그림조각 맞추기에 관한 연구

°이 동 주*, 서 일 홍*, 오 상 록**
* 한양대학교 전자공학과, ** 한국과학기술원 전기제어 연구실

A Study on the Jig - Saw Puzzle Matching

Dong-Joo Lee*, Il-Hong Suh* and Sang-Rok Oh**
*Dept. of Electronics Eng., Hanyang Univ. ** Power Controls Lab., KAIST.

ABSTRACT

A jig-saw puzzle matching technique is proposed. Specifically, the geometric patterns of the puzzle pieces are firstly extracted using a boundary tracking algorithm at low resolution. And then, features of the extracted pieces to describe jig-saw puzzle pieces such as angles and distances between corner points, and convexity or concavity of a corner point are obtained from some corner points implying discontinuity of curvature of puzzle pieces' boundary.

Finally, a boundary matching algorithm without a priori information of matched puzzle is proposed.

1. 서 론

사람이 그림조각 맞추기에 사용하는 정보및 그 조합은 상당히 복잡하며 직관적이라 할 수 있는데 그 모든 메카니즘을 컴퓨터로 프로그래밍하기란 거의 불가능하다. 그럼에도 불구하고 컴퓨터를 이용한 2차원 그림 맞추기에 대한 여러 방법이 제시되어 있는데 모두가 각각의 그림조각들의 외형선으로부터 정보를 추출하여 수행하는 Boundary Matching 방법이다[1,2].

본 논문에서는 Chain-Code 로 구한 외형선에서 물체를 설명하는 특징을 구하여 조합을 수행하는 Boundary Matching 방법에 의하여 CCD 카메라로 받은 그림조각을 맞추는 알고리즘을 제안하기로 한다. 특히 사전정보(Pre-information) 없이 그림조각들을 맞추기 위하여 한 조각에 다른 조각이 맞는다고 판단될때까지 계속 맞추는 방법을 사용하였다. 이 방법은 실제로 오차가 많은 경우에는 사전정보를 알고 수행하는 경우보다는 덜 유용하지만 그림조각의 모양에 가해지는 제약이 적다는 점에서 효과적이라 할 수 있다.

실험대상인 그림조각들은 Back-Tracking 없이 맞추어질 수 있는 유일성(Uniqueness)을 갖도록 제한한다. 이 유일성은 그림조각들이 한 가지 방법만으로 알맞게 맞추어진다는 것을 의미하는데 그림조각의 기하학적 특성과 잡음에 관련된다. Back-Tracking 을 제외시키는 것은 문제에 대한 충분한 이해가 없이는 그림조각을 맞추어나가는 프로그램의 복잡성이 상당히 커지기 때문이다. 물론 그림조각들이 유일성을 지니고 있더라도 Back-Tracking 을 요구하는 경우가 있는데 이것은 그림조각을 맞추는 기준의 부정확성 또는 오차에 의한것으로 판단된다.

2. 전 처리 과정 (Pre-Processing)

본 장은 서론에서 언급한 컴퓨터에 의한 2차원 Jig - Saw Puzzle Matching 을 수행하기 위한 전단계로서 물체의 표현과 특징들을 구한다.

2.1 이진영상의 구성 및 Blob 의 추출

더욱 빠르고 간단한 영상의 해석을 위하여 그림조각(물체)은 흰색('255'Gray-Level)으로하고 그림조각이 놓여지는 바탕(배경)은 검은색('0' Gray-Level)으로하여 이진영상을 구성하였다.

이진영상에서 Blob 을 추출하기 위하여 Raster Scan 방식을 사용하였다[3]. Blob 의 경계점을 찾기위하여 왼쪽에서 오른쪽으로 일정한 갯수의 화소(Pixel)를 건너뛰면서 Scan 하고, 위에서 아래로는 몇 열을 건너뛰어 Scan 을 한다. 화소의 값이 '255' 이면 그 행에서 화소의 값이 '0' 에서 '255' (물체)로 변하는 화소를 찾아 경계점추적을 시작한다. 또 다른 Blob 을 찾기 위하여 이미 찾아진 Blob 에 속한 모든 화소의 값을 '0' 으로 바꾸어준 다음에 Raster Scan 을 계속한다.

2.2 경계점 추적 (Boundary Tracking)

이진영상에서 찾아낸 물체의 외형선을 추출하기 위하여 Chain Code Scheme 을 사용하는데 이것은 2 차원 영상의 정보를 가진 1 차원적인 묘사로서 이루어진다. 이러한 Chain Code 는 Freeman 에 의하여 폭 넓게 연구되어져 있다[4]. 본 논문은 그림 1. 의 8-방향 Chain coding Scheme 에 의하여 경계점 추적을 수행한다.

경계점 추적은 Blob 의 추출과정에서 선택한 시작점에서 시계방향으로 돌아가면서 추적하여 처음 시작한 경계점이 나올때까지 계속한다. 한 경계점에서 다음의 경계점을 찾기 위하여 지금의 경계점을 중심으로 화소의 값이 '0' 에서 '255'로 전이될때 '255' 의 값을 갖는 화소가 다음 경계점이 되며 이때의 방향이 벡터의 방향이 된다. 이처럼 다음의 경계점을 찾기위해 8 방향 화소를 조사해 나가는 것을 Local Tracking 이라 한다.

첫 경계점의 Local-Tracking 때에는 '0' 방향 화소는 배경에 속해있고 따라서 출발방향은 '1' 의 방향이 된다. 그 이외의 경계점에서의 Local Tracking 때에는 지난 Local Tracking 때에 이미 조사받은 화소들이 존재하므로 인접한 8 방향 모두 조사할 필요가 없다.

위의 Chain-Code 구성방법은 Tracking 하는데 걸리는 시간을 단축시키며, 하나의 화소로 구성된 부분에서도 Back Tracking 할 필요가 없어진다. Table 1 에서 이들의 관계를 설명하였다.

Table 1

전 벡터의 방향	출발방향
0	7
1	7
2	1
3	1
4	3
5	3
6	5
7	5

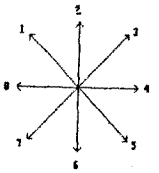


그림 1. 8-방향 Chain Coding Scheme

2.3 근사화 (Approximation) [5-8]

물체의 방향과 크기에 무관한 특징들을 구하기 위하여 Chain Code 로부터 곡률반경의 불연속성을 계산하여 구한 구석점(Coner Point)을 이용한다 [8]. 이러한 구석점들은 Chain 을 구성하는 s 개의 Line Segment 를 이동시키면서 찾는다.

주어진 n - Link Chain A^n 을 $A^n = C_{i=1}^{n} a_i = a_1\ a_2\ a_3\ \dots\ a_n$ 으로 나타내면 j 점을 끝점으로 S 개의 Link 를 갖는 Line Segment L_j^s 는 다음과 같이 표시된다.

$$L_j^s = C_{i=j-s+1}^{j} a_i \qquad j = 1,2,3,..,n\ ,\ n > s$$

L_j^s 의 X 와 Y 성분 X_j^s 와 Y_j^s 는 아래와 같다.

$$X_j^s = \sum_{i=j-s+1}^{j} a_{ix}\ , \qquad Y_j^s = \sum_{i=j-s+1}^{j} a_{iy}$$

여기서 a_{ix} 와 a_{iy} 는 각각 Chain Link a_i 의 X 와 Y 성분이며 a_{ix} , $a_{iy} \in \{-1, 0, 1\}$ 이다.

X 축에 대한 L_j^s 의 각도 Θ_j^s 는

$$\Theta_j^s = \begin{cases} \tan^{-1} Y_j^s / X_j^s & \text{if } |X_j^s| \geq |Y_j^s| \\ \cot^{-1} X_j^s / Y_j^s & \text{if } |X_j^s| < |Y_j^s| \end{cases}$$

로 주어진다. 이 Θ_j^s 로 부터 j 가 1 부터 n 까지 변할때의 Chain 의 곡률반경을 구하는데, 잡음의 영향을 감소시키기 위하여 Incremental Curvature 를 다음과 같이 정의한다.

$$\delta_j^s = 2 \cdot \frac{(\Theta_{j+1}^s - \Theta_j^s) + (\Theta_j^s - \Theta_{j-1}^s)}{2} = \Theta_{j+1}^s - \Theta_{j-1}^s$$

적당한 한계값 이내의 δ_j^s 값을 갖는 연속적인 Chain-node 들을 Discontinuity Free 영역 이라고 하며 다음과 같이 표시한다.

$t1 = \max \{ t \mid \delta_{j-v}^s \in (-\Delta, \Delta),\ 1 < v < t \}$
$t2 = \max \{ t \mid \delta_{j+v+1}^s \in (-\Delta, \Delta),\ 1 < v < t \}$
Δ = Discontinuity Free 영역을 구하기 위한 δ_j^s 의 한계값

따라서 어떠한 점에서의 Corner 의 정도는 아래의 두값에 의해서 결정된다.

- Incremental Curvature δ_j^s 의 Discontinuity
- 양변의 Discontinuity - Free 영역의 길이

즉 어떤점 j 의 Corner 의 정도를 나타내는 값을 그 점의 Cornerity K_j 라고하면 K_j 는 다음과 같이 표시된다.

$$K_j = \sqrt{t1} \bullet \sqrt{t2} \bullet \sum_{i=j}^{s+j} \delta_i^s$$

따라서 물체의 구석점은 각 점의 Cornerity K_j 와 그 점 주위의 적당한 범위내의 다른 Cornerity 값을 비교하여 가장 큰 값을 갖는 점을 그 물체의 구석점이라고 인식한다.

2.4. 구석점의 특징

구석점을 이용하여 구할 수 있는 특징들로는 다음 세 가지를 들 수 있다.

1> 구석점의 Convexty 또는 Concavity 의 판별
한 구석점의 좌표값을 그 점에 좌우로 인접한 두 점을 시계방향으로 연결하는 직선의 방정식에 대입한후 부호로써 결정한다. i 번째 그림조각의 j 번째 구석점의 Convexity 혹은 Concavity 를 CONi(j) 로 표시한다.

2> Corner Angle
하나의 구석점이 좌우로 이웃한 두 구석점과 이루는 각으로서 임의의 두 벡터 사이의 각을 구하여 계산한다. i 번째 그림조각의 j 번째 구석점의 Corner-Angle 은 다음과 같다.

$$ANGi(j) = \begin{cases} \Theta & : \text{Convex - Point} \\ 360 - \Theta & : \text{Concave - Point} \end{cases}$$

3> Corner Distance
하나의 구석점에서 시계방향으로 이웃한 다음의 구석점까지의 거리를 말하며 i 번째 그림조각의 j 번째 구석점의 거리를 DSTi(j) 로 나타낸다.

3. Jig - Saw Puzzle 의 조합

본 장에서는 전 장에서 구한 특징들을 이용한 Boundary Matching 방법에 의하여 그림조각을 조합하는 알고리즘을 설명한다.

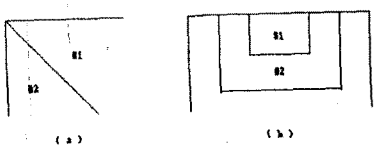
서론에서 언급한 바와같이 모든 경우에 대하여 알고리즘을 구현하기란 거의 불가능하므로 각 알고리즘에 따라서 그림의 모양에 제한을 두고있는데 그 대표적인 예를 들면 다음과 같다.

- 맞춰진 퍼즐의 구석점을 두 개 이상의 조각이 공유하는 경우. 그림 2.a
- 맞춰진 퍼즐의 한 변을 한 조각이 두 번 이상 공유하는 경우. 그림 2.b

본 논문에서는 그림 2. 와 같은 모양을 갖는 퍼즐에 대해서도 조합을 수행할 수 있는 알고리즘을 제안한다. 그러나 이 알고리즘역시 완전한 것이 아니므로 본 알고리즘이 효과적이기 위한 제한요소가 필요하다. 서로 다른 두 퍼즐에 속한 한쌍의 구석점의 Corner-Angle 의 합이 360° 또는 180° 인 구석점들이 있으면 좋은 결과를 얻지만, 이러한 점들이 없으면 알고리즘의 수행 결과가 좋지는 않다. 따라서 아래의 설명은 위 조건을 만족하는 퍼즐에 대한것으로 제한한다.

〈용어설명〉

- Master Blob : 두 Blob 이 조합될때에 이동과 회전의 기준이 되는 Blob ; #m
- Slave Blob : 두 Blob 이 조합될 때에 master 에 붙여지는 Blob ; #s
- Pm(i) : m 번째 Blob 에 속한 i 번째 구석점
- NUMm : m 번째 Blob 에 속한 모든 구석점의 수



(a) (b)

그림 2. 그림모양에 가해지는 제한

1° 초기화 : Object = Total number of Blob.
ublob = 0, flag = Off

2° Select Two Initial Candidate Blobs ;
Master & Slave-Blob

3° Checking Match-Condition.
IF not pass, IF flag = off GO TO 2°
ELSE GO TO 10°

```
4 °  Set Candidate Match-Pair.
5 °  Find    Match-Range.
6 °  Merging.
7 °  Decide Fit-or-Not.
     IF  not pass,   IF flag = off  GO TO  2 °
                     ELSE           GO TO 10 °
8 °  Save The  Information  of  Used Blob.
     flag = on,  ublob = ublob + 1
9 °  Construct the Patial Matched Blob   ;
     Master-Blob
10 ° IF ublob < Object
        Select One Candidate Blob ; Slave-Blob
        GO TO 3 °
11 ° Graphic Representation of Matched Puzzle
12 ° Accuracy  Measurement
13 ° END
```

그림 3. Fitting 알고리즘

3.1 후보조합 Blob(Candidate Match Blob)의 선택

조합을 수행하기 위해서는 우선 기준이 되는 퍼즐을 정하여야 한다. 기준이 되는 퍼즐의 선택은 Blob 의 추출과정에서 찾아진 순서대로 각 Blob 에 지정된 번호에 따라 조합조건을 적용하여 두 개의 Blob 을 선택한다. 알고리즘의 첫 부분에서는 여러 Blob 중에서 2 개를 후보조합 Blob 으로 선택하고 뒤의 3.5 절에서 설명하고 있는 Partial Matched Blob 을 구성한 다음에는 이 Blob 을 Master-Blob 으로 정하고 남은 Blob 중에서 하나를 선택하여 Slave-Blob 으로 한다.

3.2 조합조건 및 조합범위의 설정

조합조건(Match-Condition)은 서로 다른 두 Blob 에서 구한 후보조합점 (Candidate Match point) 조합여부를 결정하는 조건을 말한다. 본 알고리즘 이 효과적이기 위한 두가지 조건을 앞에서 제시하 였는데, 이들 조건에 따라서 조합조건 및 조합범위(Match-Range)의 설정방법이 달라진다.

● CASE A

1) 조합조건 : 서로 다른 두 Blob 에 속한 두개의 구석점이 내부점을 구성하는 경우
   $ANGm(i) + ANGs(j) = 360°$, $m \neq s$
   각 Blob 의 구석점에 시계방향으로 붙여진 번호를 이용하여 조합조건을 정한다. i 와 j 가 두 Blob 의 후보조합점이라면 조합조건은 다음과 같다.
   a) $CONm(i) + CONs(j) \neq 0$
   b) $|ANGm(i) + ANGs(j) - 360°| < 1$
   c) $|DSTm(i-1) - DSTs(j)| < 2$ and
      $|DSTm(i) - DSTs(j-1)| < 2$
   d) $CONm(i-1) + CONs(j+1) \neq 0$ and
      $CONm(i+1) + CONs(j-1) \neq 0$

   위의 조건 중에서 두 개의 볼록점은 360 °를 구성하지 못하므로 우선 a) 조건을 검사한다.

2) 조합범위 설정
   조합조건을 검사하여 두 구석점 Pm(i)와 Ps(j) ($m \neq n$) 가 조합점으로 판정되면 이들과 이웃한 점들 역시 조합조건을 만족하는 경우가 있어서 두 Blob 이 만나는 범위를 정하여야한다.그림 4. 의 알고리즘에서 Range-1 과 Range-2 는 Blob #m 의 조합범위이고 Range-2 와 Range-4 는 Blob #s 의 조합범위이다.

```
1 °  Set match-point ; Pm(i) & Ps(j)
2 °  direction = 1,   indirection = -1
     ii = i,          jj = j
3 °  ii = ii + indirection
     jj = jj + direction
4 °  Check match-condition ; Pm(ii) & Ps(jj)
     IF pass      GO TO 3 °
     ELSE         range_1 = ii, range_2 = jj
                  ii = i      , jj = j
5 °  ii = ii + direction
     jj = jj + indirection
6 °  Check match-condition ; Pm(ii) & Ps(jj)
     IF pass      GO TO 5 °
     ELSE         range_3 = ii, range_4 = jj
7 °  END
```

그림 4. 조합범위의 설정을 위한 알고리즘

● CASE B

1) 조합조건 : 서로다른 두 Blob 에 속한 두 개의 구석점이 변상의 점(Side-Point)를 구성하는 경우, 즉
   $ANGm(i) + ANGs(j) = 180°$, $m \neq n$
   이조건은 CASE-A 의 조합조건을 만족하는 구석점이 없을때에 적용한다. Master-Blob #m 의 구석점 i 와 Slave-blob 의 구석점 j 가 후보조합점인 조합조건은 다음과 같다. 그림 5.a
   a) $CONm(i) + CONs(j) \neq 0$
   b) $|ANGm(i) + ANGs(j) - 180°| < 1$
   c) $|DSTm(i) - DSTs(j-1)| < 2$

   위의 조건이 그림 5.b 에서는 성립하지 않는데 조합점을 중심으로 한쪽 방향으로만 비교하기 때문에 a), b) 조건을 만족하는 점의 위치에 영향을 받기때문이다. 이러한 경우에는 단지 Master-Blob 과 Slave-Blob 을 맞바꾼 다음에 위의 조건을 적용한다.그림 5.c 2) 조합범위 설정
   CASE-B 의 조합조건을 만족하면서 서로 맞닿는 한 변이 조합범위가 된다.

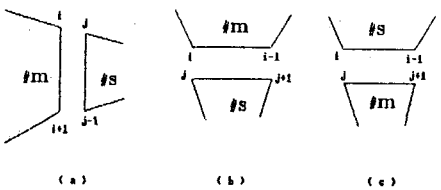


그림 5. CASE - B 에 대한 설명

3.3 Merging

두 개의 Blob #m 과 #s 가 조합된다고 판정되면 Master-Blob 에 Slave-Blob 을 일치시키기 위한 이동벡터와 회전각을 구하여야 한다. i 와 j, k 와 l 을 각각 Blob #m 과 #s 의 조합범위라고 하자.

1) 이동벡터
   두 개의 Blob #m 과 #s 의 서로 맞닿는 구석점 Pm(i) 와 Ps(k) 사이의 거리를 계산하여 구한다.
2) 회전각
   각 Blob 의 조합범위를 나타내는 두 구석점을 연결하는 벡터가 X-축과 이루는 각을 구한후 이들 두 각의 차이를 회전각으로 정한다.

3.4 Fit - or - Not 의 결정

전 과정까지 수행하여 두 Blob 을 일치시킨 다음에 이들 두 Blob 이 조합범위 밖에서 서로 겹치는지의 여부를 검사하여야 한다. 다음의 알고리즘을 두 Blob 의 조합범위 밖에있는 모든 변에 대하여 적용한다.

```
1 °  Slave-Blob 의  조합범위 밖에 있는 한 변을
     선택한다.
2 °  Master-Blob 의 조합범위 밖에 있는 한 변에
     대한 직선의 방정식 F(x,y) 를 구한다.
3 °  1 ° 에서 선택한 변의  양끝점 (Pi,Pe)을 2 °
     에서 구한 방정식에 대입한다.
4 °  IF  F(Pi) X F(Pe) > 0          GO TO 1 °
     ELSE  re_find the Match-Range of two Blobs
           #m and #s
```

3.5 Partial Matched Blob 의 구성

두 Blob 이 서로 조합된다고 판정하여 Merging 과정을 수행한 다음에는 남아있는 Blob 에 대하여 지금까지의 과정을 반복하여야 한다. 이 경우에 기준이 되는 새로운 Master- Blob 을 만들어 주어야 하는데 Merging 과정에서 새로이 만들어진 Blob 을 사용한다. 이 Blob 을 그대로 사용하기 보다는 다음과 같은 보정작업을 수행한 다음에 사용한다.

1) 사라지는 점(Hidden Point)의 처리
두 Blob 이 맞추어지면 서로 맞닿는 부분의 구석점들은 사라지는 점이므로 새로이 구성되는 Blob 의 구석점이 될 수 없다.
2) 변상의 점(Side-Point)의 처리
두 Blob 이 맞추어질때 두 Blob 의 조합범위의 끝 부분의 구석점들이 만나서 180° 를 이루면 이들 점은 새 Blob 의 구석점이 아니다.

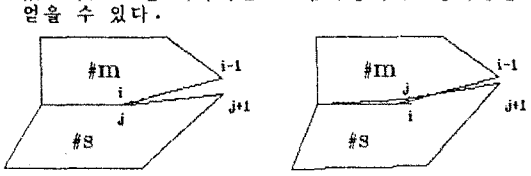
3) 형태의 조정(Form Adjustment)
두 Blob 이 조합범위에 따라 맞추어지고 나면 두 Blob 의 조합범위에 바로 이웃한 구석점들이 이루는 틈새가 생긴다. 이 틈새는 Slave-Blob 의 회전과정에서 생기거나 혹은 구석점의 부정확한 설정에 의한 것으로서 Merging 과정 전에서는 알아내기 힘들다. 따라서 두 Blob 이 맞추어진 다음에 이 틈새들을 보정하여 주는 형태의 조정 과정이 꼭 필요하다. 그림 6.a 의 Blob #m 과 #s 의 조합범위는 다음과 같다.
Blob #m ; range_1 = i, range_3 = i+1
Blob #s ; range_2 = j, range_4 = j-1
a) 두 구석점 Pm(i) 와 Ps(j) 가 다음조건을 만족하면 틈새를 형성한다고 판단한다.
- $|360° - ANGm(i) - ANGs(j)| < threshold$
- $|DSTm(i-1) - DSTs(j)| < threshold$

(b) 위의 두 Blob 에 대하여 조합범위를 확장시킨다.
Blob #m ; range_1 = i-1
Blob #s ; range_2 = j+1
c) 그림 6.a 의 Θ 는 벌어진 틈새의 각도로서 이 각에 대하여 b) 에서 새로이 구한 조합범위에 Merging 과정을 적용한다. 그리고 점 Pm(i) 와 Ps(j) 는 사라지는 점으로 처리한다. 그림 6.b
d) 새로이 구성된 Blob 에 대하여 위의 과정을 다시 실행한다.

위의 방법을 그림 6.c 에 적용하면 성립하지 않는다. 이 경우는 각 Blob 의 조합범위의 확장이 달라지게 된다.
Blob #m ; range_1 = i-1
Blob #s ; range_3 = k
이들의 조합범위에 따라서 다시 Merging 과정을 수행한다. 그림 6.d

4) 위의 과정을 거쳐서 만들어지는 Blob 에 대하여 각 구석점에 지정되는 번호와 특징들을 재조정한다.

위의 방법을 적용하다보면 조합범위 내에서 두 Blob 이 겹치는 경우가 종종 생긴다. 그러나 내부의 겹치는 정도가 심하지 않은 경우에는 전체적인 외형의 모습이 중요하다고 볼 수 있으며, 또한 계속되는 조합과정에서 정확성을 얻을 수 있다.



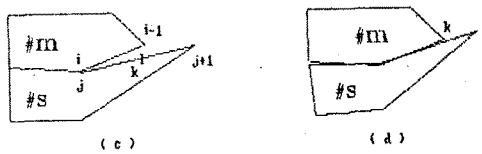


그림 6. 형태 조정이 필요한 조합된 그림조각

4. 결 론

본 논문에서는 카메라로 부터 그림조각의 영상을 받아서 맞추어진 그림에 대한 사전정보없이 그림조각 맞추기를 수행하였다.
알고리즘의 수행 결과는 그림조각이 놓여진 원래의 모습 그리고 근사화된 모양과 함께 나타내었다.
본 논문에서 다룬 그림맞추기는 Graphical Data 를 광범위하게 다루고 그 결과를 인식하는 문제들의 대표적인 보기이다. 이러한 문제와 관련되어 앞으로 더욱 개발해야 할 분야는 다음과 같다.

- 카메라 Lenz 에 의한 외각의 찌그러짐의 방지.
- Boundary Matching 의 근간이 되는 구석점을 보다 정확하게 찾는 알고리즘의 개발.
- 잘못 맞추어진 그림조각을 다시 맞추는 Back Tracking 알고리즘의 개발.

REFERRENCE

1. H. Freeman & L. Garder "A Pictorial Jigsaw Puzzles"
EEE Trans.on Electric Comput, Vol.EC-13, pp118-127, 1964
2. G. M. Rodack & N.I. Badler "Jigsaw Puzzle Matching using Boundary Centered Ploar Encoding"
C. G. I. P, Vol , No.19, pp1-17, 1982
3. R. L. T. Cederberg "Chain - Link Coding and Segmentation for Raster Scan Devices"
C. G. I. P, Vol.10, No. , pp224-234, 1979
4. H. Freeman "Computer Processing of Line - Drawing Images"
Computing Surveys,Vol.8,No.1, pp57-97, 1979
5. L. S. Davis "Understanding : Angles and Sides"
IEEE Trans.on Comput, Vol.26, No.3 pp236-242, 1977
6. T. Pavlidis & S. L. Horowitz "Segmentation of Plane Curves"
IEEE Trans.on Comput, Vol.23, No.8, pp860-870, 1974
7. J. G. Dunham "Optimum Uniform Piecewise Linear Approximation of Planar Curves"
IEEE Trans.on Pattern Anal.Mach.Intell, Vol.8, No.1, pp67-75,1986
8. H. Freeman & L. S. Davis "A Corner - Finding Algorithm for Chain - Coded Curves"
IEEE Trans.Comput, Vol.26, pp 205-216, 1977

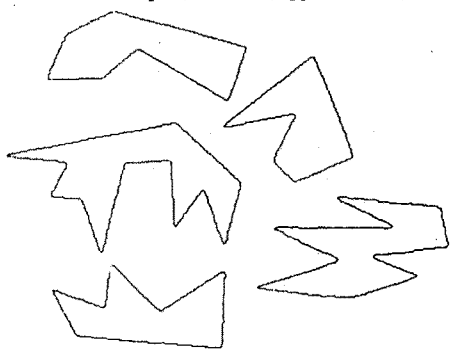


그림 7 . 원래의 그림조각의 모양

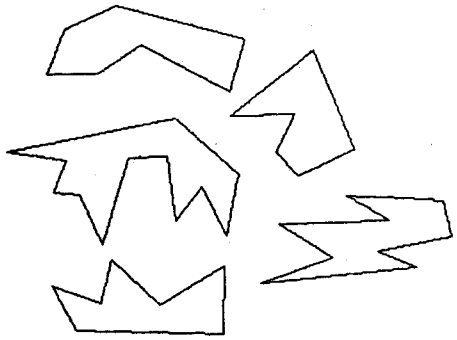


그림8 · 그림7 · 의 그림조각들을 근사화시킨 모양

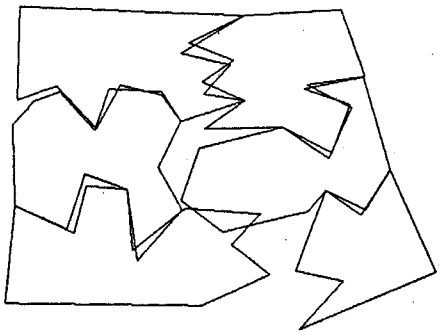


그림11 · 그림10 · 의 그림조각들의 맞추어진 모양

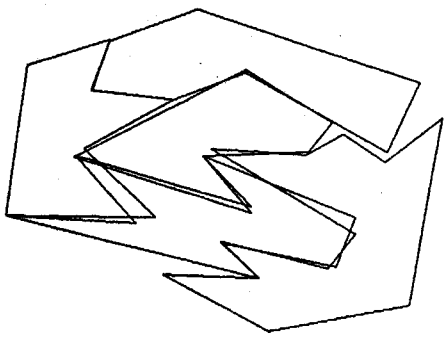


그림9 · 그림7 · 의 그림조각들의 맞추어진 모양

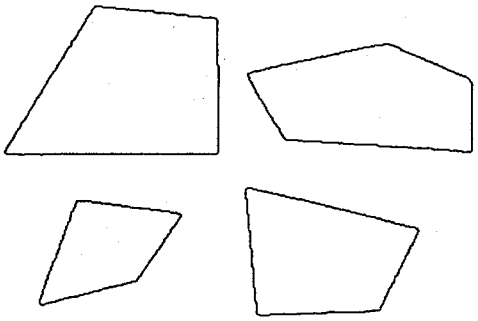


그림12 · 원래의 그림조각의 모양

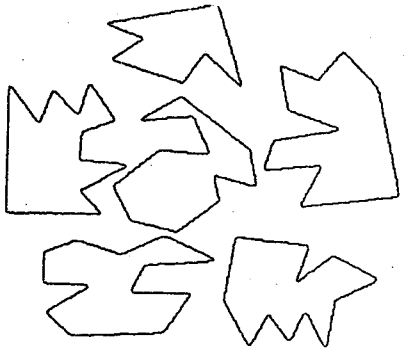


그림10 · 원래의 그림조각들의 모습

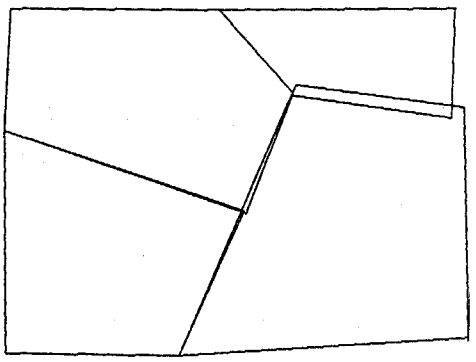


그림13 · 맞추어진 그림조각의 모양