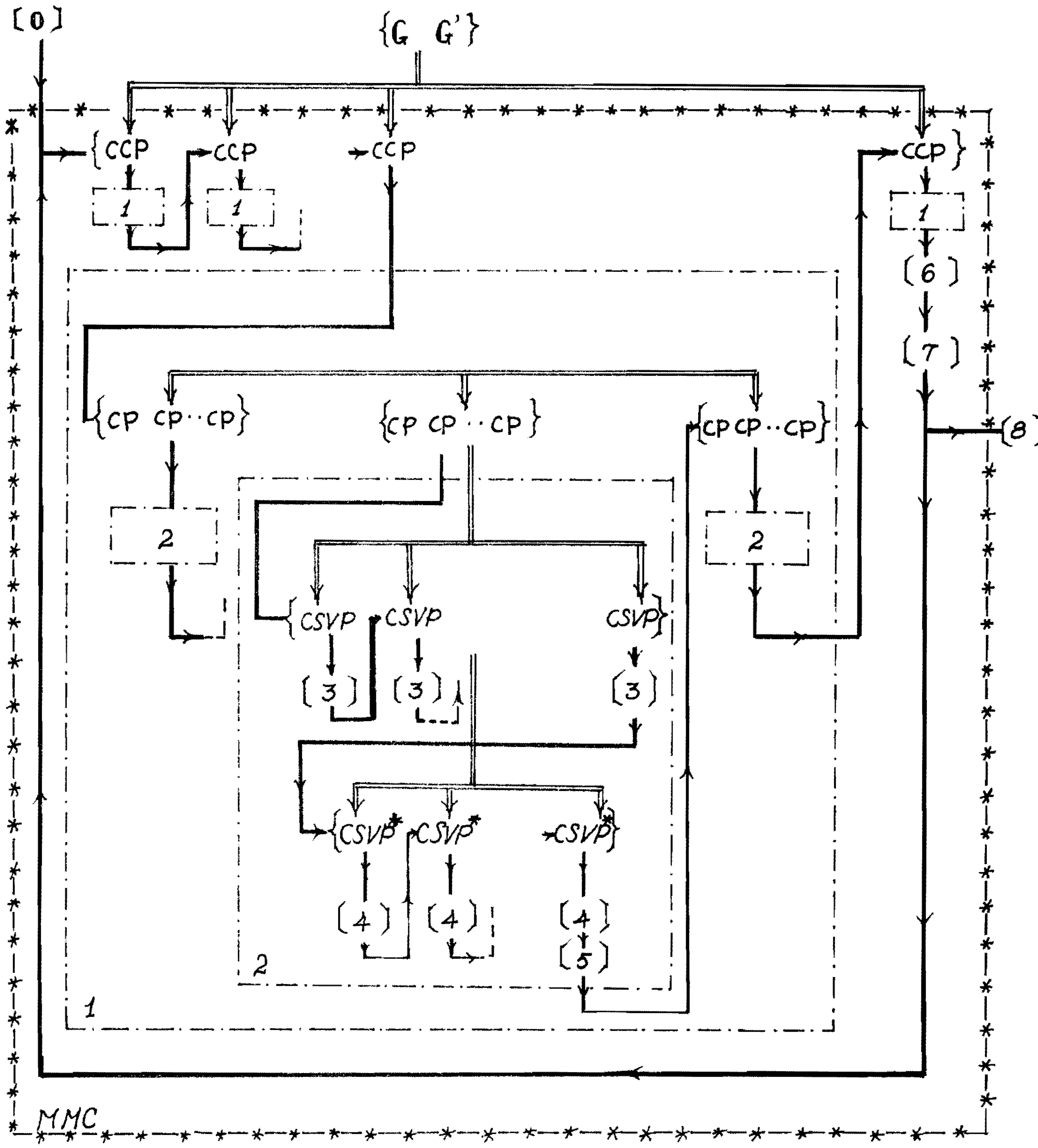


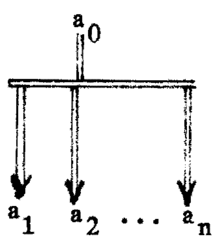
Fig 5.7.2

	all the circuits in G and G' have the same RLCC code;
{CP CP...CP}	the set of circuit pairs in a CCP,
{CSVP CSVP}	the set of CSVP belongs to above set of circuit pairs;
$\{CSVP^* \ CSVP^*\}$	the set of consistency CSVP;

as soon as the consistency checking period for a circuit has ended , the accumulating period begins. In the accumulating period, for each vertex pair in all the consistent CSVPs assotiated with the circuit in G, an increment 1/r is added to an accumulator of MT, which is assotiated wih that verex pair. After the accumulating period the r--counter is set to zero to prepare for a nex circuit in G.

5.7 Main matching cycle (MMC)

The main matching cycle (MMC) is the main body of the matching process (see Fig. 5.7.1). Fig. 5.7.1 illustrats the entire algorithmic scheme. Fig. 5.7.2 illustrates the algorithmic structure of the MMC. the symbols used in Fig. 5.7.2 are explained as follows:

symbols:	notes:
	each of the element a_1, a_2, ..., a_n which are belong to a_0(can be a set or a element);
$\longrightarrow$	the direction of the algorithmic flow;
	block scheme of MMC;

just the corresponding vertex pair between the two graphs G and G'. However, if the second column of the row in DL is zero, then it means that the vertex of G indicated by the number of the first column has no corresponding vertex in graph G'.

5.5 Consistency among the real cyclic sets of vertex pairs (CSVP)

Suppose there are several CSVPs. We say these CSVPs are consistent if the vertex pairs in these CSVPs are all compatible, i.e., no any two of the vertex pairs in these CSVPs are conflicting each other. It is true that the real cyclic sets of vertex pairs between two graphs are consistant. We can use the consistency among the real CSVPs to distinguish the real CSVPs from the false ones. The main principle is that the vertex pairs represented by DL(see Exp. 5.4.1) is supposed to be the real corresponding vertex pairs between two graphs. And the matching process except laval 1 (see Sec. 5.2) then is repeated over all the CCP again (it will be defined as main matching cycle later). During the matching process each CSVP has to be checked to make sure whether it is compatible with DL or not. If the answer is yes, then the accumulators of MT associated with the vertex pairs of the CSVP will be incremented. Otherwise the CSVP will be discarded.

Specifically, the consistency check for each CSVP is as follows:

Comparing each vertex pair of the CSVP with that represented in DL. If any vertex pair in the CSVP is found to be inconsistant with DL then the CSVP will be discarded and skipped, the matching process then begins to search the next CSVP. A vertex pair is considered inconsistant with DL if the vertex label of G' in the second column of DL is not zero and is different vertex pairs between the two graphs because of the clustering property of them.

After searching over all the circuit category pairs (CCPs) of the two graphs, a process called decision process will be carried out. A list, which is called decision list (DL), will be used to record the decision results, see List 5.4.2. The first column of the DL is filled with the vertex labels of G, the second column of DL is filled with the vertex labels of G', which will be the corresponding vertices of the vertices represented by the first column, and the third column of DL is filled with the parameters called relative clustering degree, see Def. 5.4.1.

Def. 5.4.1 RCD: Relative Clustering Degree

It is the ratio of the largest value to the second largest value in a row of MT. Obviously, the parameter RCD reflects the clustering degree of a corresponding vertex pair.

The decision process is as follows:

1. For each row of MT and the same row of DL:

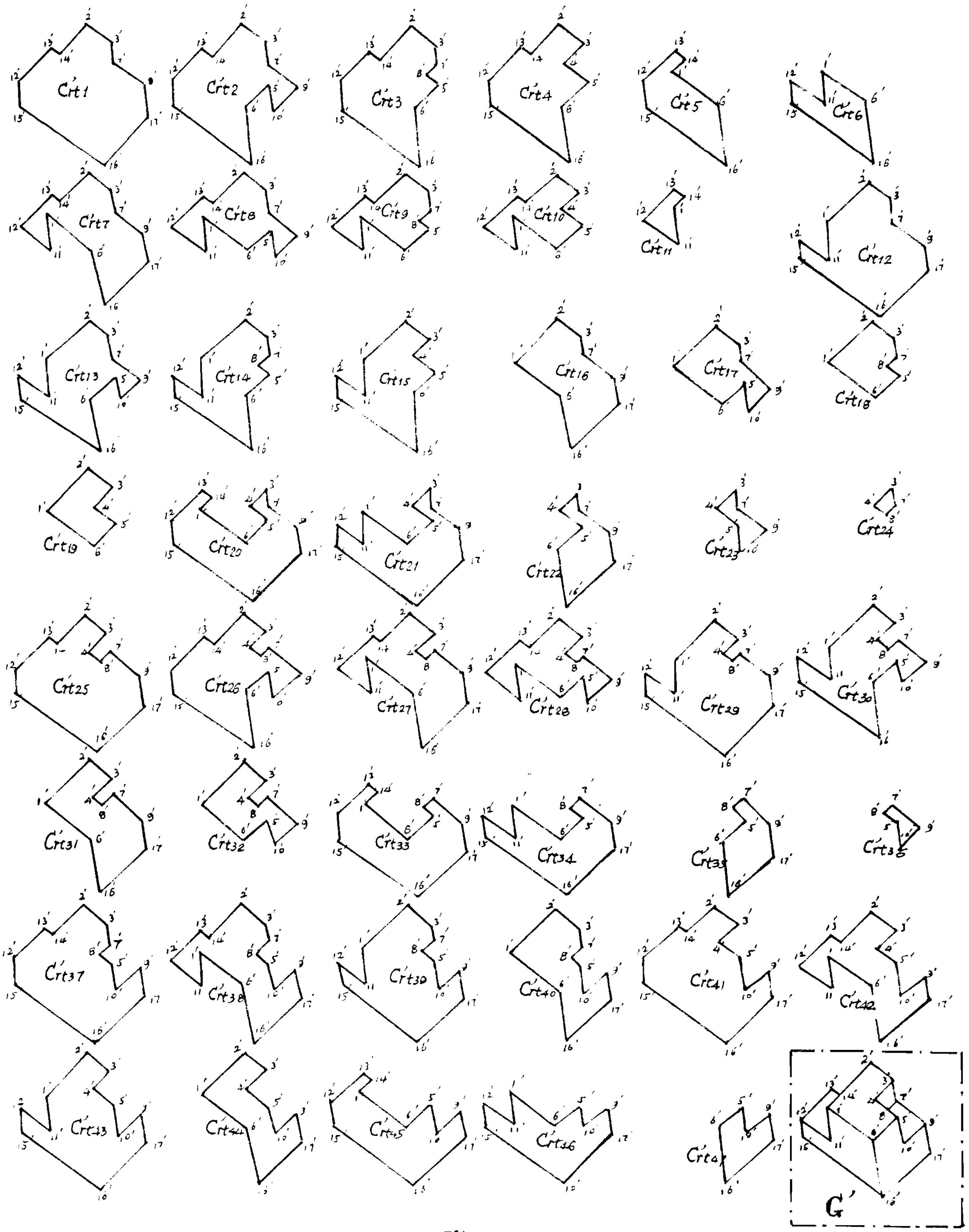
   a. Selecting an accumulator whose value is the largest one in the row of MT, and filling the second column of DL with the vetex labels of G' assotiated with the selected accumulator;

   b. Finding the second largest value in the row of MT, and filling the third column of DL with the RCD, see Def. 5.4.1;

   c. Setting all the accumulators to zero except the selected one in a.

2. For each column of MT:

   Selecting an accumulator whose value is the largest one in the column, and setting all the accumulators to zero except the selected one;



Crt1
Crt2
Crt3
Crt4
Crt5
Crt6
Crt7
Crt8
Crt9
Crt10
Crt11
Crt12
Crt13
Crt14
Crt15
Crt16
Crt17
Crt18
Crt19
Crt20
Crt21
Crt22
Crt23
Crt24
Crt25
Crt26
Crt27
Crt28
Crt29
Crt30
Crt31
Crt32
Crt33
Crt34
Crt35
Crt36
Crt37
Crt38
Crt39
Crt40
Crt41
Crt42
Crt43
Crt44
Crt45
Crt46
Crt47
G′

Fig. 5.3.2

2--3'	37
3--4'	21
4--5'	35
5--6'	35
6--1'	33
7--14'	21
8--13'	21
9--12'	32
10--11'	21
11--10'	21
12--9'	35
13--7'	33
14--8'	20
15--17'	25
16--16'	35
17--15'	24

If in a circuit pair Crt--Crt$'$,

KP_1,KP_2,......, KP_k are the k key points in circuit Crt, and SPPs KP'_2,......, KP'_k are the k key points in circuit Crt$'$, then the k point are KP_1--KP'_1, Kp_2--KP'_1,..SPP_k, repsectively, for the circuit pair. In example 5.2.1, point pairs, V_3--V'_{12}, V_6--V'_{12}, V_9--V'_{12}, V_{12}--V'_{12} are the 4 SPPs for the circuit pairs.

From the above two examples and Def. 5.2.1 as well as Def. 5.2.2 we see that all the vertex pairs in a CSVP are the potential corresponding vertex pairs between the two graphs. The reasons are as follows:

1. Each CSVP cames from a circuit pair, and a circuit pair is itself a potential corresponding circuit pair as we have mentioned before;

2. Each CSVP is synchronized by a SPP, i.e., the first vertex pair in a CSVP, and a SPP is composed of two key point which represent the common structure characteristic points between the two circuits.

From the above analysis, we can say that the matching process can be devided into three levels:

level 1:

searching for the circuit category pairs (CCP) between two graphs according to the associated RLCC codes;

level 2:

searching for the circuit pairs (CP) within each CCP;

level 3:

saerching for the cyclic sets of vertex pairs (CSVP) within each CP.

The CSVPs are the basic cell for the entire matching process,and we should pay more attantion to it.

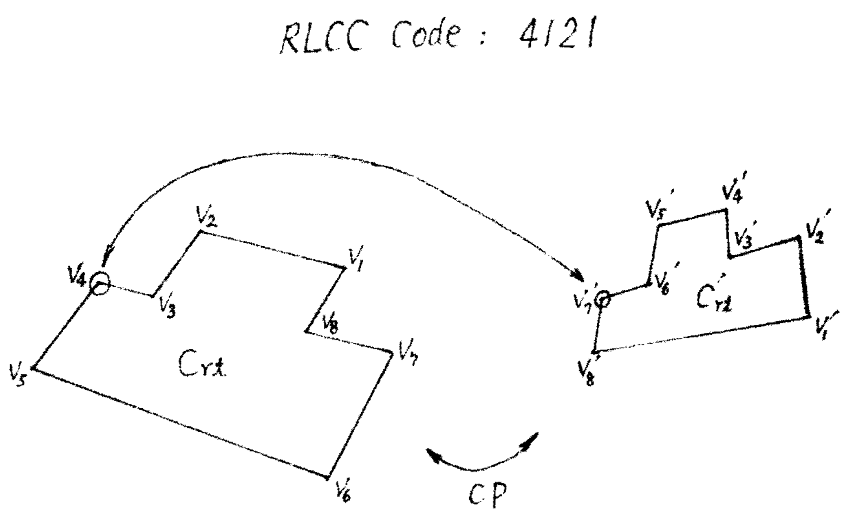


Fig. 5.2.1

V_4--V_7', V_5--V_8', V_6--V_1', V_7--V_2', V_8--V_3', V_1--V_4', V_2--V_5', V_3--V_6'.

Example 5.2.2:

Fig. 5.2.2 illustrates a circuit pair: Crt--Crt', where

Crt:$V_3V_4V_5V_6V_7V_8V_9V_{10}V_{11}V_{12}V_1V_2$,

Crt':$V_{10}'V_{11}'V_{12}'V_1'V_2'V_3'V_4'V_5'V_6'V_7'V_8'V_9'$.

The key points of Crt are $KP_1=V_3$, $KP_2=V_6$, $KP_3=V_9$, $KP_4=V_{12}$, and the key points of Crt' are $KP_1'=V_{12}'$, $KP_2'=V_3'$, $KP_3'=V_6'$, $KP_4'=V_9'$. The vertex pair in the circuit pair are divided into following four groups:

1. V_3--V_{12}', V_4--V_1' , V_5--V_2' , V_6--V_3', V_7--V_4', V_8--V_5', V_9--V_6', V_{10}--V_7', V_{11}--V_8', V_{12}--V_9', V_1--V_{10}', V_2--V_{11}';
2. V_6--V_{12}', V_7--V_1', V_8--V_2', V_9--V_3', V_{10}--V_4', V_{11}--V_5', V_{12}--V_6', V_1--V_7', V_2--V_8', V_3--V_9', V_4--V_{10}' , V_5--V_{11}';
3. V_9--V_{12}', V_{10}--V_1', V_{11}--V_2', V_{12}--V_3', V_1--V_4', V_2--V_5', V_3--V_6', V_4--V_7', V_5--V_8', V_6--V_9', V_7--V_{10}', V_8--V_{11}';

5. MATCHING PROCESS

5.1 Category pair and circuit pair

Before the matching process, the two consecutive graphs of a moving object, say G and G', have been decomposed into two sets of composite circuits individually, then all the circuits in each set have been encoded and classified into related circuit categories according to the RLCC code. The two circuit category lists, CCL and CCL', formed by the classification process are the main data source for the matching process.

Def. 5.1.1 Circuit Category Pair (CCP):

The two circuit categories which lie in the two graphs individually and are assotiated with a same RLCC code.

Def. 5.1.2 Circuit Pair (CP):

The two circuits which lie in the two graphs individually and are assotiated with a same RLCC code.

A circuit pair is a potential corresponding circuit pair, i.e., the two circuits in a circuit pair will possiblly be corresponding to each other because their RLCC codes are the same. Therefore, the corresponding vertex pairs between the two graphs should be searched within all the circuit pairs.

Accorrding to Def. 5.1.1 and Def. 5.1.2, in a circuit category pair (CCP) there will be m*n circuit pairs altogether, where m is the number of circuits in one category of the CCP and n is the number of circuits in the other , see Fig. 5.1.1. For clarity, we use the symbols CC, C, V to represent a circuit cayegory, a circuit, and a vertex in graph G, and use the CC', C', V' to

4.5 Classification process

We know that each circuit give rise to a unique normalized RLCC code. However, it does not mean that each normalized RLCC code must correspond to a unique circuit in a graph. For example, in Fig. 4.4.1, circuit $V_3V_2V_1V_5V_6V_8$, circuit $V_3V_2V_6V_8V_9V_4$, circuit $V_4V_3V_8V_9V_{11}V_7$ and circuit $V_8V_6V_5V_{10}V_{11}V_9$ have the same code 51, and circuit $V_2V_1V_5V_6$, circuit $V_3V_2V_6V_8$, circuit $V_4V_3V_8V_9$ and circuit $V_7V_4V_9V_{11}$ have the same code 4, etc..

Def. 4.5.1 Circuit category in a graph (CC):

It is a set of circuits with the same RLCC code in a graph (Sometimes we will call circuit category the category.)

After the encoding process, the classification process then classify the circuits of a graph into circuit categories according to their RLCC codes.

Def. 4.5.2 label of category:

It is used to represent circuit category of a graph with a serial number.

Def. 4.5.3 Circuit category list (CCL):

It is used to record the circuit categories and the parameters about each category.

In CCL, each row is assotiated with a circuit category. And in each row of CCL the first column is filled with the label of a category as the index of the category. The second and third columns are filled respectively with the parameters NSS and LSS assotiatd with the category. The fourth column is filled with the circuits in the category ,see List 4.5.1.

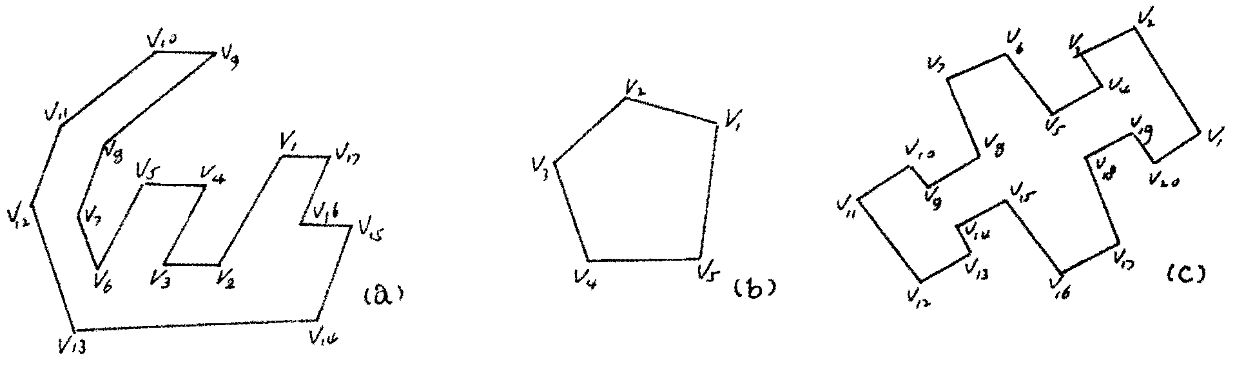


Fig. 4.3.1

The encoding process can be described in the following steps :

Step 1.

Searching for the first starting point of a convex vertex string (SP_1) along a circuit. If the circuit is a convex polygon like the circuit in Fig. 4.3.1(b), then the first vertex of the circuit is just SP_1. If the circuit is a concave polygon, then the first vertex met along the circuit, whose deflective angle DA just turns into a negative value at the present point from the positive value at the previous point, is the SP_1.

Step 2.

Encoding the circuit starting from SP_1. A counter is used to count up

Def. 4.3.4 Normalized code:

It is a code assoiated with a key point as the real encoding starting point.

For eliminating the ambiguity a RLCC code has to be a normalized code.

Def. 4.3.5 ONK:

It is the node ordering number of the first key point of a circuit in the circuit sequence list, see List 3.5.2.

Def. 4.3.6 LSS:

It is the length of the symmetrical section of the circuit, i.e., LSS=NVC/NK, where NVC is the number of vertices in the circuit and NK is the number of key points.

Example 4.3.2:

In Fig. 4.3.1,

Circuit (a) is $V_1V_2V_3V_4V_5V_6V_7V_8V_9V_{10}V_{11}V_{12}V_{13}V_{14}V_{15}V_{16}V_{17}$,

Circuit (b) is $V_1V_2V_3V_4V_5$,

Circuit (c) is

$V_1V_2V_3V_4V_5V_6V_7V_8V_9V_{10}V_{11}V_{12}V_{13}V_{14}V_{15}V_{16}V_{17}V_{18}V_{19}V_{20}$.

For circuit (a):

$SP_1=V_4$, related code: 237122, related CV: 185;

$SP_2=V_9$, related code: 712223, related CV: 273;

$SP_3=V_{17}$, related code: 222371, related CV: 141;

KP=9;

Normalized RLCC code: 712223;

NSS=1;

ONK=8;

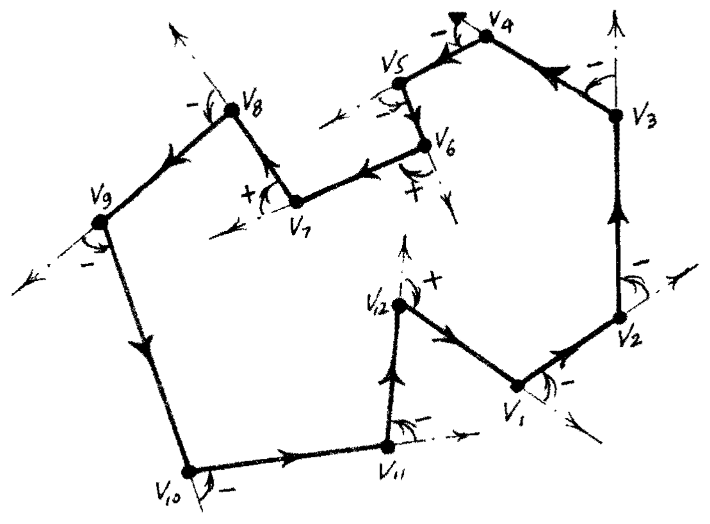


Fig. 4.1.1

$V_8V_9V_{10}V_{11}$. Vertex V_1 and vertex V_8 are the starting points of the two strings respectively.

Obviously, the starting point for encoding has to be a SP. If a circuit has more than one SP, for convinence, we distinguish the points with subscripts, for instance, SP_1, SP_2......SP_n. Each subscript number indicates the occuring order of that SP in the circuit sequence list of the circuit.

4.3 Eliminating ambiguity

If there are n convex vertex strings in a circuit, i.e., there are n SP points in the circuit, then there will be n codes assotiated with them. The ambiguity has to be overcome, otherwise the RLCC code can hardly be used in the matching process. If we have a mathod to select a point from the n SP points as a real starting point for the encoding properly, the ambiguity can be eliminated.

1. If the IV is not on the outer contour of the graph, see Fig. 3.9.3.6, then there is at least one circuit starting along the edge in the opposite direction of IE at IV which is counterclockwise. However, according to property 6 it is not contained in PTL.

2. If the IV is on the outer contour of the graph but the IE of IV is not coincided with the outer contour, see Fig.3.9.3.7, then there is at least one circuit starting along the edge in the opposite direction of IE at IV which is counterclockwise. However, according to property 6 it is not contained in PTL.

3. If the initial OE of IV is not coincided with the outer contour, see Fig. 3.9.3.8, then each direction change at IV makes OE turn a step counterclockwise until OE turns into LAST EXIT, i.e., CT(IV)=0, then the process will reach the terminal state. So there is no chance making OE of IV turn into the EXIT which is on the outer contour. Therefore, the counterclockwise circuits starting from that EXIT will not be contained in PTL.

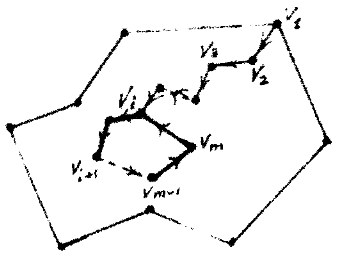
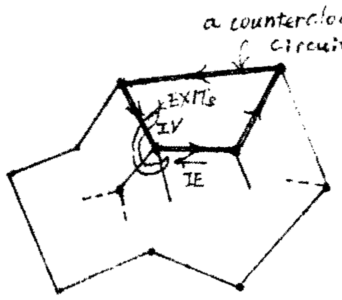
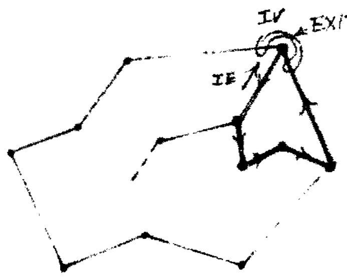


Fig. 3.9.3.5





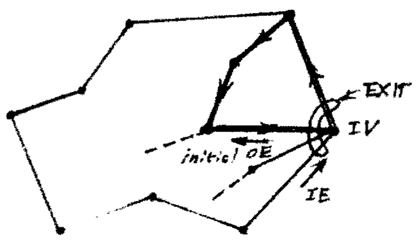


Fig. 3.9.3.6 Fig. 3.9.3.7 Fig. 3.9.3.8

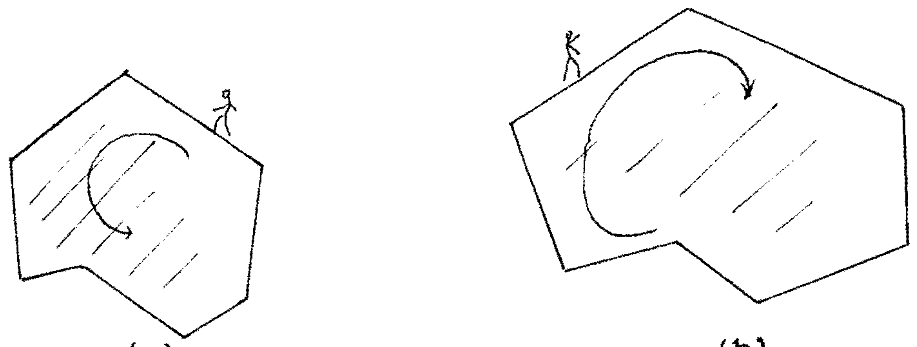


Fig. 3.9.3.3

outer contour of the graph counterclockwise, the outer contour itself and the inner plane of the counterclockwise circuit contain entire graph. If there is a circuit starting along an edge in the opposite direction of IE at IV, then the circuit cannot be a counterclockwise one, see Fig. 3.9.3.4. The reason is as below. Assume that the circuit starting along the edge in the opposite direction of IE at IV is a counterclockwise circuit. If one is marching along the edge in the opposite direction of IE at IV, i.e., he is marching along the outer contour clockwise, then the left part of him must be the outer part of the outer contour (see Def. 3.3.2), but according to Def. 3.9.3.2, the part above is the inner plane of that assumed counter-clockwise circuit. Therefore, it means that the inner plane of that assumed counterclockwise circuit is in the outer part of the outer contour of the graph. Clearly, this is absurd.

Proof of theorem 2:

1. According to the proposition 1, there is a path, say $V_1V_2......V_i$ (see Fig. 3.9.3.5), which starts from an EXIT of V_1 goes to V_i which is on an

circuit, then we will be able to split the graph into two subgraphs only through breaking the IE of IV. One of the subgraphs contains the vertex IV as well as all the vertices connecting to EXITs of IV through some paths, the other contains the circuit as well as the vertices connecting to the circuit through some paths but there is no path connecting them with EXIT of IV, see Fig. 3.9.3.1.

However, such splitting is impossible because according to hypothesis 5 there is a circuit which is the outer contour of the graph. From the condition of proposition 1, the IE of IV is a part of the contour, so that even if the IE has been broken, the two subgraphs of the graph are still connected together by the contour line, see Fig. 3.9.3.2.

Proposition 2:

The circuits connecting IV and starting along an edge in the opposite direction of IE at IV are all clockwise.

Before proving proposition 2, we provide the following definitions:

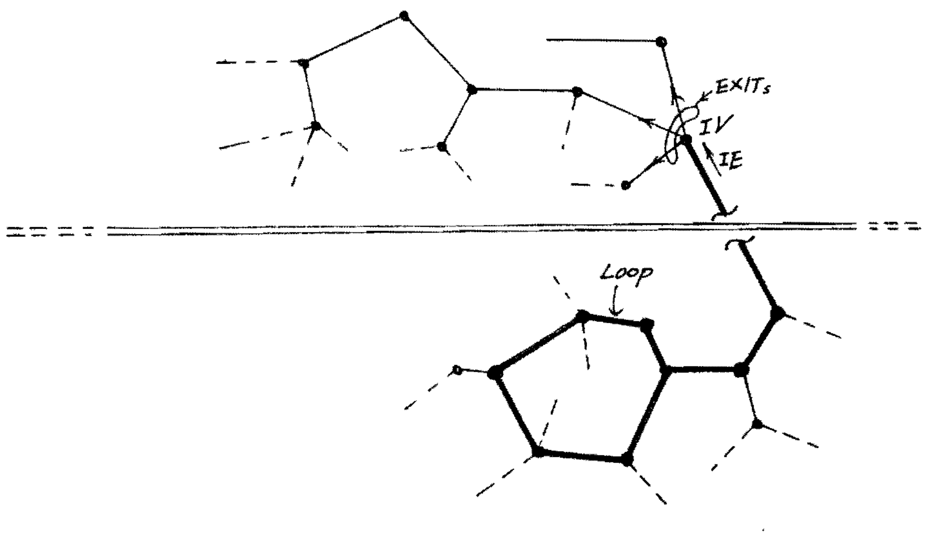


Fig. 3.9.3.1

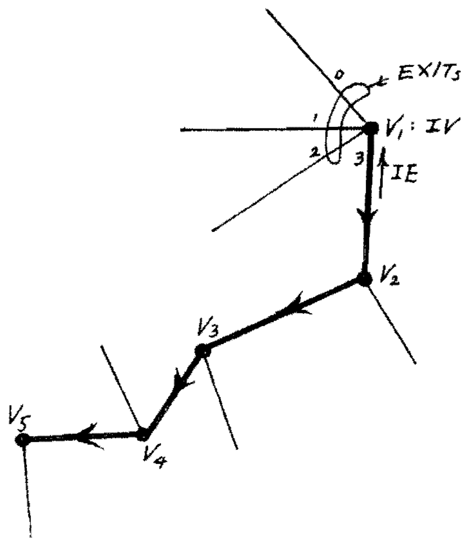


Fig. 3.9.2.4

EXIT, the related CT will be zero. If CT(V_1)=0 and there is any vertex to the right of PP_1 on a basic path line has not undergone all the changes, then turning point can not be at PP_1, so the path transfering will not affect the vertex at PP_1. Therefore, the OE at IV will remain on the LAST EXIT unchanged. If CT(V_1)=0 and all the OE of vertices to the right of PP_1 have undergone all the changes, i.e., all the CT values on the basic path line are zero, then the path transfering has reached terminal state and the basic path line has become the bottom line of the PTL. Therefore, the OE at IV can not be turned into the edge in the opposite direction of IE at IV. So that the path starts from the edge in the opposite direction of IE at IV will not be contained in PTL.

3.9.3 Two important theorems about the circuit decomposition algorithm

Theorem 1:

Property 3:

In successive lines of a basic path, all the elements to the left of any PP_i will remain unchanged until the OE of the point at PP_i and the OE of all the points to the right of PP_i have undergone all the changes.

Proof: If there is an OE of a point at PP_j, $j \geq i$, which has not undergone all the changes, then it means $CT(PP_j) > 0$. According to Def. 3.6.2, the turning point in the related line can not lie to the left of PP_i so the path transfering will not affect the elements to the left of PP_i.

Property 4:

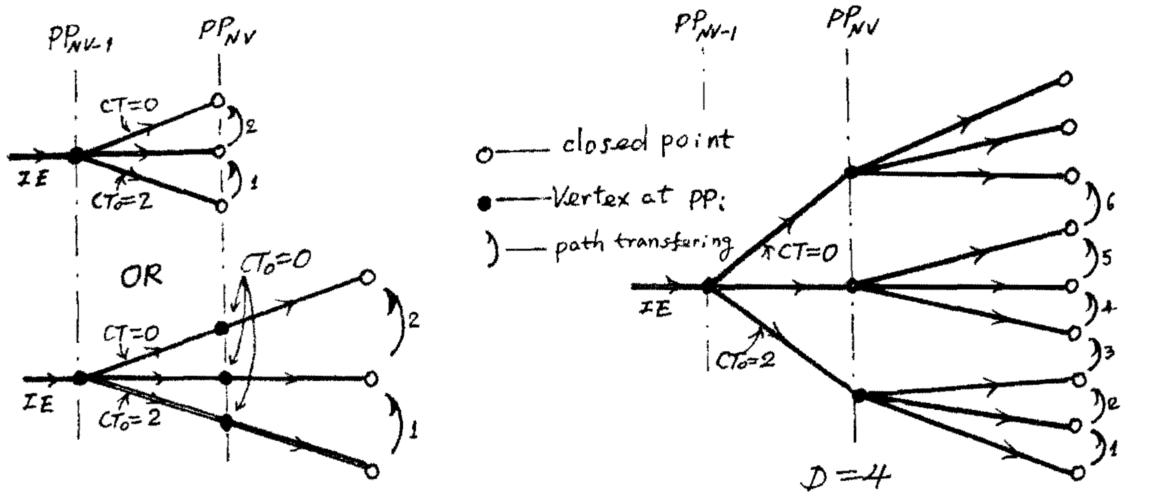
In any of the basic path lines, the the vertex of the point at PP_1 is unchanged.

Proof: The turning point on each basic path line can lie only to the right of PP_1 or at PP_1. In the former case, the path transfering will not change the element at PP_1. In the later case, the path transfering will make $CT(PP_1) := CT(PP_1) - 1$, however, it will not change the vertex of the point itself, i.e., the vertex will not be changed.

Property 5:

Any path in the graph, which starts from any one of EXITs at initial node IN (see Def. 3.7.2), will be contained in a basic path line in PTL related to the graph.

Proof: Fig. 3.9.2.3 illustrates a part of the graph. The vertex V_1 is an initial vertex IV. Assume that $V_1V_2V_3V_4V_5V_6V_7$ is an arbitrary path which starts from an EXIT at V_1. In PTL, the initial vertex V_1 is always located at PP_1, see property 4. According to property 2, the OE at V_1 through limited times of path transfering will be turned into the EXIT which links up with V_2. So that, on the relative basic line, the V_1 will be located at PP_1 and V_2 will be located at PP_2. According to property 2, the OE at V_2



Through $CT_0 = 2$ times of path transfering, OE at PP_{NV-1} will undego all the changes.

Through $CT_0 * (D-1) = 2 * 3 = 6$ times of path transfering, OE at PP_{NV-1} will undego all the changes.

Fig 3.9.2.1

$CT(PP_{nv})=CT_0(PP_{nv})>0$, then the next point must be a closed point and the point P_{nv} must be a terminal point (see property 1 and Def. 3.6.2). Therefore, each path transfering causes the OE of P_{nv} turning the direction once counterclockwise and $CT(PP_{nv}):=CT(PP_{nv})-1$ until $CT(PP_{nv})=0$. In this case the OE at point p_{nv} will undergo all the changes, and the number of possible maximum changes is D-2, related to the case of $CT_0(PP_{nv})=D-2$.

2. If there is a point P_{nv-1} at the position PP_{nv-1} and the initial $CT(PP_{nv-1})=CT_0(PP_{nv-1})>0$ then there are two extreme cases which have to be considered. One is that the next point is a closed point (or the next point is a vertex with $CT_0=0$), then the point P_{nv-1} is a turning point itself and the path transfering will cause the OE at P_{nv-1} turning the direction once and $CT(PP_{nv-1}):=CT(PP_{nv-1})-1$. The other is that the next point is not a closed point but a point, say P_{nv}, with $CT_0(PP_{nv})=D-2$, then through at most D-2 times of path transfering $CT(PP_{nv})$ will be 0 and the point P_{nv-1} will become the turning point, so after the path transfering again the OE at P_{nv-1} will be associated with $CT(PP_{nv-1}):=CT(PP_{nv-1})-1$. The real case is situated between the two extreme cases above, so through 1 to D-1 times of path transfering the OE at P_{nv-1} will turn the direction once, therefore the OE at P_{nv-1} will finally undergo all the changes until $CT(PP_{nv-1})=0$. The total times of path transfering needed for the OE at P_{nv-1} to undergo all the changes, i.e., making $CT(PP_{nv-1})$ from $CT_0(PP_{nv})$ to 0, will be between $CT_0(PP_{nv-1})$ and $CT_0(PP_{nv-1})*(D-1)$, (see Fig. 3.9.2.1).

3. If there is a point P_i at a arbitrary position PP_i and the initial $CT(PPi)=CT_0(PPi)>0$, then one of the extreme case is that the next point related to P_i is a closed point(or the points to the right of P_i are all with $CT_0=0$), then the point P_i is a turning point. Therefore through one time of path transfering the OE at P_i will turn direction once and

3.9 Convergence and exhaustiveness

Here we will summarize some properties of the circuit decomposition algorithm illustrated in PTL, then, we will give two important theorems about the algorithm; one is that the algorithm is convergent and the other is that the circuit set decomposed from a graph is complete under certain conditions.

3.9.1 Some hypotheses about the graph decomposition

Hypothesis 1:

The graph is a connected graph, i.e., between any two vertices in the grsph there is at least one path connecting them.

Hypothesis 2:

The graph is a limited graph, i.e., the number of vertices in the graph is finite.

Hypothesis 3:

The graph is a simple graph, i.e., there is no multiple edges between two adjacent vertices.

Hypothesis 4:

The outer contour of the graph is a circuit, i.e., all the vertices on the outer contour are distinct.

Hypothesis 5:

There is no vertex in the graph which is a deadpoint, i.e., the degree of any vertex in the graph is greater than 1.

3.9.2 Some properties about the algorithm

											Circuit	Note
$4'$	$3'$	$2'$	1°	$5'$	10°	$11'_T$	7°				1	
						11°	$9'_T$				2	
							9°	$8'_T$			3	
								8°	$6'_T$		4	
				$5'_T$					6°		5	
				5°	$6'$	$8'$	$9'$	$11'_T$	10°		/	CD
							$9'_T$	11°	7°		6	
						$8'_T$	9°				7	
					$6'_T$	8°					8	
		$2'_T$			6°						9	
		2°	$6'$	$5'_T$	1°						/	CD
				5°	10°	$11'_T$	7°				10	
						11°	$9'_T$				11	
							9°	$8'_T$			12	
			$6'_T$					8°			/	1SM to 5
			6°	$8'$	$9'$	$11'$	10°	$5'_T$			/	CD
						$11'_T$		5°	1°		/	CD
					$9'_T$	11°	7°				13	
				$8'_T$	9°						14	
	$3'_T$			8°							15	
	3°	$8'$	$6'$	$2'_T$							/	CD
				2°	1°	$5'$	10°	$11'_T$	7°		16	
								11°	$9'_T$		17	
						$5'_T$			9°		/	1SM to 4
			$6'_T$			5°					/	" " 9
			6°	$5'$	1°	$2'_T$					/	CD
				$5'_T$		2°					/	CD
				5°	10°	$11'_T$	7°				18	
						11°	$9'_T$				19	
		$8'_T$					9°				/	1SM to 5
		8°	$9'$	$11'$	10°	$5'$	$6'_T$				/	CD
							6°	$2'_T$			/	CD
						$5'_T$		2°	1°		/	
						5°	1°	$2'$	$6'_T$		/	CD
								$2'_T$	6°		/	CD
				$11'_T$				2°			/	CD
			$9'_T$	11°	7°						20	
$4'_T$			9°								21	
4°	$9'$	$8'$	$3'_T$								/	CD
			3°	$2'$	1°	$5'$	10°	$11'_T$	7°		22	
						$5'_T$		11°			/	1SM to 3
						5°	$6'_T$				/	" 8
				$2'_T$			6°				/	" 9
				2°	$6'$	$5'_T$	1°				/	CD
						5°	10°	$11'_T$	7°		23	
					$6'_T$			11°			/	1SM to 12
		$8'_T$			6°						/	" 15
		8°	$6'$	$2'$	$3'_T$						/	CD
				$2'_T$	3°						/	CD
				2°	1°	$5'$	10°	$11'_T$	7°		24	
						$5'_T$		11°			/	1SM to 4
			$6'_T$			5°					/	" 9
			6°	$5'$	1°	$2'_T$					/	CD
						2°	$3'_T$				/	CD
				$5'_T$			3°				/	CD
				5°	10°	$11'_T$	7°				25	
	$9'_T$					11°					/	1SM to 5
	9°	$11'$	10°	$5'$	$6'$	$8'_T$					/	CD
						8°	$3'_T$				/	CD
							3°	$2'_T$			/	1SM to 8
					$6'_T$			2°			/	" 15
					6°	$2'$	$3'$	$8'_T$			/	CD
							$3'_T$	8°			/	CD
						$2'_T$	3°				/	CD
				$5'_T$		2°	1°				/	1SM to 9
				5°	1°	$2'$	$6'_T$				/	CD
							6°	$8'_T$			/	CD
								8°	$3'_T$		/	CD
						$2'_T$			3°		/	1SM to 15
						2°	$3'$	$8'$	$6'_T$		/	CD
								$8'_T$	6°		/	CD
							$3'_T$	8°			/	CD
		$11'_T$					3°				/	CD
		11°	7°								26	

Fig. 3.8.2

While the circuit decomposition process reaches the terminal state, i.e., the terminal line of PTL is turned out, the process stops automatically and all the circuits desired for the graph have been generated exhaustively, see Sec. 3.9.

3.8 An example of the circuit decomposition process.

Fig. 3.8.1 is a graph of an object. There are eleven vertices altogether in the graph, which are labeled with numbers from 1 to 11. The relative PTL is shown in Fig. 3.8.2. In Fig. 3.8.2, there are two auxiliary columns attached with the list, one is used to record the labels of the circuits which passed the two kinds of checks, the other is used to point out why the circuits generated are discarded. "CD" indicates that the circuit is discarded because of its clockwise direction and "IMS to i" indicates that the circuit is discarded because that it is circuit--i's isomorphic partner. Fig. 3.8.3 illustrates a set of circuits decomposed from the graph of Fig. 3.8.1.

point, and an underline below a path line indicates that that part of the line is the circuit part, see Def. 3.4.8. Because the path transition list is filled point by point and line by line during the point searching and the path transfering processes, from it one can see the whole circuit decomposition process clearly. From the path transfering rules we know that the part of the line, which is to the left of TP, remains the same and the another part is changed because the path transfering takes place at TP. The remaining part and the changed part together compose a new completed basic path line which is just the successor of previous one. If we draw two consecutive lines from the path transition list randomly, it is true that the elements located to the left of the turning point of the line above are the same as the corresponding elements of the line belcw, see Fig. 3.6.3.

3.7 Initial state and terminal state of circuit decomposition process

Obviously, the initial state corresponds to the top line of the PTL and the terminal state corresponds to the buttom line of the PTL.

Def. 3.7.1

IL: The Initial Line of PTL (the first line of the PTL).

Def. 3.7.2

IN: The Initial Node of IL.

The IN includes:

1. The initial vertex, IV.
2. The initial OE at the vertex.
3. The initial virtual IE at the vertex.

The virtual IE means that it is not a real IE because the initial

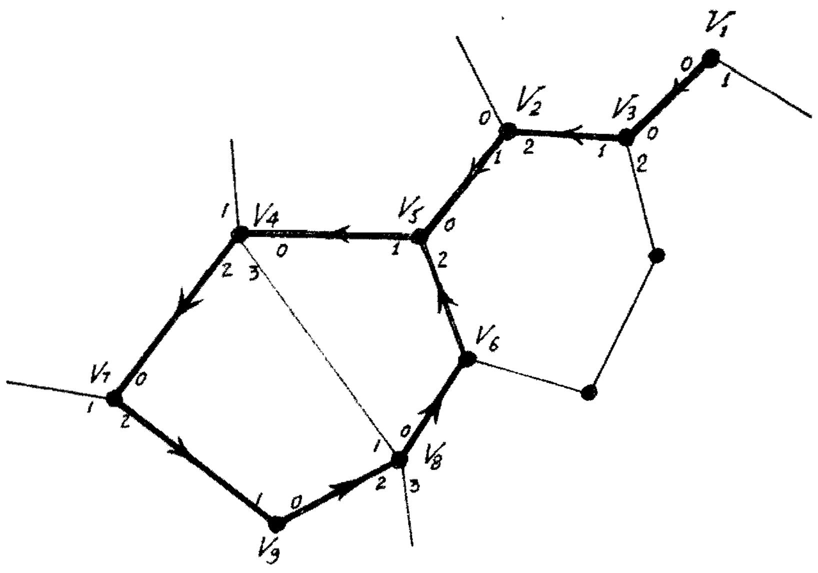


Fig. 3.6.2

3.6.2 are as follows:

CT(1)=0; CT(3)=1; CT(2)=0; CT(5)=1; CT(4)=1;

CT(7)=0; CT(9)=0; CT(8)=1; CT(6)=0.

According to Def. 3.6.2, the TP of the search path is V_8 because it is the right-most point on the basic path line and with CT=1>0.

Briefly, the path transtering rules are as follows:

1. When a searching pointer reached the terminal point, the searching pointer moves to TP on the path;
2. At TP of the path, the OE at that point changes to a new EXIT, i.e., OE:=(OE+1) mod D(V);
3. CT:=CT-1 at TP;
4. A new point searching process discribed in Sec. 3.4 begins, and the searching pointer goes ahead from OE at TP step by step until reaching a

3.6 Path transfering and path transition list (PTL).

In Sec. 3.5 we have discussed that as soon as a closed point has been found, a circuit is generated and the searching pointer has reached the terminal point of the search path. The searching pointer has to seek a new path to find a new circuit.

At what point and in what manner the searching pointer turns to a new path are to be described here.

Def. 3.6.1 Capacity of turning (CT):

The CT is a state parameter associated with a point in a basic path line, which is a positive integer indicating how many times of OE turning will be able to take place at the point. And it is stipulated that each OE turning will cause the OE of path at the point changing to its next left EXIT, i.e.,

$$OE := [(OE+1) \bmod D(V)] \qquad \text{(Eq. 3.6.1)}.$$

According to Def. 3.6.1, the capacity of turning CT associated with a point can be calculated by the following equation.

$$CT=\begin{cases}[\text{LAST EXIT}]-OE, & \text{if } [\text{LAST EXIT}]-OE\geq 0\\ D(V)+\{[\text{LAST EXIT}]-OE\}, & \text{if } [\text{LAST EXIT}]-OE<0\end{cases} \qquad \text{(Eq. 3.6.2)}.$$

Example 3.6.1 :

In Fig. 3.6.1, V_1, V_2, V_3 are three vertices of a graph, the thick piecewise-linear line is a path. According to Eq.3.6.2 , the three state parameters CT(1), CT(2), CT(3) related to the three vertices are as follow:

After passing the two checks, a circuit is labeled and recorded in the two lists: circuit global list and circuit sequence list, see list 3.5.1 and 3.5.2 .

In the circuit global list, the global parameters of each circuit are recorded into five columns from left to right:

1. Circuit label;
2. NVC: The number of vertices in the circuit;
3. NSS: The number of symmetrical sections of the circuit;
4. ONK: Ordering number of the first key point of a circuit;
5. Length of symmetrical section;
6. RLCC code word.

The three encoding parameters of a circuit, ONK, NSS and RCLL code, will be explained in Sec. 4 .

The circuit sequence list consists of NC sublists,where NC is the number of circuits of a graph. The sequential parameters of a circuit, i.e., the vertex labels, the edge angles and the deflective angles around the circuit, are filled into the list row by row from top to down. The left-most column of list is filled with the serial numbers of nodes in circuits starting from 0 which is related with the circuit nodes of the closed points of the basic path lines.

Both the global parameters and the sequential parameters of a circuit can be looked up readily from the above two lists, and these parameters are essantial for the circuit coding process and the matching process.

point.

While the point searching process reaches a terminal point, a circuit represented by the circuit part of the basic path line is generated in the same time. The circuit generated in the searching process is called the crude circuit.

Briefly, the point searching process can be divided into the following steps:

1. Look up tracking list to get next searched point, i.e., To.V.
2. Check the next searched point.
3. If the next searched point is not a closed point, the searching pointer moves to the next point and set the FIRST EXIT at that point as the OE and repeat step 1-2.
4. If a closed point is found, the search process reaches the terminal point and a crude circuit represented by the circuit part of basic path line is generated.

3.5 Circuit checking and circuit recording

While the point searching process reaches its terminal point, a crude circuit is generated at the same time. As the only circuits with the counter-clockwise direction are desired in our case (see sec. 2.3), it is necessary to check up the direction of each crude circuit just generated.

At the each point of searching steps the deflective angle DA is calculated with Eq.3.3.1. If the summation of DA around a crude circuit approximates to $+360^{0}$, i.e., the crude circuit is clockwise one, then it is

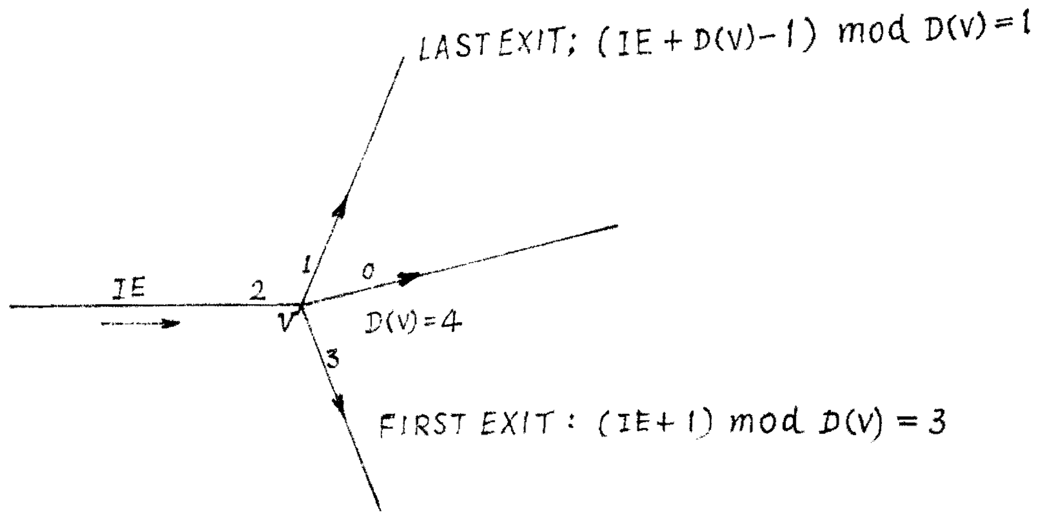

LAST EXIT; (IE + D(V) - 1) mod D(V) = 1
IE
2
1
0
V
D(V) = 4
3
FIRST EXIT : (IE + 1) mod D(V) = 3

Fig. 3.4.1

in point searching process, similar to the roles played by place names, signposts and map of a traffic road.

Assume that there is a searching pointer which is always placed at the advanced point of the search path, and that the vertex and the OE at the advanced point are known, then the next search point of the search path can be looked up from the tracking list(TL), see Sec. 3.2 . In the tracking list, the items To.V and To.E related to the indices, which are the vertex label and OE label at the advanced point (here OE relates to the labeled edge in the TL), are just the next searched vertex and the IE at that vertex respectively. Once a next searched vertex is found through looking up the TL, it is necessary to check whether it is eligible to be a new member of the search path so as to guarantee all the vertices on the path are distinct. If the next searched vertex is a new one, i.e., it is different from all of the vertices

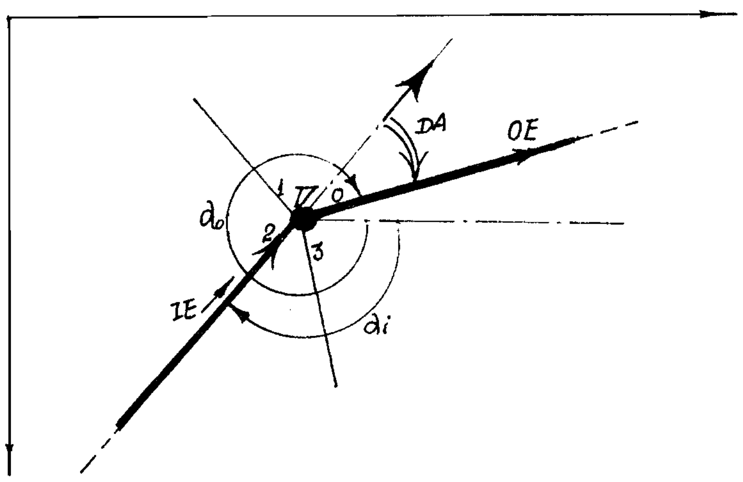


Fig. 3.3.1

by a cycle, one can reach the same edge, i.e., the total directional change around a circuit is either $+360^0$ or -360^0. Furthermore, according to Def. 3.3.3, the sum of all DAs for a clockwise circuit is 360^0, and for a counter-clockwise circuit is -360^0. We will use the $\sum$ DA around a circuit to determine the circuit direction later.

3.4 Point searching and circuit generating processes

Def. 3.4.1 EXIT of IE at a vertex:

The edge at the vertex can potentially be OE (see Def. 3.3.2) of a path at the vertex, so except IE itself, any edge can be a EXIT of the IE.

α: The edge angle.

Example 3.2.2:

Fig. 3.2.2 is the tracking list of the Fig. 3.2.1.

V	D(V)	EDGE: 0			EDGE: 1			EDGE: 2			EDGE: 3		
		TO·V	TO·E	a	TO·V	TO·E	a	TO·V	TO·E	a	TO·V	TO·E	a
1	2	3	0		2	0							
2	3	1	1		4	0		5	0				
3	3	1	0		10	0		4	1				
4	3	2	1		3	2		7	1				
5	3	2	2		7	0		6	0				
6	3	5	2		8	0		9	0				
7	3	5	1		4	2		8	1				
8	3	6	1		7	2		11	1				
9	2	6	2		11	0							
10	2	3	1		11	2							
11	3	9	1		8	2		10	1				

Fig. 3.2.2

Vertex	x	y
1	100	30
2	250	30
3	50	80
4	200	80
5	250	130
6	50	160
7	200	160

Vertex	Connecting to
1	2, 3
2	1, 4, 5
3	1, 4, 6
4	2, 3, 7
5	2, 7
6	3, 7
7	4, 6, 5

Fig. 3.1.4

Def. 3.2.1 Degree of a Vertex D(v):

The number of edges radiating from a vertex V.

Def. 3.2.2 Labels of edges at a vertex:

All the edges radiating from a vertex V are denoted by integer numbers 0--D(v)-1 around the vertex counterclockwise, and the label 0 is used for the edge with the largest edge angle.

Example 3.2.1

The vertex and edge labeled graph of Fig. 3.1.1 is show in Fig. 3.2.1 where the bigger numbers represent vertex labels and smaller numbers represent edge labels.

Coordinate list

Vertex	X	Y
1		
2		
3		
.		
.		
.		
N		

List 3.1.1

Vertex connection list

Vertex	Connecting to
1	
2	
3	
.	
.	
.	
N	

List 3.1.2

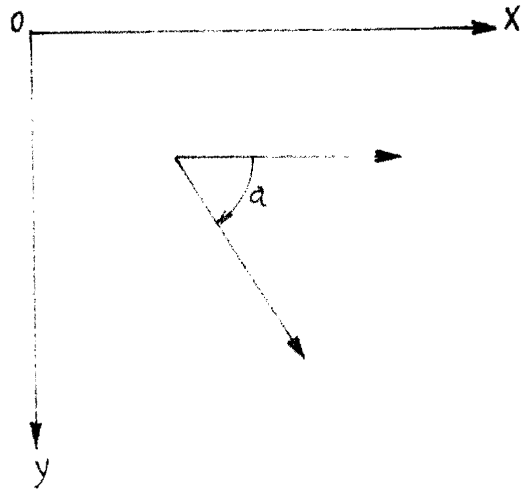


Fig. 3.1.2

The vertex connection list indicates the adjacent connection relationship among the vertices in a graph. In the first column of the list, all the

3. CIRCUIT DECOMPOSITION

3.1 The input of the algorithm

We assume that we have already converted two consecutive images of a moving object into two graphs. Each graph is described by two lists (see below).

For not making this paper too long, we shall report the preprocessing steps in another paper, see[14].

For convinence, we will frequently insert some definitions and examples in the main body of the description, and will make the definitions as close to that of "graph theory" as possible.

Def. 3.1.1 Path:

A sequence of edges in which all the vertices are distinct.

Def. 3.1.2 Connected graph:

It is a graph between any two vertices in it there is at least a path connecting them.

Def. 3.1.3 Circuit:

A closed path.

Def. 3.1.4 Labels of vertices:

The numbers from 1 to N which are used to denote the N vertices of a graph.

Example 3.1.1:

In Fig. 3.1.1, the numbers from 1 to 11 are the labels of vertices; the sequence 1-2-4-7-8-6-9-11 is a path, 1-2-5-6-8-7-4-3-1 is a circuit; but

AN ALGORITHM FOR MATCHING TWO GRAPHS OF A 3-D MOVING OBJECT

W. K. Gu[*], J. Y. Yang[**] and T. S. Huang

Coordinated Science Laboratory

University of Illinois at Urbana-Champaign

1101 W. Springfield Avenue

Urbana, IL. 61801 USA.

ABSTRACT

We present an approach for finding corresponding points between two graphs derived from perspective views of a moving object whose surface is composed of planar polygons. In our approach, each circuit of the graph is encoded with a kind of shape code which we call the RLCC code, then a relaxation algorithm is used to raise the confidence of matching, and to get the final matching result between the two graphs. The basic idea of our approach is different from those of existing matching methods using cross correlation, relational stracture, elastic templates, and so on.

A series of experiments have shown that our approach works well even when considerable perspective shape distortions, scaling and dissimilarities exist between the two graphs.

Keywords: Matching, Shape, Relaxation, Graphs.

---

[*] On leave from Zhejiang University, Hangzhou, China.

[**] On leave from East-China Engineering Institute, Nanjing, China.

1. INTRODUCTION

Finding corresponding points in images is of fundamental importance in image analysis problems such as the estimation of the three-dimensional motion parameters of a rigid body from two consecutive images [1, 2, 3].

After three-dimensional translation and rotation, the image of a object will suffer severe perspective shape distortion and scaling. It is very difficult for the traditional approaches based on local measurements of cross correlation between two images to tolerate these geometrical changes.

Recently, a number of papers have presented techniques of matching corners, edges, and by using their geometrical and topological relationships. For instance, Yam and Davis used the Generalized Hough Transform to solve the problem of line drawing graph registration [4]; Ranade and A. Rosenfeld, J. Q. Fang and T.S.Huang used relaxation algorithms to find the correspondence between two corner patterns extracted from two images [5, 6]; Burr, Bajcsy and Broit applied the elastic method to deal with the matching line drawing problem [7,8]; Hwang and Hall implemented the method of two stereo images with the aid of sets of relational tables[9].

In this paper, we shall present a new approach of matching two line drawing graphs corresponding to two consecutive images of a moving object whose surface is composed of polygons. The results of a series of experiments have indicated that our approach of using RLCC code of composite circuits between two graphs is quite powerful in case where considerable perspective shape distortions and scaling ex ist between the two graphs.

Breifly, our approach consists of the following steps:

1. Through low level image processing, get two connected line graphs from the two consecutive images of a moving object [10, 11];
2. Decompose the two graphs into two sets of composite circuits exhaustively;
3. Encode every circuit with RLCC code;
4. Search candidates of matched circuit pairs according to RLCC code;
5. According to the clustering property of correct matched point pairs and the consistent property among the real corresponding circuit pairs between two graphs, use relaxation algorithm to raise the confidence iteratively and get the final matching results.

Apart from tolerating perspective shape distortions and scaling, another advantage of our matching approach is that: Since it uses composite circuits as the basis of matching, it can stand up to considerable dissimilarities between the two graphs caused by shading or noise in the two original images (see Sec. 2). So our approach is quite stable for operating in an environment of noise and variation of illumination.

In Sec. 2, we describe the main concept of the composite circuits and the RLCC code. In Sec. 3, we present an algorithm for decompose a graph into a complete set of the composite circuits, and give the proofs of convergence and exaustiveness of the decomposition algorithm. In Sec. 4, we discuss the RLCC coding process. In Sec. 5 and Sec. 6, we present the matching process and enumerate a series of experimental results. The experimental results illustrate that our algorithm is effective and reliable.

2. COMPOSITE CIRCUIT AND RLCC CODE

2.1 Simple circuit

A simple circuit is a closed path in a graph and it contains no interior branches, in our case, it is a polygon which represents one of the following three cases:

1. An exposed face of an object (polyhedron), such as the circuit-$V_1V_2V_3V_4$ and the circuit-$V_3V_{20}V_{23}V_4$ in Fig. 2.1.1;
2. The exposed part of an object face, such as the circuit-$V_{20}V_{21}V_{22}V_{23}$ in Fig. 2.1.1;
3. A part of background enclosed by an object, such as the circuit-$V_{23}V_{22}V_{10}V_{25}V_{24}$ in Fig. 2.1.1.

Hereafter, we will use a vertex sequence ordered in counterclockwise direction to represent a circuit. At each vertex of the sequence the path changes the direction once.

2.2 The dissimilarity and the merge of simple circuits

In two consecutive graphs corresponding to two sequential views of a moving object, it may happen that some edges exist in one of the two graphs but disappear in the other because of noise or variation of the illumination. This phenomenon will be called "dissimilarity" in this paper.

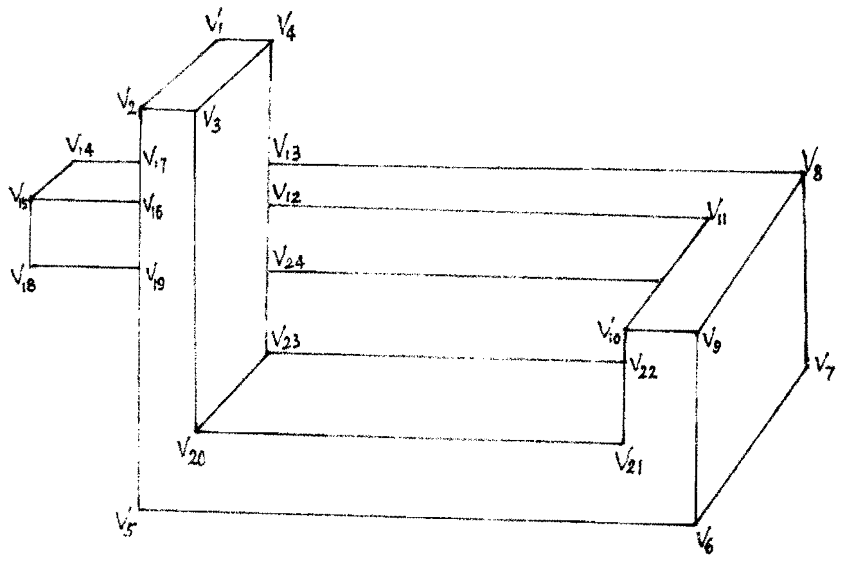


Fig. 2.1.1

In a graph, the loss of a common edge or a common edge sequence between two adjacent simple circuits makes the two circuits merge into one. Similarly, three or more simple circuits may merge into one, see Fig. 2.2.1.

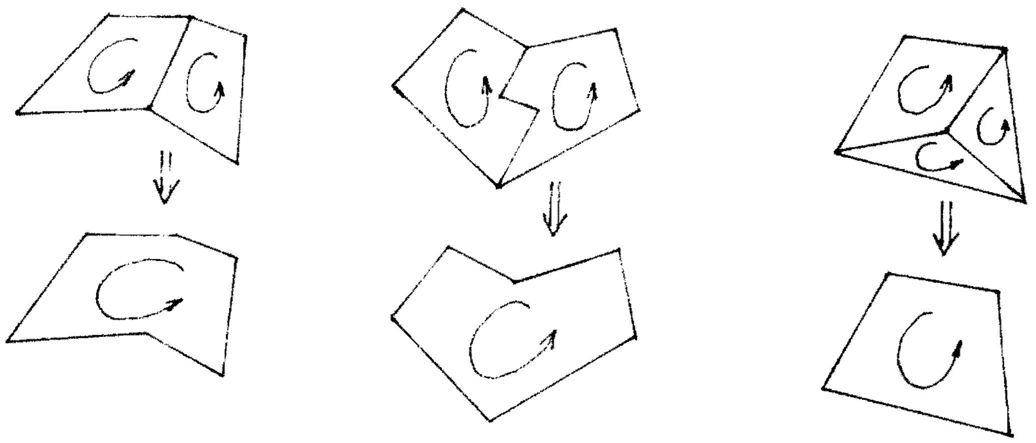
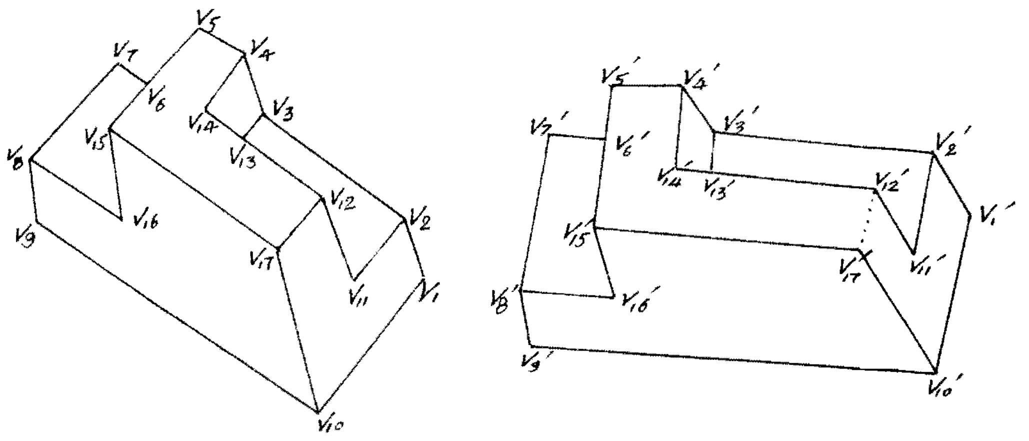


Fig. 2.2.1

In Fig. 2.2.2, (a) and (b) is a pair of graphs of the same moving object. Assume that edge $V_{12}V_{17}$ exists in (a) but its corresponding edge $V'_{12}V'_{17}$disappears in (b), then, it is impossible that the simple circuits which contain edge $V_{12}V_{17}$ in (a), such as circuit-$V_{12}V_{14}V_{4}V_{5}V_{15}V_{17}$ and circuit-$V_{1}V_{2}V_{11}V_{12}V_{17}V_{10}$, have the corresponding circuits in (b). However, the merged circuit-$V_{12}V_{14}V_{4}V_{5}V_{15}V_{17}V_{10}V_{1}V_{2}V_{11}$ has its corresponding circuits in (b), i.e., circuit-$V'_{12}V'_{14}V'_{4}V'_{5}V'_{15}V'_{17}V'_{10}V'_{1}V'_{2}V'_{11}$. We define simple circuits and possible merged circuits together as composite circuits, simply, as circuits.

It can be imagined that using the composite circuits as bases for matching instead of using the simple circuits the capability of standing up to the



(a) (b)

Fig. 2.2.2

dissimilarities will be much stronger. Not being able to predict the dissimilarities within two graphs, in order to deal with the problem we should design an approach to generate all the composite circuits of two graphs exhaustively before the begining of the matching process.

2.3 RLCC code

What features should be extracted from a circuit for the efficient matching is to be discussed here. As we know, a circuit is a polygon in a graph plane which contains certain geometrical information, such as the area, perimeter etc. Besides, a circuit is composed of a series of edges around it , and each of the edges contains its own geometrical information too, such as edge length and edge angle, i.e., the angle between the edge and the X-coordinate axis.

Because of the scaling and shape distortions, the absolute geometrical measurements of a circuit in a graph such as area, perimeter, edge length and edge angles can hardly be used as the features to identify its correspondent in the other graph.

Although the relative geometrical measurements, such as the series of relative length of edges, i.e., the series of ratios of edge lengths to the perimeter of the circuit, are unvaried with scaling and two-dimentional rotation, they are still somewhat sensitive to the shape distortions caused by perspective projection. Hence, it is unprofitable to use the relative geometrical measurements as the main features in the matching process either.

The motivation of seeking the features of circuits which are insensitive to the scaling, rotation and shape distortion had stimulated us to use the

RLCC code of circuit.

RLCC code is the abbreviation of the Run Length Code for the Convex and Concave Vertex String of a circuit.

The RLCC code is a sort of circuit shape which consists of a string of integer numbers. Each number equals to the run length of convex vertex string or run length of concave vertex string around a circuit alternately with the stipulation that the first number in code word has to be related to a convex vertex string in a circuit. Therefore, the numbers located at the odd positions of a RLCC code are related to the convex vertex string and the numbers located at the even positions are related to the concave vertex string constantly.

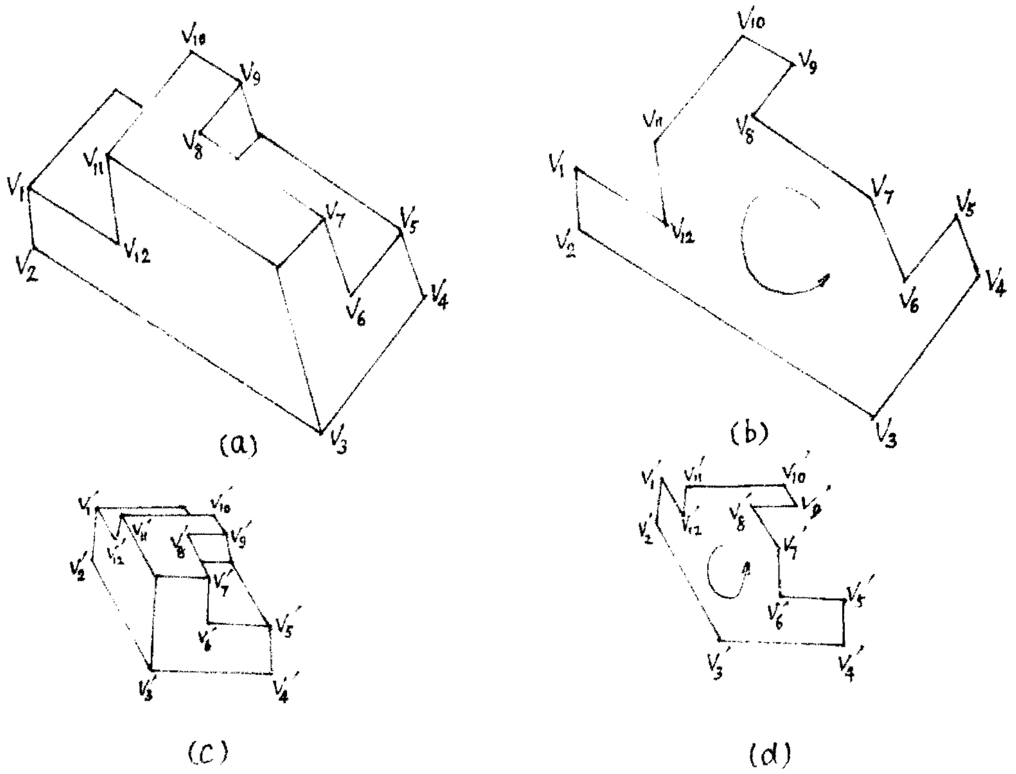


Fig. 2.3.1

For example, in Fig. 2.3.1, (a) and (c) are two consecutive graphs of same object, (b) is a composite circuit of (a), i.e.,

$V_1V_2V_3V_4V_5V_6V_7V_8V_9V_{10}V_{11}V_{12}$,

(d) is a composite circuit of (c), i.e.,

$V'_1V'_2V'_3V'_4V'_5V'_6V'_7V'_8V'_9V'_{10}V'_{11}V'_{12}$.

In (b), there arrange the convex string $V_1V_2V_3V_4V_5$, concave string V_6 (only one element in the string), convex string V_7, concave string V_8, convex string $V_9V_{10}V_{11}$ and concave string V_{12} around the circuit ordinally. According to the definition, the circuit in Fig.2.3.1 (b) has its RLCC code 511131. Likewise, the circuit in Fig. 2.3.1 (d) has its RLCC code 511131 too.

From Fig. 2.3.1, we see that the two graphs (a) and (c) are different because of the considerable deformation caused by scaling, rotation and perspective distortion. However, the two corresponding circuits in (b) and (d) have the same RLCC code: 511131.

It is clear that unlike the geometrical measurements and their varieties, for instance, the CHAIN code of a circuit [12, 13], the RLCC code is indeed insensitive to the shape deformation. Therefore, we will use the RLCC code of circuits for the matching process. How to encode circuits and how to remove the ambiguity in the RLCC code due to different starting points will be described in Sec. 4.

1-2-4-7-8-11-10-3-4-2-5 is not a path and 1-2-5-7-4-2 is not a path either.

One kind of input lists is a coordinate list, see List 3.1.1, and the other is a vertex connection list, see List 3.1.2 . A graph is completely characterized by these two lists.

In the coordinate list, every vertex's coordinate values (x,y) were filled during the preprocess. Here we have to point out that in our case the y-axis in coordinate plane points downward, so any edge angle, i.e., the angle between a edge and x-axis, has the positive incremental value along the clock-wise direction, see Fig. 3.1.2.

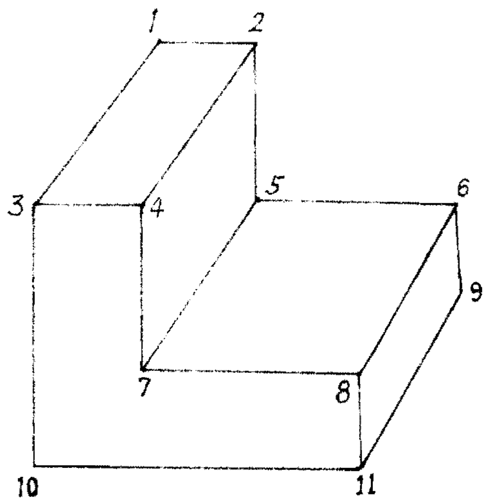


Fig. 3.1.1

vertices of the graph were listed row by row, and in each row of second column, all the adjacently connected vertices to the vertex in the first column were listed.

Example 3.1.2 ;

Fig. 3.1.3 is a connected graph. The two lists associated with the graph of Fig. 3.1.3 are illustrated in Fig. 3.1.4.

3.2 Tracking list (TL)

By now, what we have in hand are the two input lists for each graph. In order to use the input data readily in the circuit decomposition process, we should devise a suitable data structure to store all the useful information which can be deduced from the original input data. The tracking list is the very structure we desired.

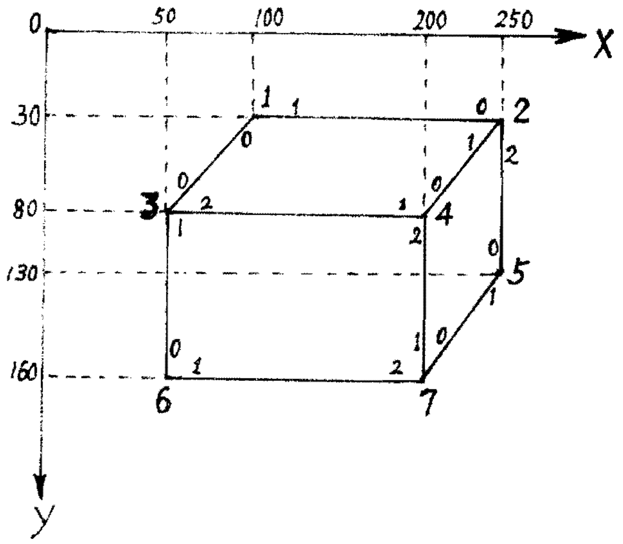


Fig. 3.1.3

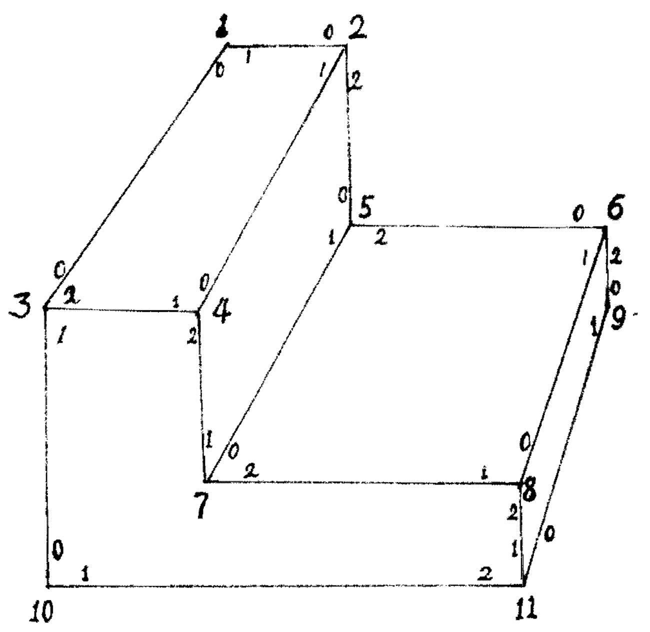


Fig. 3.2.1

Here we should point out that each edge in a graph, generally, has two different labels at its two ends.

The tracking list is a group of sublists indexed with the vertex labels located at the leftest column. Each sublist is filled with the following data from left to right:

1. Vertex label;

2. Degree of vertex;

3. All the labeled edges radiated from the vertex and their attributes:

   To.V: The opposite vertex label which the edge connects to.

   To.E: The opposite edge label, i.e., the edge label at another end of the edge;

3.3 The circuit direction

There are two circuit directions, clockwise and counterclockwise. We only select the circuits with counterclockwise direction as the desired circuits to be used to the matching process.

Def. 3.3.1 IE:

Incoming edge at a vertex on a path.

Def. 3.3.2 OE:

Outgoing edge at a vertex on a path.

Def. 3.3.3 DA:

The deflective angle between the directions of IE and OE.

Example 3.3.1:

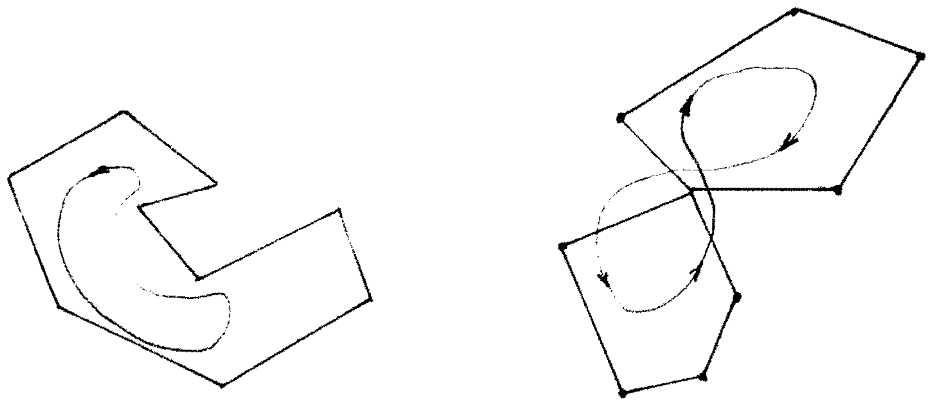
In Fig.3.3.1, V is a arbitrary vertex of a graph; 0-3 are the edge labels at the vertex; the thick piecewise-linear line is a part of a path; the arrows of the path indicate the path direction. Then, the IE of the path at V is edge 2 (but the direction is opposite), and the OE of the path at V is edge 0 as well as the deflective angle of the path at V is DA.

From Fig.3.3.1, DA satisfies following equation:

$$DA=\begin{cases}\alpha_0-\alpha_i, & \text{if } (0^0 \leq \alpha_0-\alpha_i \leq 180^0) \text{ or } (-180^0 \leq \alpha_0-\alpha_i \leq 0^0) \\ \alpha_0-\alpha_i-360^0, & \text{if } \alpha_0-\alpha_i > 180^0 \\ \alpha_0-\alpha_i+360^0, & \text{if } \alpha_0-\alpha_i < -180^0\end{cases} \qquad \text{(Eq. 3.3.1)}$$

where α_0 is the angle of edge 0 (OE) and α_i is the angle of edge 2 (IE) in Fig. 3.3.1.

According to Def.3.1.1 and Def.3.1.3, any circuit is a closed path without crossing point on it, see Fig.3.3.2. Setting off from any edge along a circuit



a circuit not a circuit

Fig. 3.3.2

Def. 3.4.2 FIRST EXIT of IE at a vertex:

The immediate right EXIT faced by the IE at the vertex, i.e., FIRST EXIT=[(IE+1) mod D(V)].

Def. 3.4.3 LAST EXIT of IE at a vertex:

The immediate left EXIT faced by the IE at the vertex, i.e., LAST EXIT=[(IE+D(v)-1) mod D(v)].

Example 3.4.1:

In Fig. 3.4.1, V is a arbitrary vertex of a graph, the degree of V is four, i.e., D(v)=4, the edge 2 is the IE of a search path at V, edge 3,0,1 are all EXITs of IE, edge 3 is the FIRST EXIT and edge 1 is the LAST EXIT.

In the point searching process, a search path is extended vertex by vertex under the control of certain rules to be described. Each vertex and OE of the path at the vertex as well as the tracking list play the important roles which have already been on the path, it is accepted to be a new member of search path, and the search path is said to be extented one step. The searching pointer then moves to the new vertex. The OE of the new vertex is stipulated to be the FIRST EXIT of the IE (IE has been looked up from TL ,i.e., To.E). This process goes on repeatedly, and the search path is extended step by step. If the next searched vertex looked up from TL is the old one, i.e., just the one which has already been on the path, then a circuit has been generated.

Apart from the items To.V and To.E, the edge angle α is picked up from the tracking list in every searching step, and the deflective angle DA then is calculated with Eq. 3.3.1 . The parameters at every node of a circuit are useful in the matching process.

Def. 3.4.4 closed point:

In the point searching process, if the next searched vertex is just the old one on the path, then this old vertex is called a closed point.

Def. 3.4.5 basic path line:

It is a recording line of vertex labels which is used to record the search path.

Def. 3.4.6 Terminal point of a basic path line:

When a closed point is found, the vertex at the end of a besic path line is the terminal point(notice the closed point is not the terminal point).

Def. 3.4.7 Inducing part of a basic path line:

The part of a basic path line to the left of the closed point.

Def. 3.4.8 Circuit part of a basic path line:

The part of basic path line, which is from closed point to terminal determineted that the circuit is not desired and is discarded at once.

Def. 3.5.1 Isomorphic circuit pair:

If the vertex sequence of a circuit can be transfered by circular shifts into that of another circuit, then the two circuits are an isomorphic circuit pair, and each circuit is called the isomorphic partner of the other.

Example 3.5.1:

Assume that circuit i and circuit j are associated with the following vertex sequences, respectively:

circuit i: 3, 4, 5, 7, 9, 11;

circuit j: 5, 7, 9, 11, 3, 4;

The vertex sequence of circuit i through 2 times of circular left shifts will become that of circuit j, so that circuit i and circuit j are an isomorphic circuit pair, and circuit i is the isomorphic partner of circuit j, vice versa.

It is clear that if two circuits are an isomorphic circuit pair, they will conserve the same information in them, i.e., keeping one of them and discarding another will not cause information loss at all. Hence, in our algorithm, before circuit recording, a circuit has to be compared with all of the circuits which have already been recorded. If the circuit is an isomorphic partner of any circuit which has already been recorded, then it will be discarded at once, otherwise it will be labeled and recorded in a proper list.

Def. 3.5.2 circuit label:

Each circuit which has passed the above two checks, i.e., the directional check and the isomorphic check, is denoted by a integer number sequentially.

Circuit global list

Circuit	NVC	Encoding	Parameters		
		NSS	ONK	LSS	RLCC code
1					
2					
3					
.					
.					
.					
.					
NL					

List 3.5.1

Circuit sequence list

Node №	Crt 1			Crt 2			Crt 3			Crt 4			.			.			.			.		
	V	α	DA	V	α	DA	V	α	DA	V	α	DA	V	α	DA	V	α	DA	V	α	DA	V	α	DA
0																								
1																								
2																								
3																								
4																								
5																								
.																								
.																								
.																								
.																								
.																								
.																								
.																								

List 3.5.2

at V_1 , {[LAST EXIT]-OE}=2-0=2$\geq$0, so

CT(1)=2;

at V_2 , {[LAST EXIT]-OE}=1-0=1$\geq$0, so

CT(2)=1;

at V_3 , {[LAST EXIT]-OE}=2-4=-2<0, so

CT(3)=D(V_3)+(-2)=5-2=3.

Def. 3.6.2 Turning Point (TP):

The path transfering takes place at that point in a basic path line at which the searching pointer turns to a new path. It is stipulated that the turning point is the right-most point on a basic path line with CT>0.

Example 3.6.2

In Fig. 3.6.2, V_1--V_6 are vertices of a graph, the thick piecewise-linear line is a completed search path, V_5 is the closed point and V_6 is the terminal point. The basic path line is below:

$V_1V_3V_2\underline{V_5V_4V_7V_9V_8V_6}$ or $132\underline{547986}$

Here the underline below the basic path line, according to Def. 3.4.8, indicates the circuit part of the line. The CT(1)--CT(6) associated with the vertices along the basic path line have been calculated with Eq.

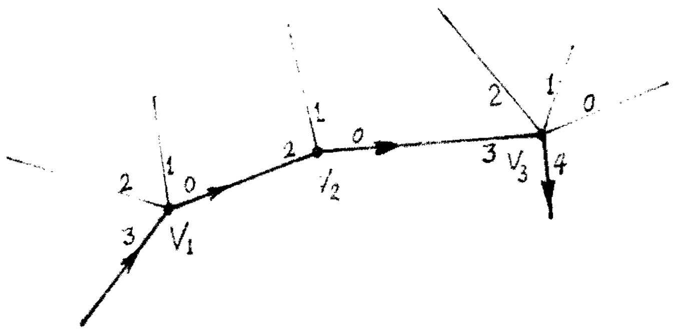


Fig. 3.6.1

new terminal point.

The path transition list (PTL) is used to record the entire circuit decomposition process. The path transition list is actually a set of basic path lines located from top to bottom to illustrate how each basic path line is deduced from its ancestor which is a basic path line located above and how it is transtered to its successor which is a basic path line located below.

A segment of PTL

$\underline{4'}$	$\underline{3'}$	$\underline{2'}$	$\underline{1^0}$	$\underline{5'}$	$\underline{10^0}$	$\underline{11'_T}$	$\underline{7^0}$		
$\underline{4'}$	$\underline{3'}$	$\underline{2'}$	$\underline{1^0}$	$\underline{5'}$	$\underline{10^0}$	$\underline{11^0}$	$\underline{9'_T}$		
$4'$	$\underline{3'}$	$\underline{2'}$	$\underline{1^0}$	$\underline{5'}$	$\underline{10^0}$	$\underline{11^0}$	$\underline{9^0}$	$\underline{8'_T}$	
$4'$	$3'$	$\underline{2'}$	$\underline{1^0}$	$\underline{5'}$	$\underline{10^0}$	$\underline{11^0}$	$\underline{9^0}$	$\underline{8^0}$	$\underline{6'_T}$
$4'$	$3'$	$2'$	1^0	$\underline{5'_T}$	$\underline{10^0}$	$\underline{11^0}$	$\underline{9^0}$	$\underline{8^0}$	$\underline{6^0}$
$4'$	$3'$	$2'$	1^0	$\underline{5^0}$	$\underline{6'}$	$\underline{8'}$	$\underline{9'}$	$\underline{11'_T}$	$\underline{10^0}$
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.

Fig. 3.6.3

Fig. 3.6.3 is a segment of a path transition list, each row in it is a basic path line, the numbers are the vertex labels on the path lines. A superscript number of vertex label indicates the value of CT associated with the point, a subscript T of a vertex label indicates that the point is the turning vertex is a starting point itself on a path. The initial virtual IE at the initial vertex is used only to decide the value of CT at it.

From the point searching and path transfering process, we know that as long as the IN has been chosen the initial line can be developed, and the successive lines of the initial line, i.e., the second line, then the third line in the PTL and so on, can be developed too.

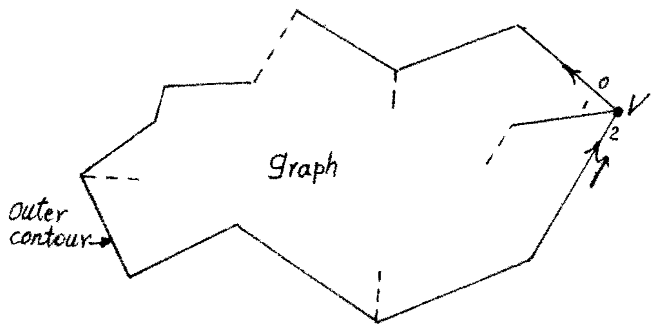
In order to obtain all the desired circuits for a graph exhaustively, the initial node IN has to meet the following conditions. (this will be proved in Sec.3.9.)

1. The initial vertex is on the outer contour of the graph.

2. The initial OE and the initial virtual IE at the IV coincide with the outer contour counterclockwise, see Fig. 3.7.1.

Def. 3.7.3 The terminal line of PTL:

A basic path line on which all the points with CT=0, i.e., no turning point can be found on the line.



V: initial vertex. E_0: initial OE at V.

E_2: initial virtual IE at V. CT(V)=1 at initial state.

Fig. 3.7.1

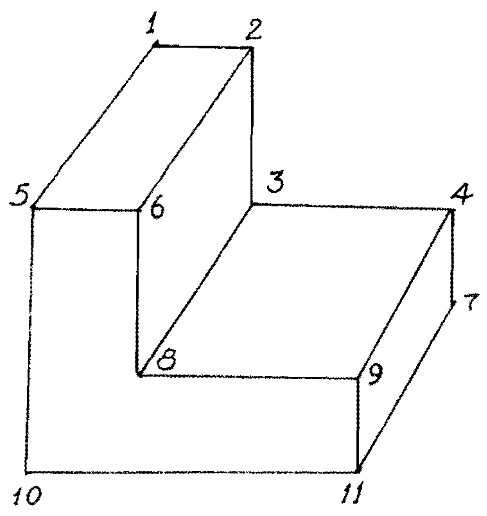


Fig. 3.8.1

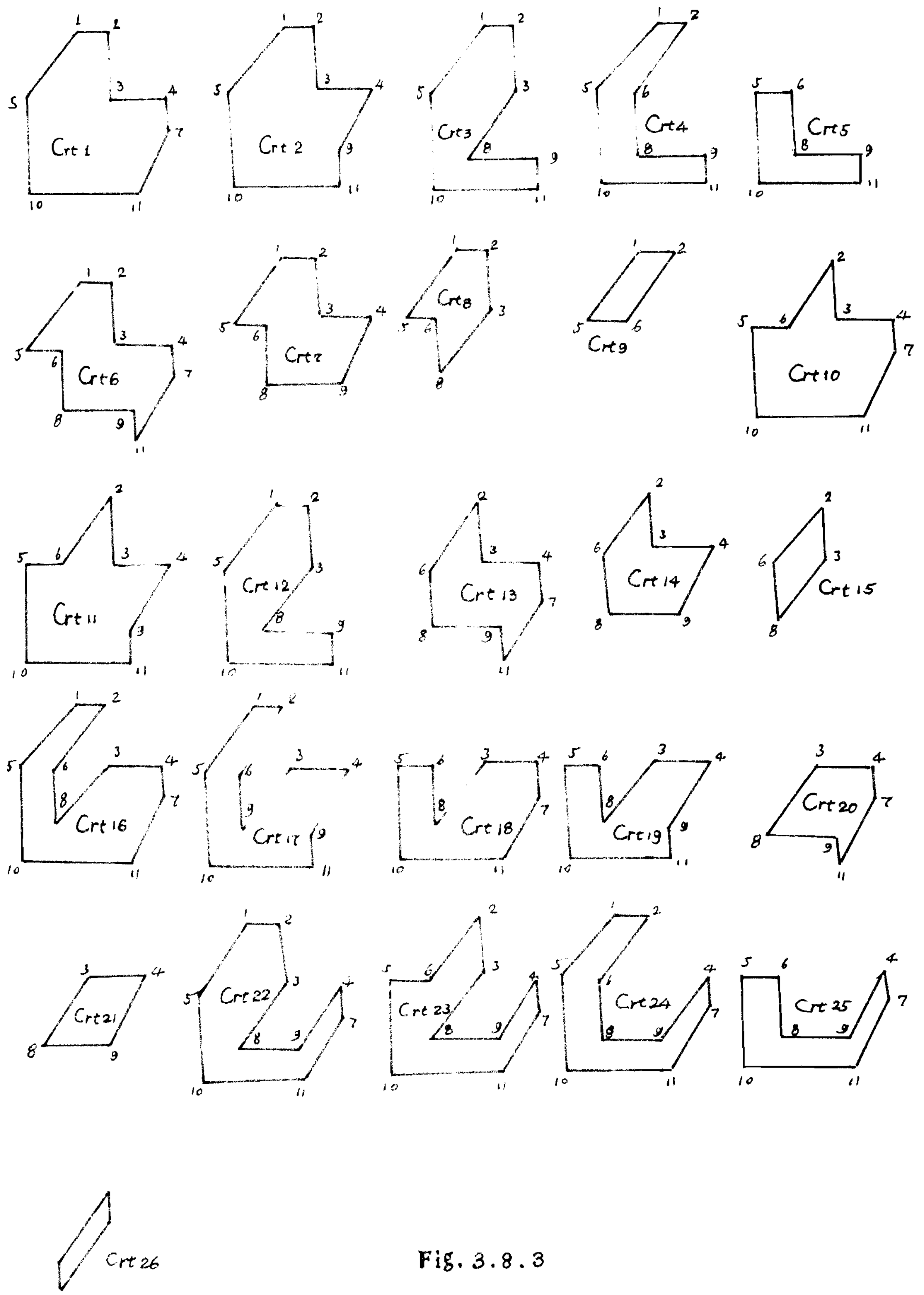

Crt 1
Crt 2
Crt3
Crt4
Crt5
Crt6
Crt7
Crt8
Crt9
Crt10
Crt 11
Crt 12
Crt 13
Crt 14
Crt 15
Crt 16
Crt 17
Crt 18
Crt 19
Crt 20
Crt 21
Crt 22
Crt 23
Crt 24
Crt 25
Crt 26

Fig. 3.8.3

Property 1:

The length of any basic path line is limited. (Here we define the number of vertices on the line as the length of the basic path line.) And the longest basic path line in PTL cannot exceed NV, where NV is the number of vertices in the graph.

Proof: Because the number of vertices in the graph is finite and the vertices on a basic path line are distinct, the length of the basic path line has to be limited. If the graph is a Hamiltonian graph, the longest basic path line in the PTL will just relates to the Hamiltonian circuit of the graph whose length will equal to NV, otherwise any basic path line in PTL will be shorter than NV.

Def. 3.9.1 PP:

The PP representes the position of a point in a basic path line of PTL ordering from left to right.

Property 2:

Through a limited number of path transfering at any PP, an OE of a vertex will undergo all the changes from the FIRST EXIT to the LAST EXIT with the value of CT changing from the maximum value $CT_0 = D(V)-2$ to 0.

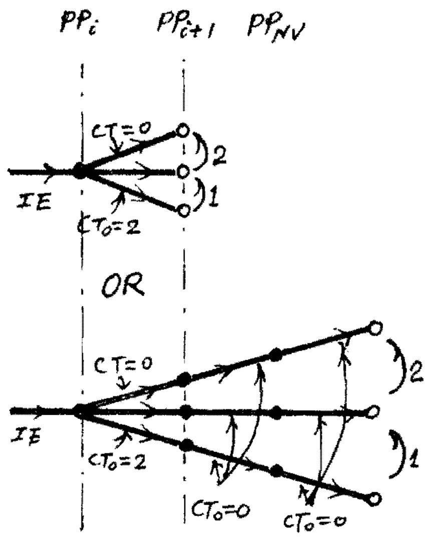
(In the proofs of the properties 2, 3, and 4, we will, for convenience, use the symbol of point posision PP_i as the argument of CT, for instance, $CT(PP_1)$, $CT(PP_{nv})$ etc. Generally, $CT(PP_i)$ is equivalent to CT(V) where V is a vertex related to a point at PP_i on the basic path line.)

Proof: Assume $\max\{D(V_i)\}=D$, $i=1,\ldots\ldots,NV$,

then $\max\{CT(V_i)\}=D-2$.

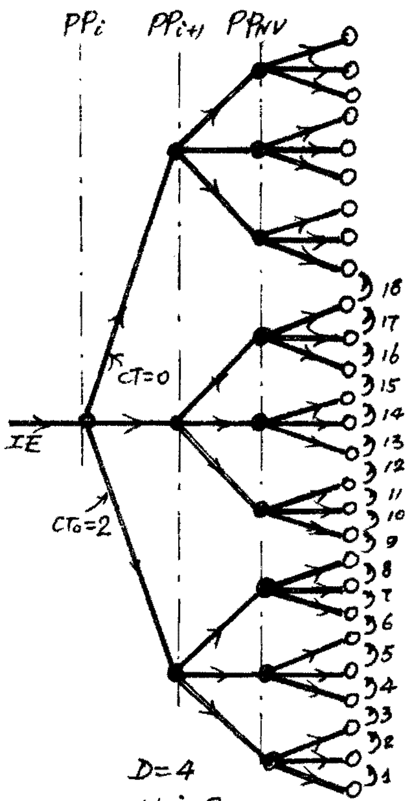
1. If there is a point P_{nv} at the position PP_{nv} and the initial

$CT(PP_i):=CT(PP_i)-1$. Another extreme case is that after each path transfering there are NV−i points to the right of P_i on the path line and each successive vertex with $CT_0(PP_{i+j})=D-2$, j=1 to (NV−i), then through $(D-1)^{nv-i}$ times of path transfering $CT(PP_{i+1})$, $CT(PP_{i+2})$,......, $CT(PP_{NV})$ will be all zero and vertex P_i will be the turning point. At the next path transfering the OE at P_i will turn direction once and $CT(PP_i):=CT(PP_i)-1$. The real case is situated between the two extrem cases, so through 1 to $(D-1)^{nv-i}$ times of path transfering the OE at P_i will turn the direction once. Therefore, the OE at P_i will finally undergo all the changes until $CT(PP_i)=0$. The total times of path transfering needed for the OE at P_i to undergo all the changes, i.e., making $CT(PP_i)$ changing from $CT_0(PP_i)$ to 0, will be between $CT_0(PP_i)$ and $CT_0(PP_i)*(D-1)^{nv-i}$, (see Fig. 3.9.2.2).



$NV-i=2$

Through $CT_0=2$ times of Path transfering, OE at PP_i will undego all the changes.



$D=4$
$NV-i=2$

Through $CT_0 * (D-1)^{NV-i} = 2 * 3^2 = 18$ times of path transfering, OE at PP_i will undego all the changes.

Fig 3.9.2.2

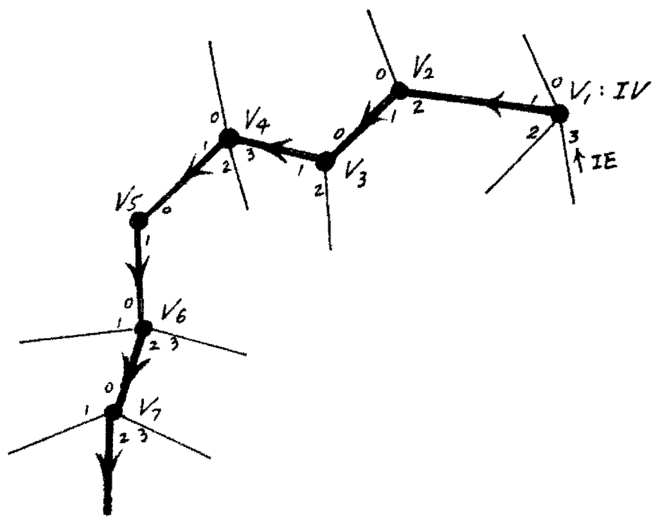


Fig. 3.9.2.3

through limited times of path transfering will be turned into an EXIT which links up with V_3. So that, on the relative basic line, the V_3 will be located at PP_3, and according to property 3 the vertex V_1 and V_2 at PP_1 and PP_2 are unchanged respectively. For the same reason, a basic path line, on which V_1 is located at PP_1, V_2 is located at PP_2, ..., V_7 is located at PP_7, will be turned out through limited times of path transfering.

Property 6:

Any path in the graph which starts from IN and along an edge in the opposite direction of IE is not contained at PTL.

Proof: Fig. 3.9.2.4 illustrates a part of graph. The vertex V_1 is an initial vertex IV. Assume that $V_1V_2V_3V_4$ is a arbitrary path which starts from an edge in the opposite direction of IE at IV. In PTL, the initial vertex is always located at PP_1. According to property 2, the OE at IV will undergo all the changes from FIRST EXIT to LAST EXIT, i.e., the OE at IV changes from edge 0 to edge 2. When the OE has been turned into the LAST

The algorithm is convergent.

Proof: According to hypothesis 4 and property 1, there is no vertex with D(V)<2 in the graph, i.e., no deadpoint in the graph. It is guaranteed that each point searching process will reach a terminal point through a limited number of searching steps (see property 1), so that each point searching process is convergent. If we start from the LAST EXIT of IV and at each vertex always select the LAST EXIT as the EXIT to go out, we can reach a closed point. The path which was just passed corresponds to a basic path line. According to the property 5, the basic path line is in the PTL. From the description above we know that the OE at any vertex in the path line is the LAST EXIT, i.e., all the CTs on the line are zero. Therefore, it is the bottom line of the PTL. Therefore, the PTL always has the bottom line, i.e., the decomposition process is always convergent.

Theorem 2:

All the desired circuits are contained in the PTL (i.e., the circuit decomposition process is exhaustive), iff the IV lies on the outer contour of the graph and the initial OE and the initial virtual IE (see Def.3.7.2) coincide with the outer contour counterclockwise.

Before proving theorem 2, let us prove the following propositions:

Proposition 1:

If the IV is on the outer contour of the graph and if there is a circuit which does not contain the IV, then we can always find out a path which starts from one of the EXITs at IV and connects to the circuit.

Proof: Assume there is no path connecting IV with the circuit, then it is contradictory to hypothesis 1, so it is impossible. Assume that the paths connecting IV with the circuit only pass along the edge with the opposite direction of IE at IV, i.e., there is no other paths connecting IV with the

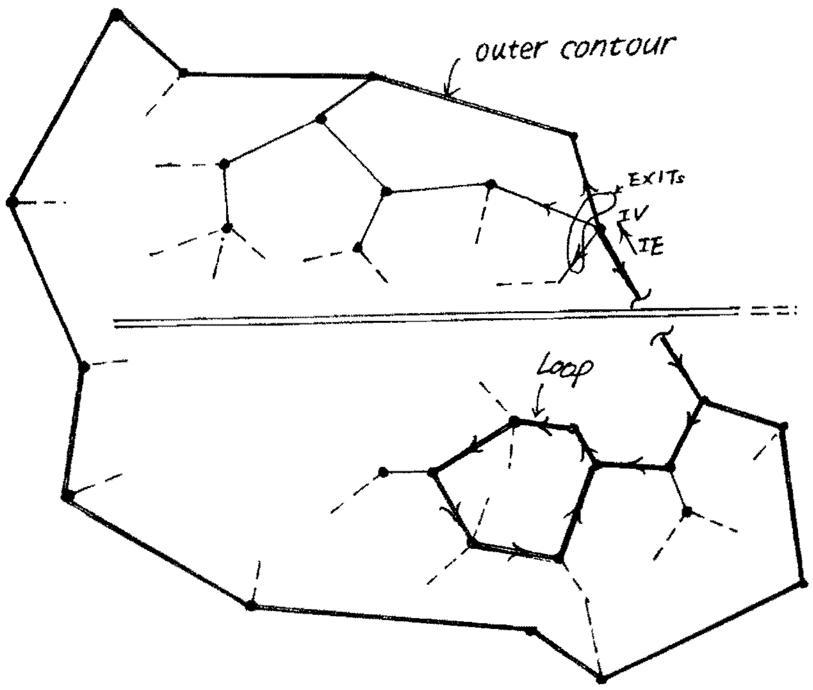


Fig. 3.9.3.2

Def. 3.9.3.1 Inner plane and outer plane of counterclockwise circuit:

When one marches along a circuit counterclockwise, the plane located to the left of the marching direction is the inner plane of the counterclockwise circuit. The plane located to the right of the matching direction is the outer plane of the circuit, see Fig. 3.9.3.3(a).

Def. 3.9.3.2 Inner plane and outer plane of clockwise circuit:

When one marches along a circuit clockwise, the plane located to the right of the marching direction is the inner plane of clockwse circuit. The plane located to the left of the matching direction is the outer plane of the circuit, see Fig. 3.9.3.3(b).

Now we give the proof of proposition 2.

Proof: A circuit divides a plane into two parts, one is the inner part, the other is the outer part. According to Def. 3.9.3.1, martching along the

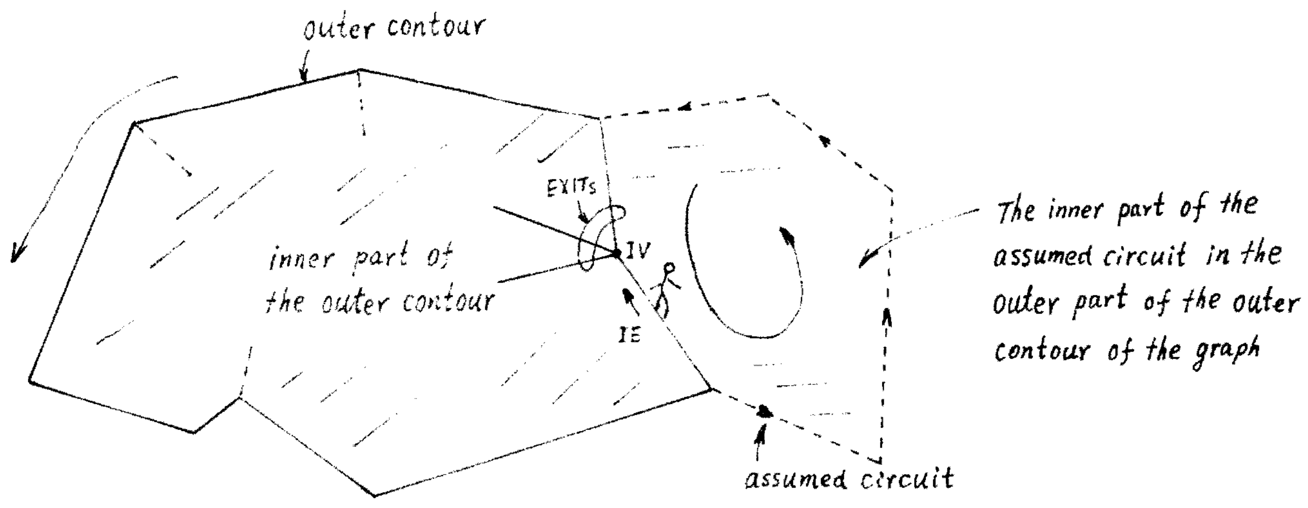

outer contour
EXITs
IV
IE
inner part of the outer contour
The inner part of the assumed circuit in the outer part of the outer contour of the graph
assumed circuit

Fig. 3.9.3.4

arbitrary circuit, say $V_iV_{i+1}V_{i+2}\ldots\ldots V_m$. The basic path line $V_1V_2\ldots V_iV_{i+1}\ldots V_m$ can be turned out during the path transfering process. Therefore, according to property 5, the circuit which does not contain IV is contained in PTL.

2. any circuit which contains IV starts either from an EXIT at IV or from the edge in the opposite direction of IE at IV. However, according to proposition 2, the circuits starting from the edge in the opposite direction of IE at IV are clockwise, i.e., they are undesired. Furthermore, according to property 5, any circuit starting from an EXIT of IV is contained in PTL, so that any desired circuit starting from an EXIT of IV is contained in PTL.

3. Summarising the above two caseses, we conclude that the desired circuits are all contained in PTL.

The sufficiency of theorem 2 has been proved. Now we are going to prove the necessity of the conditions.

4. RLCC CODING PROCESS

4.1 Convex vertex and concave vertex in a circuit

We have described in Sec. 2 that the RLCC code is a kind of circuit shape code, each number in a RLCC code represents the length of a convex vertex string or the length of a concave vertex string alternatively around a circuit. Hence, how to identify whether a vertex is convex or covcave is a problem which has to be studied at first.

The circuit to be encoded is traced in the counterclockwise direction. According to the definition of angle sign (see Sec. 3), at each convex vertex the associated deflective angle DA is negetive, and at each concave vertex the associated deflective angle DA is positive (see Fig.4.1.1). Therefore, The '+' sign strings and the '-' sign strings of DA are equivalent to the convex strings and concave strings of vertices in a circuit, respectively. For each vertex around a circuit the related DA value has been recorded in the circuit sequence list, (see List 3.5.2).

4.2 Starting point for encoding

We have mentioned that the first number in a RLCC code has to represent the length of a convex string in a circuit.

Def. 4.2.1 SP:

The starting point of a convex vertex string.

Example 4.2.1 :

In Fig. 4.1.1, the circuit has two convex vertex strings: $V_1V_2V_3V_4V_5$ and

Def. 4.3.1 Code Value (CV):

It is the weighted summation of the numbers in a code.

$$CV = \sum_{i=1}^{N} 2^{N-i} * a_i \qquad (4.3.1)$$

where N is the length of the code,and

a_i is the number in the code which is located in the i-th position.

Example 4.3.1:

$CV\ (5241)=2^{4-1}*5+2^{4-2}*2+2^{4-3}*4+2^{4-4}*1=57;$

$CV(4152)=2^{4-1}*4+2^{4-2}*1+2^{4-3}*5+2^{4-4}*2=48.$

Def. 4.3.2 Key Point (KP):

It is a vertex selected from the n SP_s in a circuit as a real starting point for the encoding such that the code assotiated with the point has the largest code value(CV).

According to Def. 4.3.1 and Def. 4.3.2, we should, at first, encode out n codes assotiated with the n SP_s, then get n code values CV_1, CV_2,......,CV_n, and then select the key point.

Def. 4.3.3 NSS: The number of symmetrical sections of a circuit:

It is the number of periodic sections around a circuit, i.e., the number of the key points in a circuit.

If a circuit has only one key point(i.e.,its NSS=1) , we call it an asymmetrical circuit, otherwise, it is a symmetrical circuit (with NSS>1). When there are more than one key point in a circuit, we use the subscripts of key points to distinguish them. Despite what the NSS equals to, with any one of the key points as the real starting point for the encoding, a circuit will correspondes a RLCC code uniquely.

LSS=17.

For circuit (b):

$SP_1=V_1$, related code: 5, related CV: 5;

$SP_2=V_2$, related code: 5, related CV: 5;

$SP_3=V_3$, related code: 5, related CV: 5;

$SP_4=V_4$, related code: 5, related CV: 5;

$SP_5=V_5$, related code: 5, related CV: 5;

$KP_1=V_1$, $KP_2=V_2$, $KP_3=V_3$, $KP_4=V_4$,$KP_5=V_5$;

Normalized RLCC code: 5;

NSS=5;

ONK=0;

LSS=1.

For circuit (c):

$SP_1=V_6$, related code: 22422242, related CV: 580;

$SP_2=V_{10}$, related code: 42224222, related CV: 782;

$SP_3=V_{16}$, related code: 22422242, related CV: 580;

$SP_4=V_{22}$, related code: 42224222, related CV: 782;

$KP_1=V_{10}$, $KP_2=V_{20}$;

Normalized RLCC code: 42224222;

NSS=2;

ONK=9;

LSS=10.

4.4 Encoding process

alternatively the length of each string of negative DA and the length of each string of positive DA around the circuit starting from SP_1.

Step 3.

Generating all the codes related with each SP individually. According to the circular structure of a circuit, the different codes for a circuit can be deduced from the code encoded in step 2, through even times of left circular shifts. For example, for the circuit in Fig. 4.3.4, the code assotiated with SP_2: 712223, and the code assotiated with SP_3: 222371 can be generated through 2 steps and 4 steps of left circular shift of the code 237122 respectively.

Step 4.

Calculating the code values of all the codes (assotiated with all the SP_s).

Step 5.

Comparing all the code values, then finding out all the key points, which are assotiated with the largest code values.

Step 6.

Recording the following encoding parameters for a circuit into the circuit global list:

1. ONK, see Def. 4.3.5;
2. Normalized RLCC code, see Def. 4.3.4;
3. NSS, see Def. 4.3.3.;
4. LSS, see Def. 4.3.6.

Notice, from now we will call the normalized RLCC code, simply, the RLCC code or code.

Catagory №	RLCC Code	NSS	LSS	Loops of Catagory

List 4.5.1

The classification process is indeed the process that reorganizes the partial contents of the circuit global list into a circuit category list according to the RLCC code in order that the matching process will be able to use the information conveniently.

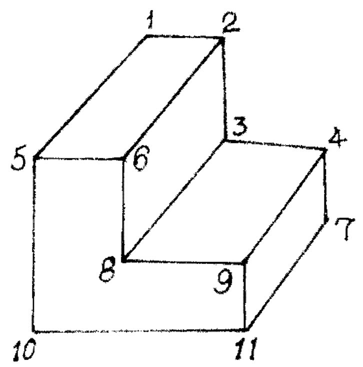


Fig. 4.4.1

represent those in graph G'.

5.2 Vertex pairs in a circuit pair (CP)

Before discussing what is the vertex pairs in a circuit pair, let us first look at the two examples below.

Example 5.2.1 :

Fig. 5.2.1 illustrates a circuit pair: Crt--Crt', Crt: $V_1V_2V_3V_4V_5V_6V_7V_8$, Crt': $V_2'V_3'V_4'V_5'V_6'V_7'V_8'V_1'$.

the key point (KP) of Crt is V_4, and the key point (KP') of Crt' is V_7', see Def. 4.3.2,

the related vertex pairs in the circuit pair are as follows:

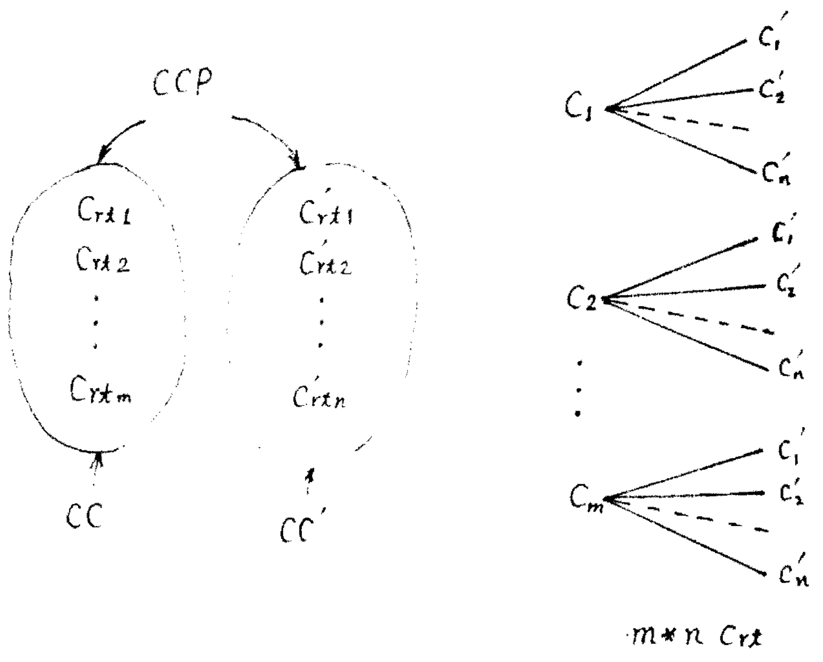


Fig. 5.1.1

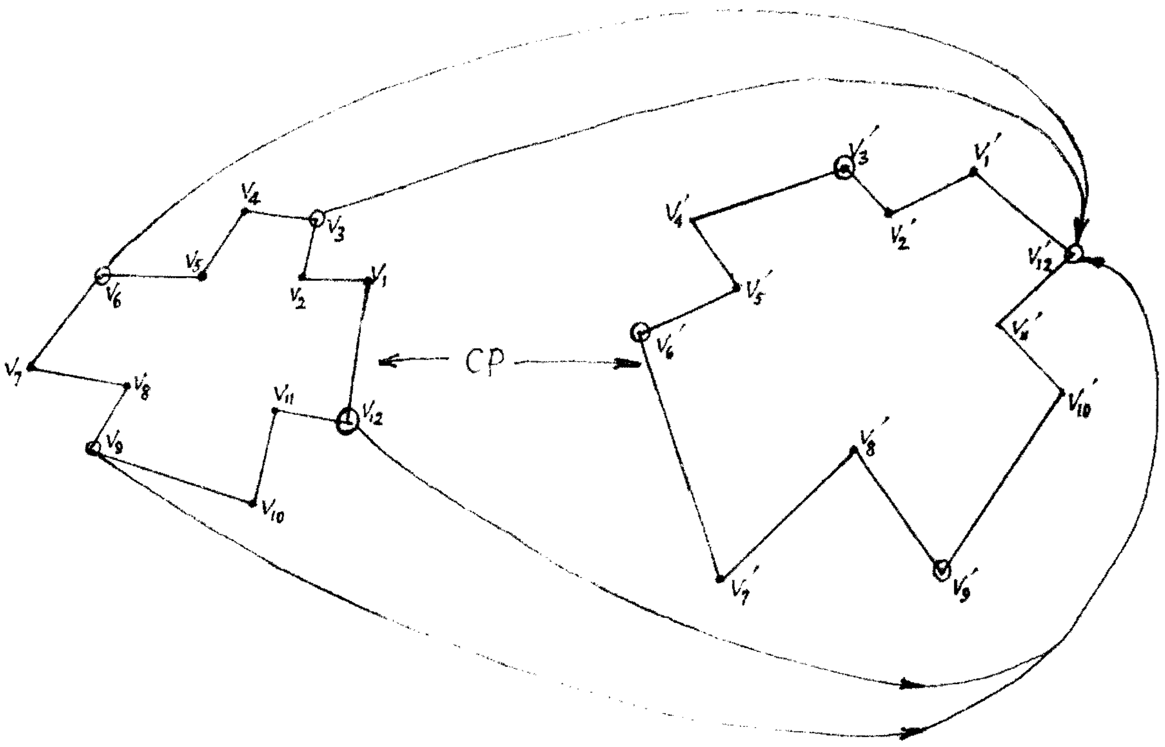


Fig. 5.2.2

4. V_{12}--V_{12}', V_1--V_1', V_2--V_2', V_3--V_3', V_4--V_4', V_5--V_5', V_6--V_6', V_7--V_7', V_8--V_8', V_9--V_9',V_{10}--V_{10}', V_{11}--V_{11}'.

Def. 5.2.1 CSVP:

It is the abbreviation of "Cyclic Set of Vertex Point pairs in a circuit pair".

In example 5.2.1, the vertex pairs enumerated are a CSVP in the circuit pair. However, in example 5.2.2, each group of vertex pairs enumerated is a CSVP in the circuit pair, so that there are four sets of CSVP altogether in the circuit pair.

Def. 5.2.2 SPP:

It is the abbreviation of "Synchronous Point Pairs in a circuit pair"

5.3 The clustering property of the true corresponding vertex pairs.

For a circuit in a graph there are no more than one circuits in another graph corresponding to it. If the circuit has N vertices then there are only N vertices in another graph corresponding to them, i.e., there is only a CSVP which is the real corresponding vertex pairs. Unfotunately, for a circuit in a CCP there will be n circuit pairs assotiated with it if the number of circuits in the other category is n (see Fig. 5.1.1). For a circuit pair there will be k CSVPs related to it (see example 5.2.2) if the NSS (number of symmetric sections) is k. So that for a circuit there will be n*k related CSVPs, but only one of them is possiblly the real corresponding vertex pair set, i.e., there are as many as n*k-1 CSVPs which are false corresponding vertex pair sets. Therefore, the problem of how to distinguish the real corresponding vertex pairs from the false ones is imperative for the matching process. For solving this problem,an important property about the real corresponding vertex pairs has to be utilized, which is that each real corresponding vertex pair will appear many times in different CPs of different CCPs. This property is called the clustering property.

Example 5.3.1

Fig. 5.3.1 illustrates a graph G which has a set of circuits from Crt_1 to Crt_{47}. Fig. 5.3.2 illustrates another graph G' which has a set of circuits from Crt_1' to Crt_{47}'. Both of the graphs are assotiated with a same moving object. The following statistic data illustrates the above property:

real corresponding vertex pairs	times of appearance
1--$2'$	32

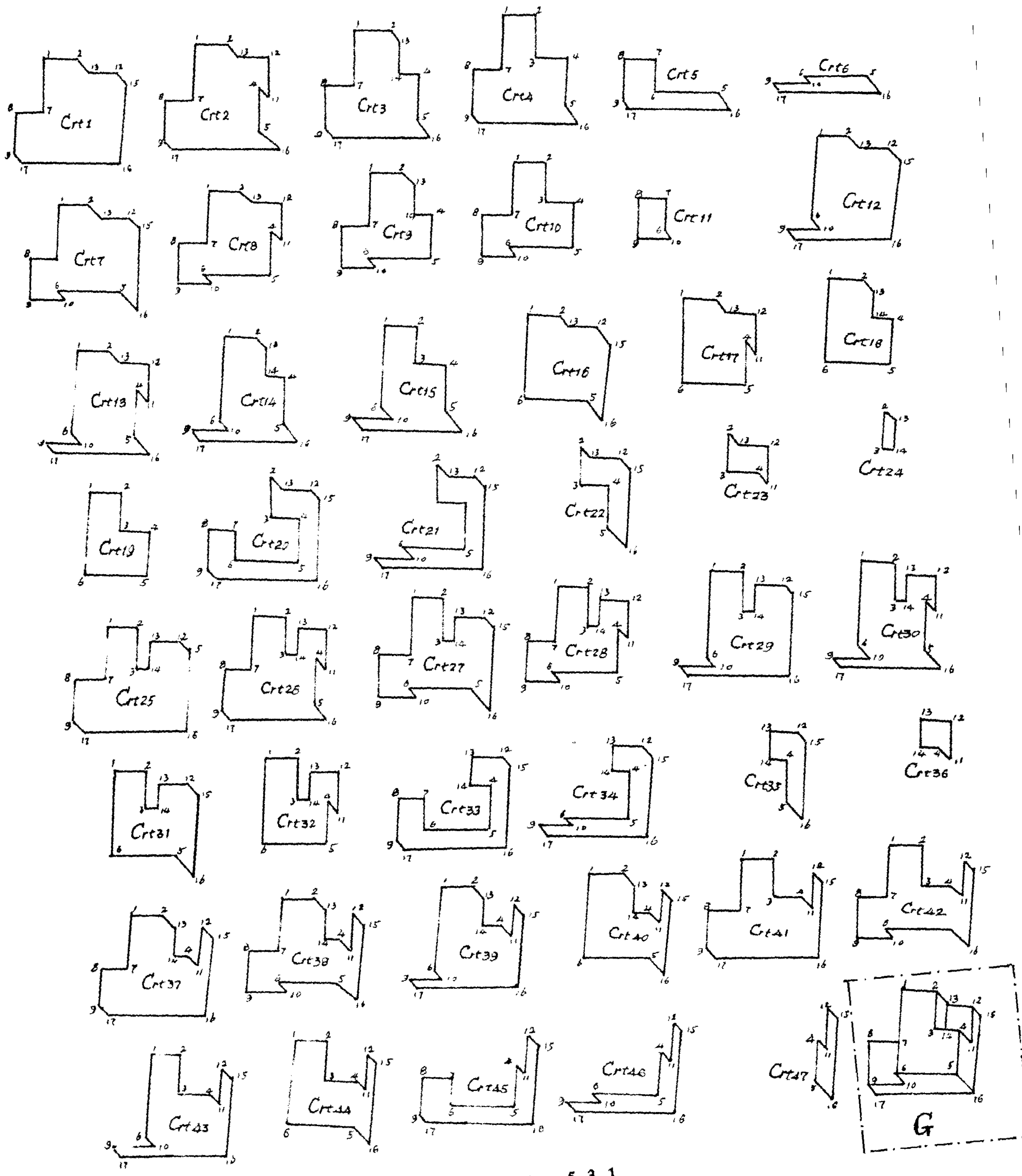

Crt1
Crt2
Crt3
Crt4
Crt5
Crt6
Crt7
Crt8
Crt9
Crt10
Crt11
Crt12
Crt13
Crt14
Crt15
Crt16
Crt17
Crt18
Crt19
Crt20
Crt21
Crt22
Crt23
Crt24
Crt25
Crt26
Crt27
Crt28
Crt29
Crt30
Crt31
Crt32
Crt33
Crt34
Crt35
Crt36
Crt37
Crt38
Crt39
Crt40
Crt41
Crt42
Crt43
Crt44
Crt45
Crt46
Crt47
G

Fig. 5.3.1

5.4 Matching table and decision list

The matching table (MT) (see List 5.4.1) is a table whose every row is assotiated with a vertex in a graph, say G, and whose every column is assotiated with a vertex in another graph, say G'. In MT, every intersection between a row and a column is an accumulator assotiated with the vertex pair between two graphs. During the matching process as soon as a vertex pair has been matched, the corresponding accumulator is incremented.

During the matching process, the values will be gradually concentrated at those accumulators in the MT which are assotiated with the real corresponding

V' V	1	2	3	4	.	.	.	.	.	N'
1										
2										
3										
4										
.										
.										
.										
.										
.										
M										

List 5.4.1

V	V'	RCD

List 5.4.2

3. Again, for each row of MT and of DL:

   If all te values of the accumulators in the row of MT are zero, then setting the second and the third columns of DL to zero;

4. Setting all the accumulators of MT to zero.

Example 5.4.1 :

Fig. 5.4.1(a) is a MT before the decision process, Fig. 5.4.1(b) is the MT after step 3 of the decision process, and Fig. 5.4.1(c) is the DL after the decision process.

After the decision process, the DL gives the matching result. In each row of DL, the vertex label of G and the vertex label of G$'$ indicated by the numbers in the first column and the second column of the list respectively, are

v \ v'	1	2	3	4	5	6
1	2.3	1.2	3.6	1.1	11.6	2.1
2	2.1	1.1	2.2	3.4	5.6	1.5
3	0	0	0	0	0	0
4	0	1.4	15.1	2.3	0	0
5	1.3	0	1.0	8.2	2	3
6	9	0	0	0	0	0

(a)

v \ v'	1	2	3	4	5	6
1	0	0	0	0	11.6	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	15.1	0	0	0
5	0	0	0	8.2	0	0
6	0	0	0	0	0	0

(b)

V	V$'$	RCD
1	5	3.4
2	0	0
3	0	0
4	3	6.6
5	4	2.7
6	1	inf.

(c)

Fig. 5.4.1

from the vertex label of G' in the vertex pair of the CSVP.

5.6 Weighted increment

In Sec. 5.3, we have mentioned that for a circuit of G there will be n*k CSVPs assotiated with it, where n is the number of circuits of the CCP in G' and k is the NSS related with the CCP, but only one of them, possiblly, is the real one. In Sec. 5.5, we have also mentioned that each CSVP has to be checked to make sure whether it is consistant or not. Suppose for a circuit of G there are r CSVPs ($r<n*k$) which are proven to be consistant with DL, then for the r CSVPs we use a weighted increment, which equals to 1/r, to be added into each accumulator in MT , which is assotiated with a vertex pair in the r CSVPs, for the circuit. The reason is that for a circuit there is only, at most, one CSVP which is the real one. Clearly, the larger the value of r ,the less the propability of that each CSVP is the real one. So that the reciprocal of r, i.e. 1/r, reflects the probability of a vertex pair in the r CSVPs being the real one.

In the main matching cycle (see Sec.5.7), all the CSVPs assotiated with a circuit in graph G have to pass two periods, the consistency checking period and the accumulating period. A special counter named r--counter is used to count up the number of consistent CSVPs for a circuit in graph G. In the consistency checking period, each CSVP assotiated with the circuit in G has to be checked. If it is consistent, then the r--counter will be added by 1. After all the CSVPs assotiated with the circuit have been checked, the number in the r--counter is just the number of the consistent CSVPs for the circuit, say r.

$\boxed{i}$	block scheme i, the content of which is illustrated in $\boxed{i}$, where i=1,2 ;
[0]	the operation: setting initial state, i.e., set MT, DL, r--counter, c--counter to zero (see Sec. 5.8);
[3]	the consistency check process for a CSVP, see Sec. 5.5;
[4]	the accumulating process: adding a weighted increment 1/r to an accumulator of MT assotiated with each of all the vertex pairs of the consistent CSVP, where r is the number in r--counter,see Sec. 5.4 and Sec. 5.6;
[5]	the operation: setting r--counter to zero;
[6]	the decision process, see Sec. 5.4;
[7]	the terminal state decision process, see Sec. 5.8;
[8]	the operation: print out the final matching result;
{ G G' }	the set of two graphs, G and G';
{CCP CCP...CCP}	the set of CCP of the two graphs, in each CCP of the set

The Flowchart of the Algorithm

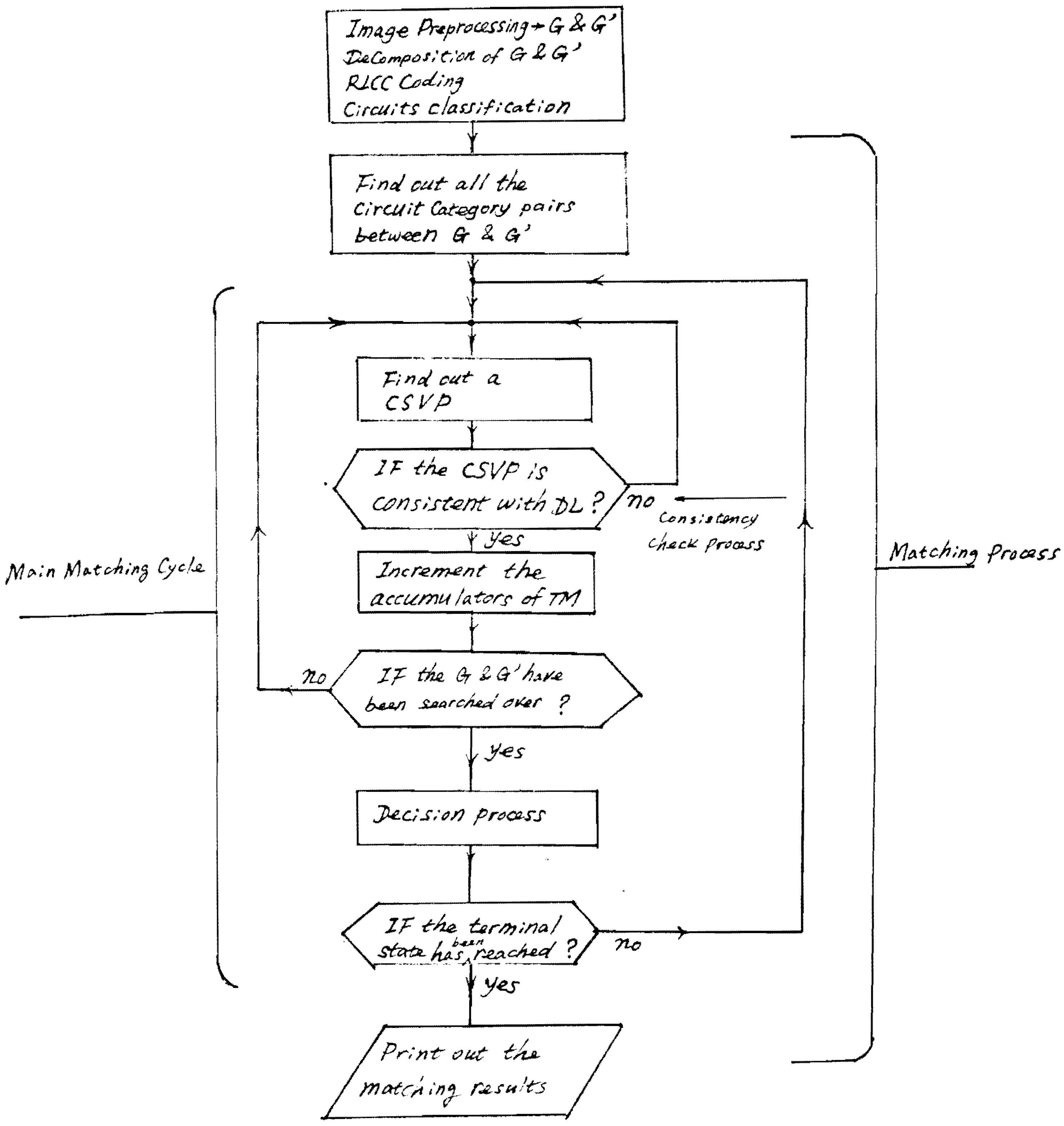


Image Preprocessing → G & G'
DeComposition of G & G'
RLCC Coding
Circuits classification
Find out all the circuit Category pairs between G & G'
Find out a CSVP
IF the CSVP is consistent with DL ?
no
Consistency check process
yes
Increment the accumulators of TM
IF the G & G' have been searched over ?
no
yes
Decision process
IF the terminal state has been reached ?
no
yes
Print out the matching results
Main Matching Cycle
Matching Process

Fig 5.7.1

5.8 Terminal state of the MMC

As we know, the main matching cycle (MMC) operates recursively. At each end of a cycle the decision list DL is modified once. Because the matching decision process, see Sec. 5, guarantees the selection of vertex pairs with high clustering degree from MT and put them into DL, the vertex pairs represented in DL enjoy higher confidence. Furthermore, during the next main matching cycle, because of the consistency check process for each CSVP, the inconsistent CSVPs are discarded. Therefore, the value distribution in MT is concentrated in those accumulators which are assotiated with the real vertex pairs between the two graphs.

Both the matching decision process and the consistency check process make the set of vertex pairs represented by DL, gradually, approximate the real corresponding vertex pairs between the two graphs as the MMC operates.

What is the terminal state of MMC ? When one of the following two cases has occured, we say that the terminal state has been reached and MMC is stoped at once.

1. After the decision process, the content of DL is not changed at all, i.e., the result of MMC has converged.
2. After N_c cycles of MMC, no matter whether the result of MMC has converged or not, where N_c is a number which has been selected beforehand to prevent the program from running too long.

It is clear that in order to decide whether the MMC has converged, a buffer is necessary to accept the old content of DL before the matching decision process, and in order to decide whether the MMC has undergone N_c cycles,

6 EXPERIMENTAL RESULTS

To test our algorithm, we did several experiments, in which some real situations were considered such as shape distortions, scaling, rotation, dissimilarities, topological change and T--joints.

We did the experiments at Vax-780 with PASCAL (PC Compiler).

Experiment 1

Image 1 and Image 1$'$ (see pp. 85) are two consecutive images of a moving object. From Image 1 and Image 1$'$, we can see that there are obvious differences between the two images. The differences are caused by three-dimensional motion and scaling. G1 and G2 are the vertex connection line graphs of Image 1 and Image 1$'$, respectively. From G1, we can see that edge 2--11 and edge 5--8 were lost because of the low contrast. The results of experiment 1 (see pp. 89) illustrate that corresponding points found by our matching procedure are completely correct although there existed considerable three-dimensional translation and rotation, scaling and dissimilarites (caused by missing edges). The algorithm can distinguish T--joint pairs between two graphs automatically. In two consecutive graphs of a moving object, a vertex pair will be recognized to be a T--joint pair if both of the vertices in their graphs are T--joints , i.e., each of them lies in a circuit in a graph, and the deflective angle of the circuit graph at the vertex is approximately zero (see Fig. 6.1). The T--joint pairs may be false corresponding vertex pairs between two graphs because the T--joint sometimes do not represent real vertices of the objects. For instance, a T--joint can be an occluding point in a caused by illusory lines. The results of experiment 2 illustrate that false vertices 19 and 18 in G2 can not find their corresponding vertices in $G'2$.

Experiment 3

The CPU time used in experiment 3 by the algorithm is much less than these in the other experiments. The reason is that the total numbers of circuits of G3 and $G'3$ are much fewer than the circuit numbers in the other experiments.

Experiment 4

In experiment 4, we try to match partial graph of one of the two consecutive graphs with the other. The images contain shadows. From experiment 4, we draw two conclusions:

(1) The algorithm still works well when the degree of dissimilarities between two consecutive graphs is very significant. In other words, missing some edges and vertices in the process of detecting edges and vertices does not affect correct matching.

(2) Shadows do not result in wrong matchings if they do not give rise to similar edges in the two graphs.

Experiment 5

In experiment 5, the scales of the two graphs are quite different and there are complicated shadows caused by multiple light sources in Image $5'$. The results of experiment 5 illustrate that the algorithm can tolerate complicated shadows and significant scaling.

Experiment 6

In experiment 6, there is topological change between Image 6 and Image $6'$, i.e., T--joint vertex 10 in G6 became T--joint vertex 20 in $G'6$ and

Experiment 1.

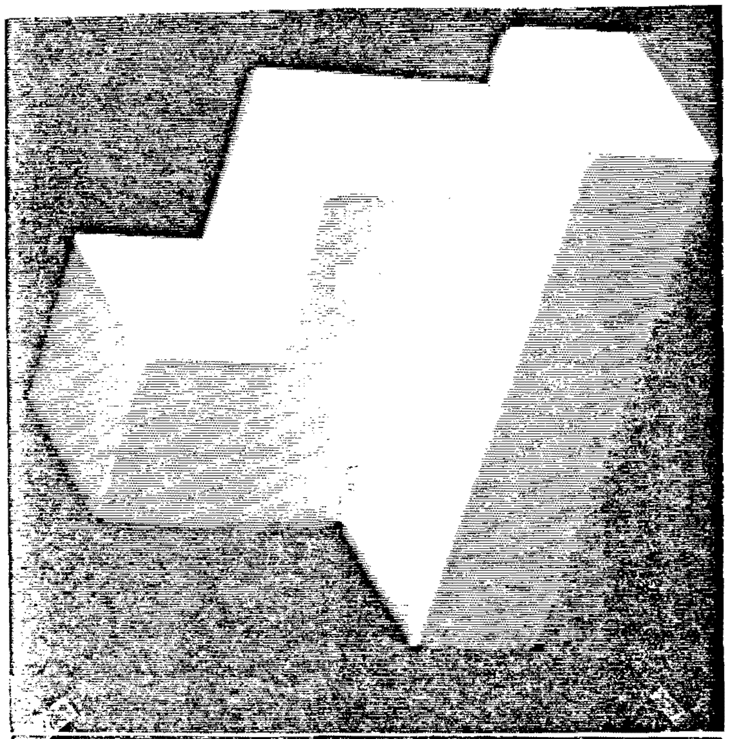


Image 1

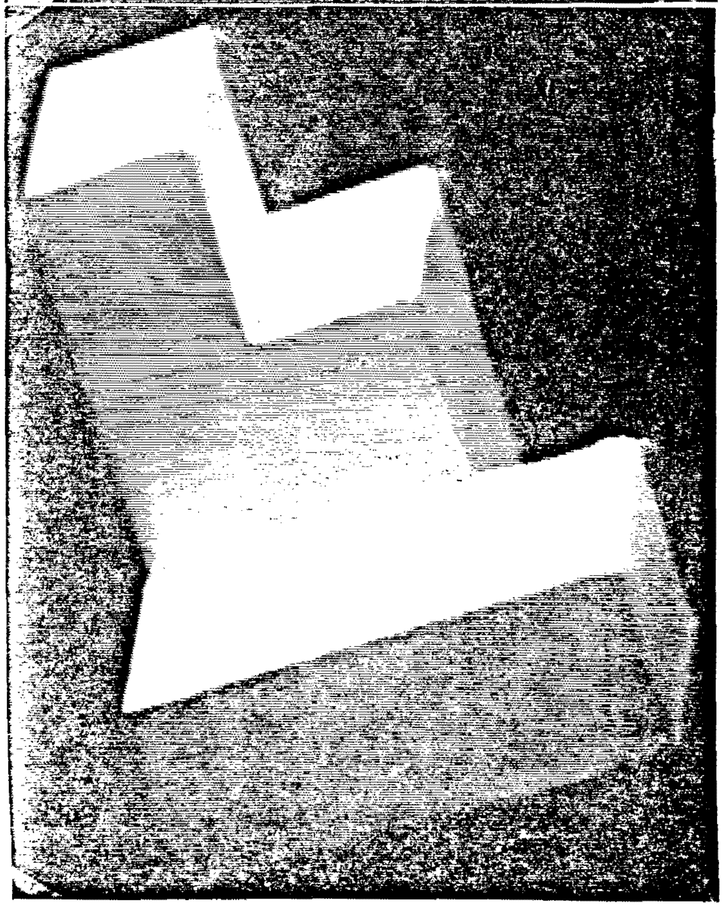


Image 1'

Coordinate List (CL) of G1

V	X	Y
1	187	202
2	242	206
3	264	138
4	363	146
5	370	124
6	424	126
7	457	174
8	406	172
9	347	189
10	294	186
11	271	255
12	217	253
13	167	268
14	198	322
15	300	321
16	328	372
17	379	371

Vertex Connection List (VCL) of G1

V	Connected with		
1	2	12	13
2	1	3	
3	2	4	10
4	3	5	9
5	4	6	
6	5	7	
7	6	8	17
8	7	16	
9	4	10	15
10	3	9	11
11	10	12	
12	1	11	14
13	1	14	
14	12	13	15
15	9	14	16
16	8	15	17
17	7	16	

Experiment 1. Input data of G1

Results of experiment 1

Number of circuits of G1: 39.

Number of circuit categories of G1: 34.

Number of circuits of $G'1$: 113.

Number of circuit categories of $G'1$: 89.

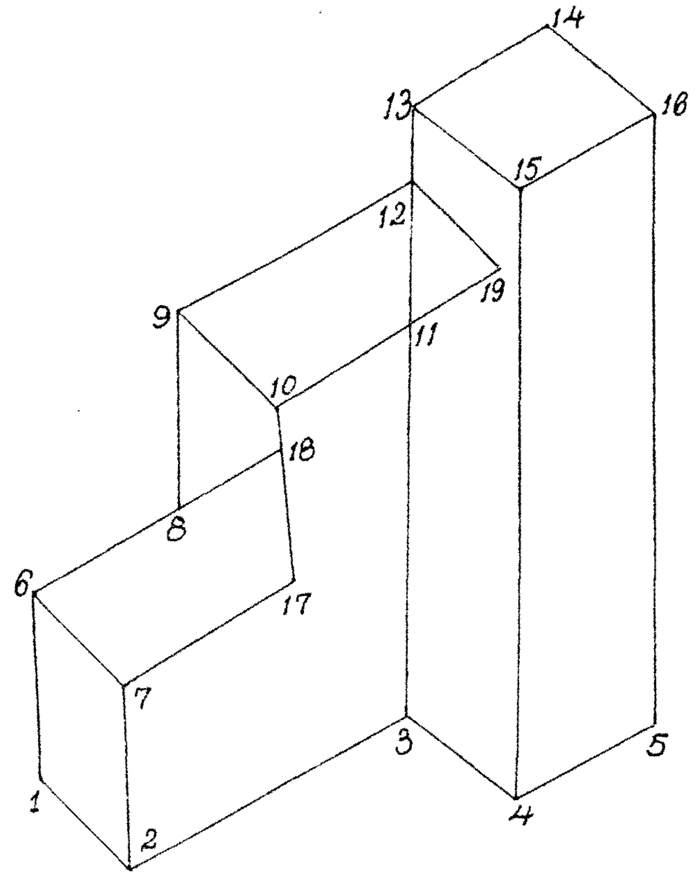
Number of category pairs between two graphs: 34.

Number of main matching cycle: 3.

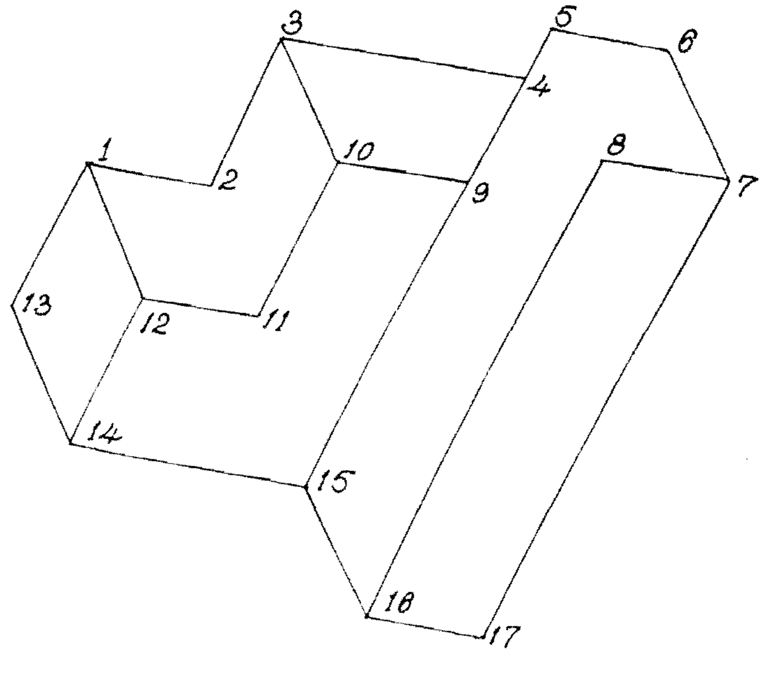
CPU time: 5.2 seconds.

Matching result:

V of G1	(corresponding to)	V' of $G'1$	*
1		11	
2		8	
3		7	
4		17	T--j
5		3	
6		16	
7		15	
8		4	
9		2	T--j
10		6	
11		9	
12		10	
13		12	
14		13	
15		1	
16		5	
17		14	


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

G2


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

G'2

CL of $G'2$

V'	X	Y
1	187	202
2	242	206
3	264	138
4	363	146
5	370	124
6	424	126
7	457	174
8	406	172
9	347	189
10	294	186
11	271	255
12	217	253
13	167	268
14	198	322
15	300	321
16	328	372
17	379	371

VCL of $G'2$

V'	Connected with		
1	2	12	13
2	1	3	
3	2	4	10
4	3	5	9
5	4	6	
6	5	7	
7	6	8	17
8	7	16	
9	4	10	15
10	3	9	11
11	10	12	
12	1	11	14
13	1	14	
14	12	13	15
15	9	14	16
16	8	15	17
17	7	16	

Experiment 2. Input data of $G'2$

Experiment 3.

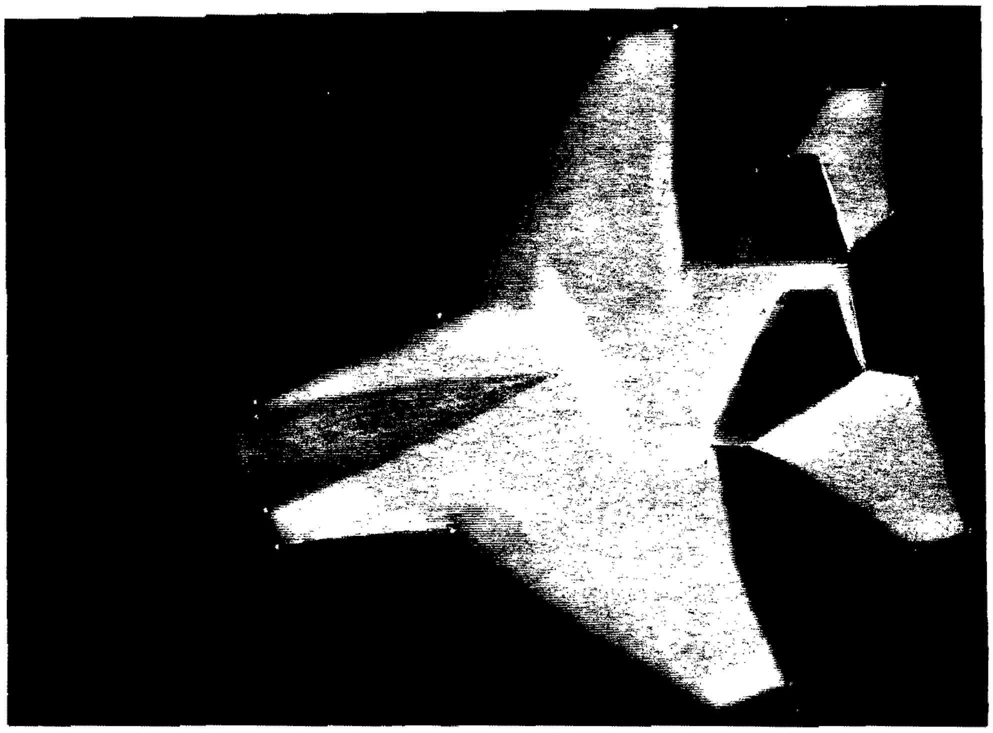


Image 3

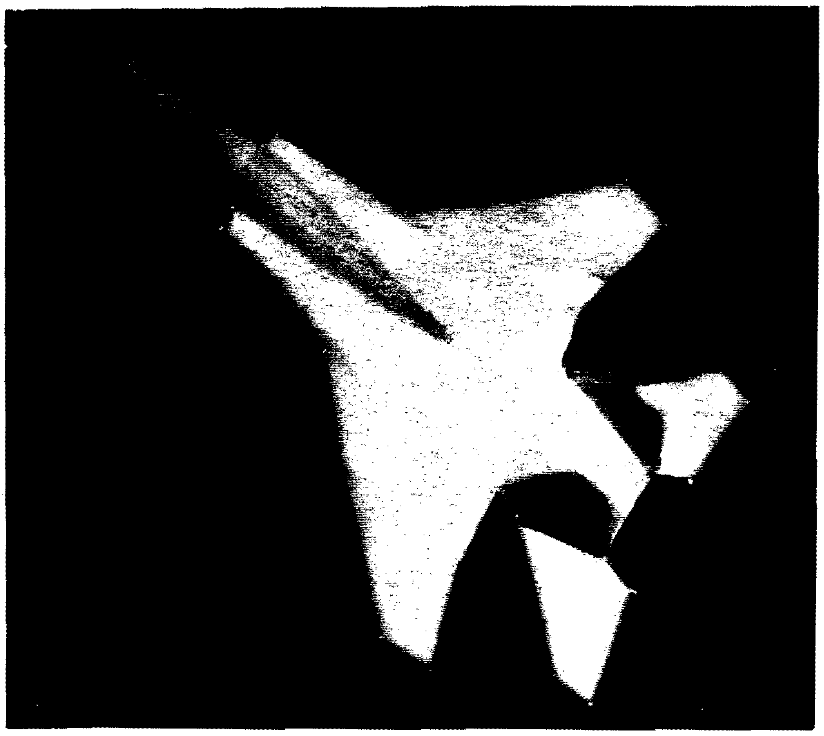


Image 3'

CL of G3

V	X	Y
1	88	332
2	186	279
3	186	271
4	262	239
5	340	126
6	367	122
7	365	184
8	372	217
9	401	180
10	413	178
11	425	176
12	433	147
13	454	147
14	456	199
15	438	214
16	443	259
17	464	263
18	487	327
19	465	334
20	399	293
21	383	297
22	390	340
23	410	388
24	379	401
25	269	326
26	193	333
27	192	317
28	320	261
29	408	237
30	433	228

VCL of G3

V	Connected with			
1	2	27		
2	1	3	28	
3	2	4		
4	3	5		
5	4	6		
6	5	7		
7	6	8		
8	7	9	15	
9	8	10		
10	9	11	12	
11	10	15		
12	10	13		
13	12	14		
14	13	15		
15	8	11	14	16
16	15	17	20	30
17	16	18		
18	17	19		
19	18	20		
20	16	19	21	
21	22	20	29	
22	21	23		
23	22	24		
24	23	25		
25	24	26		
26	25	27		
27	1	26	28	
28	2	27		
29	21	30		
30	16	29		

Experiment 3. Input data of G3

<u>Results of experiment 3</u>

Number of circuits of G3: 25.

Number of circuit categories of G3: 23.

Number of circuits of $G'3$: 25.

Number of circuit categories of $G'3$: 23.

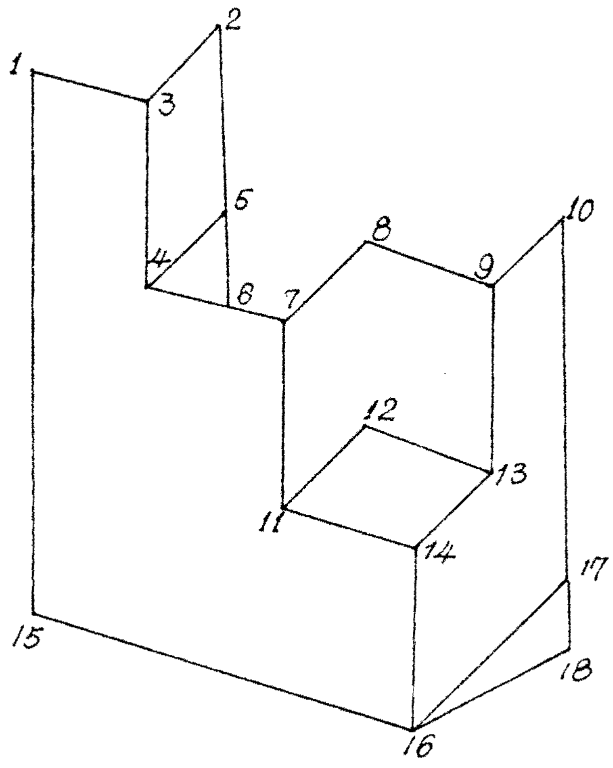
Number of category pairs between two graphs: 23.

Number of main matching cycles: 3.

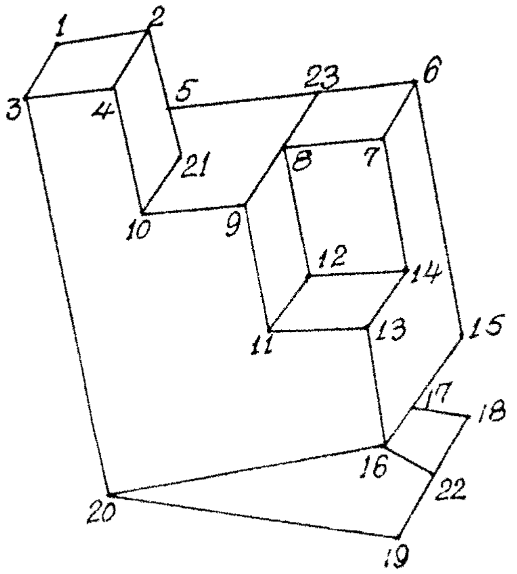
CPU time: 1.5 seconds.

Matching result:

V of G3 (corresponding to)	V' of $G'3$	*
1	30	
2	13	
3	14	
4	5	
5	15	
6	16	
7	17	
8	2	
9	20	
10	19	T--j
11	21	
12	9	
13	10	
14	23	
15	22	
16	25	
17	24	
18	8	
19	7	
20	4	
21	3	
22	18	
23	29	
24	28	
25	6	
26	11	
27	12	
28	1	
29	26	
30	27	


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

G4--1


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

G'4--1

CL of G' 4--1

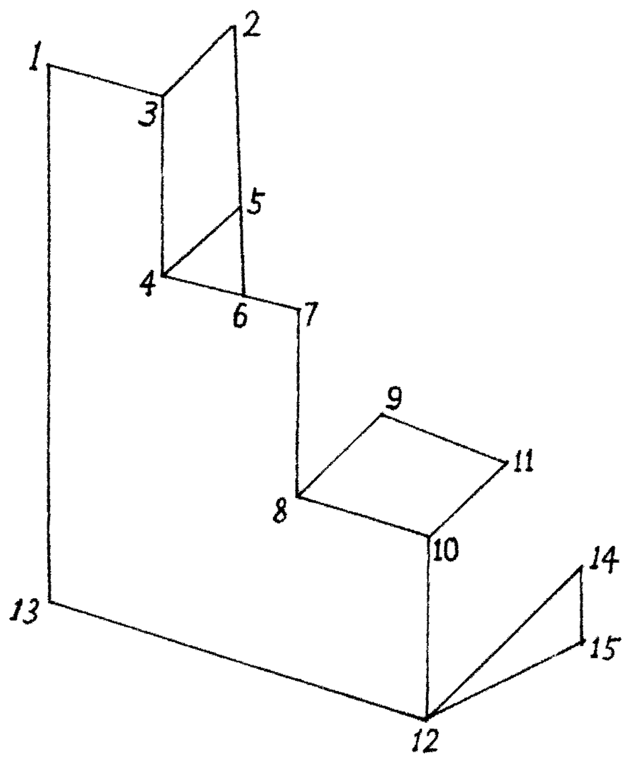
V'	X	Y
1	206	165
2	246	157
3	191	190
4	232	181
5	255	188
6	360	168
7	345	193
8	304	202
9	289	225
10	247	235
11	304	280
12	319	254
13	345	272
14	361	246
15	387	270
16	358	323
17	369	303
18	380	303
19	358	356
20	232	344
21	262	208
22	369	330
23	308	178

VCL of G' 4--1

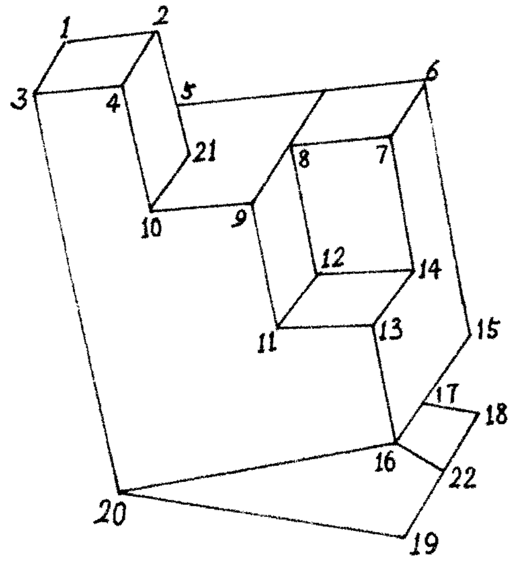
V'	Connected with			
1	2	3		
2	1	4	5	
3	1	20	4	
4	2	3	10	
5	23	2	21	
6	23	7	15	
7	8	14	6	
8	7	23	9	12
9	8	10	11	
10	21	4	9	
11	12	9	13	
12	14	8	11	
13	14	11	16	
14	7	12	13	
15	6	17		
16	17	13	20	22
17	15	16	18	
18	17	22		
19	22	20		
20	16	3	19	
21	5	10		
22	18	16	19	
23	5	8	6	

Experiment 4--1. Input data of G' 4--1

Experiment 4--2.



G4--2



G'4--1

Results of experiment 4--2

Number of circuits of G4--2: 13.

Number of circuit categories of G4--2: 10.

Number of circuits of G'4--1: 747.

Number of circuit categories of G'4--1: 560.

Number of category pairs between two graphs: 8.

Number of main matching cycles: 3.

CPU time: 20 seconds.

Matching result:

V of G4--2 (corresponding to)	V' of G'4--1
1	3
2	2
3	4
4	10
5	21
7	9
8	11
9	12
10	13
11	14
12	16
13	20

CL of G4--3

V	X	Y
1	301	234
2	332	198
3	390	210
4	416	174
5	310	307
6	341	273
7	397	282
8	366	319
9	376	389
10	432	320
11	438	357

VCL of G4--3

V	Connected with		
1	2	5	
2	1	3	
3	2	4	7
4	3	10	
5	1	6	8
6	5	7	
7	3	6	8
8	5	7	9
9	8	10	11
10	4	9	11
11	9	10	

Experiment 4--3. Input data of G4--3

<u>Experiment</u> <u>5</u>.

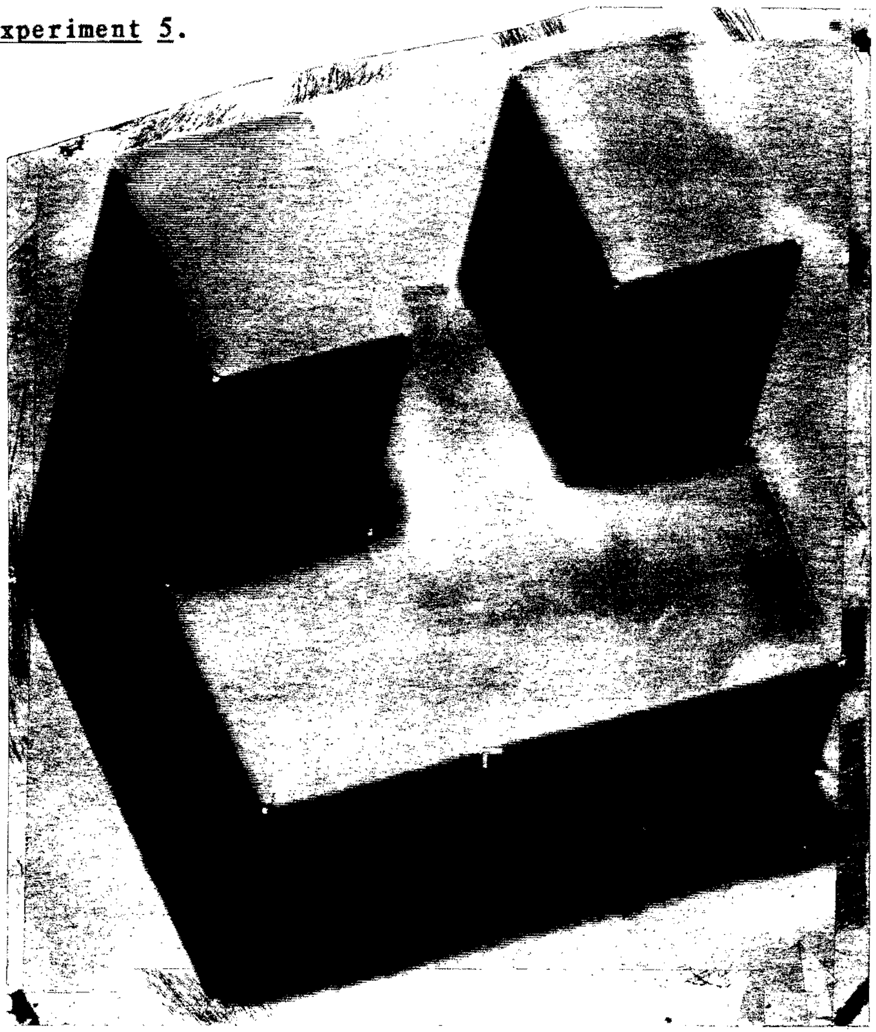


Image 5

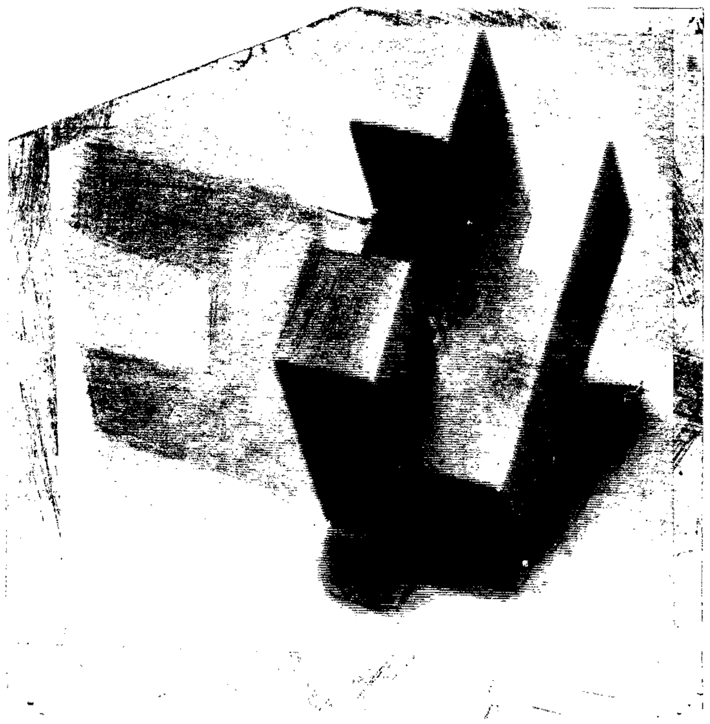


Image 5'

CL of G5

V	X	Y
1	200	374
2	156	280
3	403	223
4	323	242
5	408	234
6	440	310
7	180	220
8	261	180
9	340	160
10	428	142
11	385	58
12	307	77
13	281	159
14	254	165
15	219	93
16	138	111
17	85	280
18	176	452
19	430	390
20	440	400
21	434	359
22	255	270
23	248	233

VCL of G5

V	Connected with			
1	6	2	18	
2	1	22	7	
3	5	10		
4	5	9	13	
5	6	3	4	
6	21	5	1	
7	2	8	16	
8	23	14	7	
9	4	10	12	
10	3	11	9	
11	10	12		
12	11	13	9	
13	4	12	14	
14	8	13	15	
15	14	16		
16	7	15	17	
17	16	18		
18	20	19	1	17
19	18	21		
20	18	21		
21	20	6	19	
22	23	2		
23	8	22		

Experiment 5. Input data of G5

<u>Results of experiment 5</u>.

Number of circuits of G5: 172. Number of circuit categories of G5: 153.

Number of circuits of $G'5$:480. Number of circuit categories of $G'5$:462.

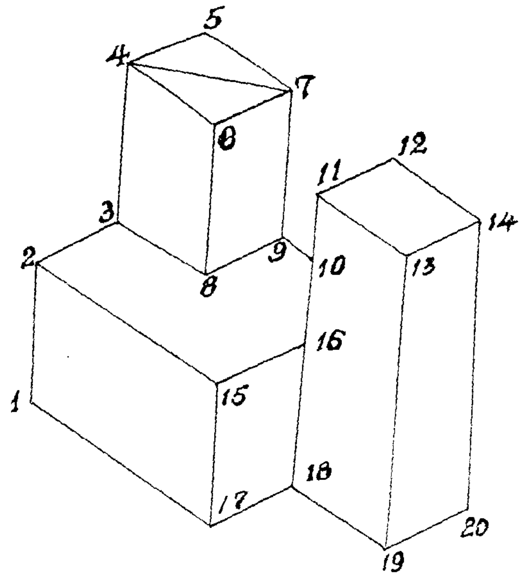
Number of category pairs between two graphs: 50.

Number of main matching cycles: 3.

CPU time: 23 seconds.

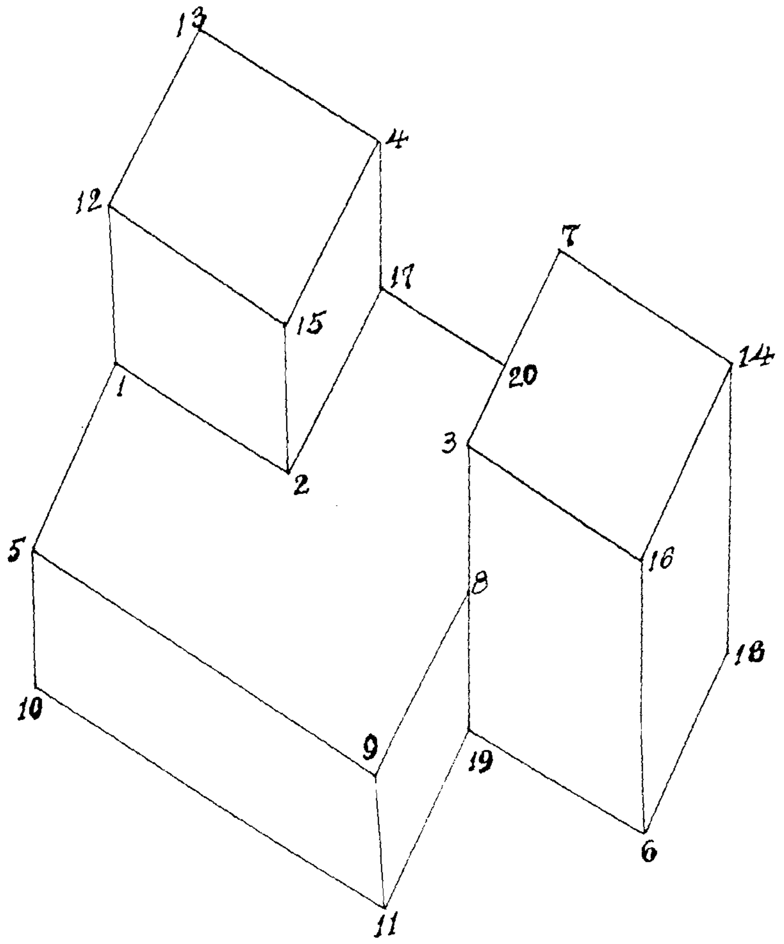
Matching result:

V of G5 (corresponding to)	V' of $G'5$	*
1	1	
2	18	
3	6	
4	5	
6	4	
7	16	
8	14	
9	7	
10	8	
11	9	
12	10	
13	11	
14	12	T--j
15	13	
16	15	
17	19	
18	2	
19	3	



4
5
7
6
3
2
8
9
11
12
14
10
13
16
1
15
18
17
20
19

G6



13
4
12
7
17
15
20
14
1
3
2
5
16
8
18
10
9
19
6
11

G'6

CL Of $G'6$

V'	X	Y
1	173	176
2	240	229
3	316	221
4	289	97
5	128	248
6	373	388
7	362	149
8	310	281
9	265	354
10	126	304
11	259	406
12	175	115
13	220	46
14	433	202
15	245	168
16	389	273
17	285	156
18	417	316
19	305	335
20	335	194

VCL of $G'6$

V'	Connected with		
1	2	5	12
2	1	15	17
3	8	16	20
4	13	15	17
5	1	9	10
6	16	18	19
7	14	20	
8	3	9	19
9	5	8	11
10	5	11	
11	9	10	19
12	1	13	15
12	4	12	
14	7	16	18
15	2	4	12
16	3	6	14
17	2	4	20
18	6	14	
19	6	8	11
20	3	7	17

Experiment 6. Input data of $G'6$

ACKNOWLEDGEMENT

This work was supported by the National Science Foundation Grant: MCS 82- 06926

[10] W. K. Gu, T. Y. Tang and T. S. Huang, "forming a connected graph from an image of an object" TR. of CSL. ,University of Illinoise at Urbarna Champiagn ,1983.

[11] W. K. Pratt, "Digital Image Processing," Wiley, 1978.

[12] A. Rosenfeld and A. C. Kak, "Digital Picture Processing," second edition, Academic Press, 1982.

[13] R. J. Wilson, "Introduction to Graph Theory," Academic Press, 1972.

[14] W. Mayeda, "Graph Theory," Wiley, 1972.

[15] J. K. Cheng and T. S. Huang, "Image recognition by matching relational structures," proc. IEEE Conf. PRIP, Aug. 3-5, 1981, Dallas, Texas.

[16] J. K. Cheng and T. S. Huang, "Algorithm for matching relational structures and their applications to image processing," Tech. Rep., School of EE, Purdue University, 1981.

[17] A. Rosenfeld, R. A. Hummel and S. W. Zucker, "Sence labeling by relaxation operations," IEEE Trans. on systems, man, and cybernetics, pp. 420-431, June 1976.

[18] C. J. Jacobus, R. T. Chien, and J. M. Selander, "Motion detection and analysis of matching graphs of intermediate level primitives," IEEE T-PAMI, pp. 495-510, Nov. 1980.