구문에 의한 3차원 영상 인식에 관한 연구

°신 광 섭 *최 성 진 나 극 환 강 준 길
광 운 대 학 전 자 공 학 과, * 인덕공업전문대학 전자과

A STUDY ON 3-D IMAGE RECOGNITION BY SYNTAX

O SHIN KWANG SEOB, *CHOI SEONG JIN, RA KEUK WHAN, KANG JUNE GIL.
KWANG WOON UNIVERSITY, *IN DUK Jr. College of Eng.

ABSTRACT

In this paper, the effective and reasonal recognition method is represented for 3-D images. For effective computation, we contrive PDL generation algorithm and show recognition process of the PDL list by Prolog. In this results, it is possible to recognize the 3-D image in real-time.

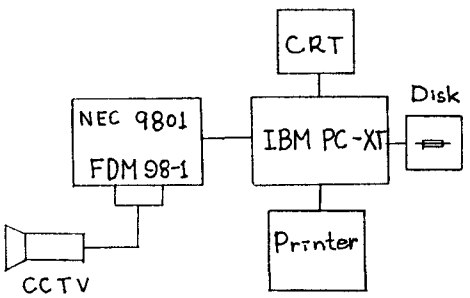
I. 서 론

그동안 A.I. 분야에서 입력 영상을 인식하려는 많은 시도가 있었고 발전을 거듭해 왔지만, 아직 인간의 능력에 비해 현저히 뒤떨어 지고 있다. 영상 인식은 고도의 영상 처리 기법이 적용된 후에야 좋은 결과를 기대 할수 있다. 다행히 이 분야에 많은 연구가 진행되어 좋은 결과를 얻고 있다.[1-8] 그러나, 가시 세계(Visible World)의 대상(Object)을 인식하는 데에는 2차원 정보에서 3차원 정보를 유도해 내야 하는등 많은 어려움이 따르게되므로 그간의 각고의 노력에도 불구하고 충분히 만족스러운 결과를 보지 못하고 있는 실정이다. 제한된 상황에서 국부적으로 원하던 결과를 얻기는 하지만 제약 사항이 많고 시간이 많이 걸린다는 단점이 있다. 인식에 있어 정확성과 시간은 가장 중요한 요소라 할수 있으므로 본 논문에서는 인식 시간을 최소한으로 줄이고 과정을 단순화 시켜 궁극적으로는 Computer 에 학습기능을 갖도록 하는데 주안을 두었다. 지금까지 Pattern 을 인식하기 위한 여러가지 방법이 시도되어 왔는데 [4] 본 논문에서는 Syntactic Pattern Recognition 방법으로 대상의 크기와 방향등에 영향을 받지않고 인식 할수 있는 방법을 제시한다.

입력된 화상을 경계선 검출(Edge Detection)과 세선화(Thining)를 거쳐 구문(Syntax)을 형성하는 전처리(Preprocessing) 과정을 거치게 되며 여기서 생성된 구문(Syntax)을 주 Program 에서 Data Base 에 저장된 정보와 비교하여 인식 하게 된다. 종래의 구문(Syntax)을 분석(Parsing) 하던 방법은 대상 물체마다 분석(Parsing)을 따로따로 해 주어야 하나 이 방법에서는 인식되지 않은 구문(Syntax)은 사용자가 정의한 이름으로 Data Base 에 저장되므로 많은 정보를 일일이 Type 입력 하지않고 CCTV로 직접 입력하여 정보를 저장 할수 있다. 즉, 학습 기능을 갖게 된다. 본 논문에서는 우선 전처리된(Preprocessed) 구문(Syntax)으로 인식하는 과정을 보여준다.

II. IMAGE RECOGNITION SYSTEM

1. 구 성



그 림 1. IMAGE RECOGNITION SYSTEM 의 구성

본 System 은 화상의 입력 및 전송부와 처리부로 나눌 수 있다. IBM PC-XT 를 주 계산기(Main computer)로 사용하는데, Hardware 의 제약 때문에 Gray Level 을 갖는 화상을 입력 할수 없다. 따라서 64 Gray Level 의 화상을 입력 할수 있는 NEC 9801 Microcomputer 의 FDM 98-1 화상 입력 장치로 화상을 입력하여 RS-232-C Serial Comunication Card 를 통하여 IBM PC-XT 로 전송하여 처리를 수행한다.

2. 영상 처리(IMAGE PROCESSING)

입력된 화상은 각각 64 치의 Gray Level을 갖는 256 * 256 크기의 화상이다. 이를 인식에 적합한 정보로 바꾸어 주기 위해 영상처리 기술에 쓰이는 다양한 기법을 적용한다. 즉, 화상의 입력 상태가 양호하지 못한 경우 잡음(Noise) 제거 및 강조(Enhancement)를 행한다. 다음으로 경계선 검출(Edge Detection)을 하고 유용한 경계선(Edge) 만을 선택하기 위해 세선화(Thining) 와 곡선의 적합화(Curve Fitting) 과정을 거친다. 이렇게 하여 처리된 화상 정보로 부터 본 논문에서 규정한 문법 규칙(Grammar Rule)을 적용하여 인식 할 대상을 언어(Language)로 바꾸어 준다. 이로써 인식을 위한 기본적인 처리는 마무리 된다. 여기서 생성된 언어를 Main Program 에서 내부 Data Base 와 비교하여 인식이 이루어진다. 이때 Data Base 에 존재하지 않는 언어가 입력되면 이는 거부(Reject) 되거나, 사용자의 정의(Definition)에 의해 Data Base 에 기록 하도록 한다. 그리하여, 다음부터 같은 언어가 입력되면 사용자가 정의한 이름으로 인식이 이루어진다. 처리 시간 단축을 위해 경계선 검출(Edge Detection) 부터 구문 생성(Syntax) 까지는 기계어(Machine Language)로 Program 하였고 인식 과정은 List 처리에 뛰어난 PROLOG LANGUAGE 를 사용하였다.

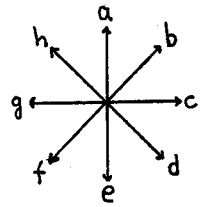
3. GRAMMAR RULES

본 논문에서는 PDL(Picture Discription Language) 을 적용하여 Computer 로 처리 하기에 편리 하도록 새로운 문법 규칙(Grammar Rule)을 고안 한다.

i) 방향

PDL 생성시 한 화소(Pixel)에서 3 * 3 격자(Template) 를 적용하게 되므로 8 방향으로 방향을 표시하게 된다.

즉,



ii) 크기

크기는 한 화소(Pixel)의 크기를 단위 길이(Unit Length) 즉, 1 로 하였다.

iii) 언어 생성 (LANGUAGE GENERATION)

논문에서는 Program 상에서 처리하기 용이하게 화상 정보를 단 하나의 List 로 생성 한다.

ㄱ) 시작점 (START POINT)

화상을 탐색(Scanning) 한 후 경계선(Edge)을 만나면 그 경계선을 탐색하여 정점(Vertex)을 찾으면 그 점을 시작점으로 한다.

ㄴ) 선분 (LINE)

같은 방향으로 연속된 화소(Pixel)를 모아 PDL List 의 원소가 된다. a 에서 h 까지의 방향과 크기를 나타내는 아라비아 숫자를 붙여 써서 표기하며 쉼표(Comma)로 분리 한다.

ㄷ) 정점 (VERTEX)

방향이 현저히 변하는 점. 즉, 선분과 선분이 만나는 점이 된다.

ㄹ) 분기점

2 개 이상의 선분이 만나는 정점(Vertex)으로써 본 논문에서 기호 ' * '로 표기하며 또한 PDL List 의 한 원소가 된다.

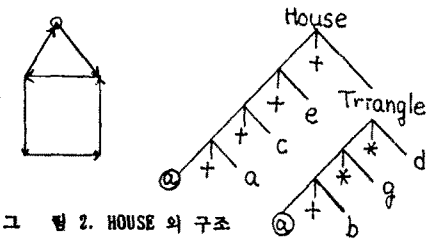
ㅁ) 종결점 (Termination Point)

선분의 끝이 아무런 연결이 없는 상태로 본 논문에서 기호 ' | '로 표기하며 이 점에서 역추적(Backtracking)이 일어난다.

ㅂ) 끝 점 (End point)

선분이 고리(Loop)를 형성 하게 되었을때 탐색(Scan) 하지않은 즉, 기록 하지않은 선분이 있으면 이 점은 분기점이 되어 분기를 하게되고 그렇지 않은 경우 분기를 할수 없으므로 이 점을 끝점이라 하고 기호' 0 '과 ' # ' 로 표기 하고 이 점에서 역추적(Backtracking)을 하게 된다. 여기서 기호 ' 0 ' 는 그 점에서 연결된 선분이 하나 뿐인 경우이고 하나 이상인 경우는 기호' # ' 로 나타내어 좀더 자세한 정보를 기술 할수 있도록 한다.

이상과 같은 규칙에 따라 그림 2. 을 나타내면 다음과 같다.



그 림 2. HOUSE 의 구조

G = (Vn, Vt, P, S)

Vn = { House, Triangle }

Vt = { d, e, g, a, b, c, r, t, y, z, +, (,) }

P : House --- (Triangle + (e + c + a + #))

Triangle --- ((d * g * b + @) + (e + c + a + @))

그러므로,

L(G) = ((d * g * b + @) + (e + c + a + #)) 이다.

여기에 각 변의 길이를 2 라 하고 이 길이 정보를 포함하여 나타내면

L(G)=((d2 * g2 * b2 + @)+(e2 + c2 + a2 + #))로 된다.

Computer 처리 시에 List 는 변과 연결 상태를 나타내는 정보들로 이루어 지기 때문에 단순히 연결 된 것 만을 나타내는 ' + ' 기호는 생략하여 간략화 하고 계산을 용이 하게 하기위해 기호 ' * ' 는 ' r ' 로, 기호 ' @ ' 과 ' # ' 는 각각 ' y ' 와 ' z ' 로, 기호 ' ! '는 ' t ' 로 바꾸어 준다.

이에 따라 PDL 을 바꾸어 다시 L'(G) 로 표시하면

L'(G)=((d2, *, g2, *, b2, y) , (e2, c2, a2, z)) 가 되고 이를 PROLOG 처리가 용이 하도록 최종 List를 형성하여 L1(G) 로 나타내면

L1(G) = ([d2, *, g2, *, b2, y, e2, c2, a2, z])가 된다.

4. Recognition.

이렇게 해서 생성된 List를 Main Program 에서 분석(Parsing) 하여 두개의 고리(Loop) 즉, 삼각형과 사각형이 서로 접해 있음을 인식 하게되고 이를 Data Base 에서 찾아 '집' 임을 인식 하게 된다. 그 과정을 보면 다음과 같다.

i) List 입력

전 절에서와 같이 생성된 List를 Main Program 에서 읽어 들인다.

ii) 분석

입력된 List를 방향 및 연결 상태와 길이 정보의 두가지 List 로 분리 한다.

iii) 인식

같은 물체라도 크기가 여러가지 이므로 크기를 제외한 방향 및 연결 상태 List 만으로 Data Base 와의 비교에서 인식이 이루어진다.

iv) 출력

인식이 이루어진 경우 즉, 같은 List 가 Data Base 에 존재 할 경우 그 이름을 출력하고 사용자의 필요에 따라 물체의 크기, 면적, 방향등도 계산하여 출력한다.

v) 재 처리

인식이 불가능 한 경우 원래 List 에서 불합리한 선을 제거함으로써 인식이 가능한 경우를 위해 불합리한 선을 제거하고 재 인식을 시도한다. 이때 인식이 되면 이를 출력하고 그렇지 않으면 원래의 List 를 사용자의 지정에 의해 새로운 정보로 Data Base 에 기록 한다.

III. 실험 및 고찰

본 논문에서의 실험 대상은 정육면체, 연필, 3각형 에서 6각형 까지의 평면 등이다. 3차원 물체는 2차원과 달리 보는 위치에 따라서 모양이 다르다. 그러므로, 각 방향에서의 PDL을 기억 해야 만이 어떤 상황에서든 3차원 물체를 인식 할수 있게된다. 그림(또는 사진)상의 3차원 물체를 2차원으로 인식하는가 아니면 3차원으로 인식 하는가 하는 문제이다. 인간은 이 같은 상황에서 '그림 속의 3차원' 으로 인식 하게 된다. 그러나 Computer는 원래 3차원 영상이든 똑같이 Memory 상에 2차원(엄밀히 말하면 1차원)으로 기억되기 때문에 이를 분간 할수 없게 된다. 이를 극복하기 위해서는 두대 이상의 영상 입력 장치를 이용하거나 [9] 다수의 거울을 적당한 각도로 떨어진 곳에 배치 하여 물체를 입력하게 되면 [10] 사진인 경우 각각의 Data는 같게 나올 것이고 3차원인 경우 서로 다르게 나온다. 따라서 복수대의 입력 장치를 사용하여 그 입력 Data 를 비교하면 '실제 3차원' 과 '그림속의 3차원' 을 분간 할수 있다. 불합리한 선의 제거시에는 세심한 고려를 해야 한다. 제거 기준에 따라 합당한 물체도 line 을 제거 하고 엉뚱한 물체로 인식 할수가 있다. 따라서 edge 에 직선적인 Noise가 가미된 경우는

쉽게 제거되나 edge 정보가 심하게 손실된 경우에는 이를 제거 함으로써 오인식이 일어나거나 인식 할수 없게 된다. 이때에는 System의 신뢰도를 높이기 위해 오인식 쪽보다는 인식 불능으로 처리 하게끔 제거 조건을 정하는 것이 좋다. 이 제거 조건에 대한 많은 연구를 하여 더 효율적인 조건을 찾아 내면 오인식을 크게 줄일수 있다. 물체가 단일 색상이 아니고 여러가지 그림이나 글씨가 씌진 경우 이를 바로 인식 할수 있어야 한다. 본 논문에서는 이점을 고려하지 않았지만 일단 물체를 인식한뒤 물체 내부의 그림이나 글씨도 같은 방법으로 PDL로 표현 할수 있다. 결국 글씨나 그림에 대한 PDL 도 전부 Data Base 에 저장하면 어렵지않게 그 물체의 종류나 제작 회사등을 인식 할수 있다. 단, 이때 글씨는 충분한 선명도를 가져야 한다. 영어 Alphabet 인식에 대한 Grammar 와 그 결과를[11] 에서 잘 보여준다. 물체가 겹쳤을 경우에 대해서는 고려하지 않았는데, 이 분야에도 많은 연구가 이루어져 있으므로[12,13] 이를 바탕으로 겹친 물체에 대한 합리적인 PDL 생성 Algorithm 이 제시되어야 할 것이다.

Ⅳ. 결론

실험 결과 본논문에서 채택한 구문(Syntax)에 의한 3차원 영상 인식(3-D Image Recognition) 방법은 거의 실시간(Real Time)으로 인식이 가능 하였다. 또한, 경계선 영상 정보(Edge Image Data)를 보관 하는데 IBM PC-AT/XT 의 경우 32Kbyte 의 저장 용량(Storage Memory)이 필요하나 구문 정보(Syntax Data)로 보관 할 경우 화면의 복잡도에 따라 차이가 있지만 통상 수십 Byte 정도면 보관이 가능하므로 정보(Data)의 저장 및 전송에 엄청난 정보 압축(Data Compression) 효과도 얻을 수 있다. Data Base의 규모가 커질 경우 인식 시간이 길어지는 단점이 있는데 이를 해결 하기위한 병렬 처리(Parallel Processing)와 효율적인 Data 검색 Algorithm의 제시가 요구된다. 본 논문에서는 간단한 직선성 잡음(Line Noise)이 첨가된 경우에 대해서도 인식이 가능함을 보였는데 앞으로 더 많고 복잡한 잡음이 첨가된 경우에 있어서도 인식이 가능하도록 하는 연구가 필요하다.

BIBLIOGRAPHY

[1] Thomas and S.Huang et al, "Advances in Computer Vision and Image Processing", JAI PRESS Inc., Vol 2, pp 1 - 88, 1986.

[2] Azriel Rosenfeld and Avinash C. Kak, "Digital Picture Processing", Academic Press, Inc., 2nd Ed. Vol 1, pp 209 - 267, 1982.

[3] S. Levialdi, "Edge Extraction Techniques", Fundamentals in Computer Vision, Cambridge University Press, pp 117 - 144, 1983.

[4] T. Pavlidis, "Structural Pattern Recognition", Springer-Verlag, 1977

[5] Rosenfeld, A. and Thurston M., "Edge and Curve Detection for Visual Scene Analysis", IEEE Trans. Computer, C-20, 562, 1971.

[6] T.M.R. ELLIS and D.H. McLAIN, "A New Method of Cubic Curve Fitting Using Local Data", ACM Trans. on Mathematical Software, Vol3, No.3, pp 175 - 178. June 1977.

[7] S.Mori, K.Yamamoto and M.Yasuda, "Research on Machine Recognition of Handprinted Characters", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.PAM 1-6, No.4, July 1984.

[8] B.Moayer and K.S. FU, "Fingerprint Classification", Syntatic Pattern Recognition, Applications, Springer-Verlag, pp 179 - 214, 1977.

[9] Y. Kuno, H. Numagamai, M.Ishikawa, H. Hoshino and M. Kidode, "Three-Dimensional Vision Techniques for An Advanced Robot System", IEEE ,1985.

[10] J. AMAT, V. LLARIO, "A Vision System with 3D Capabilities", IEEE, 1985.

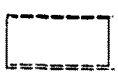
[11] King Sun Fu, "Syntactic Pattern Recognition and Applications" Prentice-Hall, Inc., pp 510 - 525, 1982.

[12] J, L. Turney, T.N. Mudge and R.A.Volz,"Recognizing Partially Hidden Objects", IEEE, 1985.

[13] R.L. Kashyap and Mark W. Koch, "Computer Vision Algorithms Used in Recognition of Occluded Objects", The First Conference on Artificial Intelligence Applications, IEEE Computer Society Press, pp150-155, 1984.

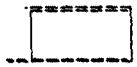
• 인식 결과

```
Goal : what([d50,g100,b50,y]).
It's a triangle !
 And derection is :
 [ d-g-b ]
And length is :
 [ 50-100-50 ]true
```

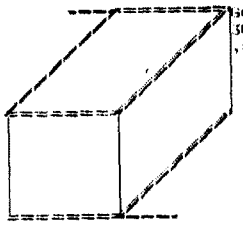


```
Goal : what([c60,e50,g60,a50,y
]).
 It's a quadrangular !
  And derection is :
  [ c-e-g-a ]
  And length is :
  [ 60-50-60-50 ]True
```

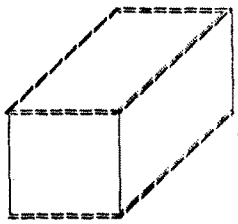
```
Goal : what([a100,b50,r,c35,t,d50,y]).
 It's a triangle !
  And derection is :
  [ a-b-d ]
  And length is :
  [ 100-50-50 ]true
```



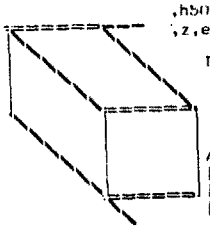
```
Goal : what([c50,e60,q60,r,a20,t,d50,y]).
   It's a quadrangular !
    And derection is :
    [ c-e-q-a ]
    And length is :
    [ 60-50-60-50 ]true
```



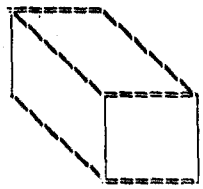
```
Goal : what([c60,r,e60,f60,r,c30,t,q60,a60,r,b60,y,q25,t,c60,r,b60,z,e60,z]).
 This shape is A CUBE !
   And consist of
  square [c-f-g-b] ,
  square [g-e-c-a] ,
  square [f-e-b-a]

 And it's length is
  [ 60-60-60-60 ]
  [ 60-60-60-60 ]
  [ 60-60-60-60 ]
   And it shares sides
  ( '60','60','60'  )true
```



```
Goal : what([c60,r,e60,f60,r,q60,a60,r,b60,y,c60,r,b60,z,e60,z]).
This shape is A CUBE !
  And consist of
 square [c-f-g-b] ,
 square [g-e-c-a] ,
 square [f-e-b-a]

And it's length is
 [ 60-60-60-60 ]
 [ 60-60-60-60 ]
 [ 60-60-60-60 ]
  And it shares sides
 ( '60','60','60'  )true
```

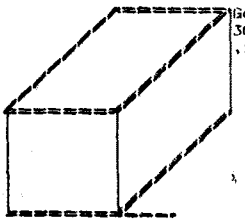


```
Goal : what([d50,r,e50,q50,r,d25,t,h50,a50,r,c50,y,c25,t,d50,r,c50,z,e50,z]).
 This shape is A CUBE .
   And consist of
  square [d-g-h-c] ,
  square [h-e-d-a] ,
  square [g-e-c-a]
 And it's length is
  [ 50-50-50-50 ]
  [ 50-50-50-50 ]
  [ 50-50-50-50 ]
   And it shares sides
  ( '50','50','50'  )True
```

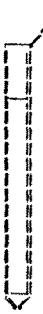


```
Goal : what([d50,r,e50,q50,r,h50,a50,r,c50,y,d50,r,c50,z,e50,z]).
This shape is A CUBE !
  And consist of
 square [d-g-h-c] ,
 square [h-e-d-a] ,
 square [g-e-c-a]

And it's length is
 [ 50-50-50-50 ]
 [ 50-50-50-50 ]
 [ 50-50-50-50 ]
  And it shares sides
 ( '50','50','50'  )True
```

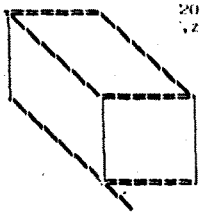
```
Goal : what([c12,e25,r,e115,r,f6,e5,g3,a5,h6,r,a115,r,a25,y,c12,z,c12,z]).

 It's A Pencil ! True
```



```
Goal : what([c60,r,e60,f60,r,c30,t,q60,a60,r,b60,y,c60,r,b60,z,e60,z]).
 This shape is A CUBE !
   And consist of
  square [c-f-g-b] ,
  square [g-e-c-a] ,
  square [f-e-b-a]
 And it's length is
  [ 60-60-60-60 ]
  [ 60-60-60-60 ]
  [ 60-60-60-60 ]
   And it shares sides
  ( '60','60','60'  )true
```



```
Goal : what([c12,r,b30,t,e25,r,e115,r,f6,e5,g3,a5,h6,r,a115,r,a25,y,c12,z,c12,z]).

 It's A Pencil ! True
```



```
Goal : what([d50,r,e50,q50,r,d20,t,h50,a50,r,c50,y,d50,r,c50,z,e50,z]).
    This shape is A CUBE !
      And consist of
    square [d-g-h-c] ,
    square [h-e-d-a] ,
    square [g-e-c-a]

    And it's length is
     [ 50-50-50-50 ]
     [ 50-50-50-50 ]
     [ 50-50-50-50 ]
      And it shares sides
     ( '50','50','50'  )true
```