표준화된 Edge기반 영상분석 알고리즘 개발을 위한 윤곽선 클래스 설계 및 구현

안기옥[O] 황혜정 채옥삼
경희대학교 컴퓨터 공학과
eroica@cvs2.kyunghee.ac.kr[O], kellt@korea.com, oschae@khu.ac.kr

Edge Class Design for the Development of Edge-based Image Analysis Algorithm

Kiok Ahn[O] Heajung Hoang Oksam Chae
Dept. of computer Science, Kyung Hee Univ.

요 약

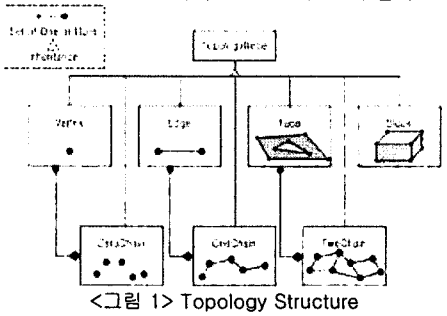
영상에 추출된 윤곽선(Edge)은 물체의 핵심적인 형태정보를 포함하고 있어서 영상인식과 분석의 근간이 되고 있다. 따라서 정확한 윤곽선 검출을 위한 많은 연구가 진행되고 있으며 그 응용분야도 다양하다. 그러나 정작 추출된 윤곽선 정보를 효율적으로 표현하고 활용하기 위한 표준화된 자료구조에 대한 연구는 많지 않아서 연구결과의 공유를 어렵게 하고 있다. 본 논문에서는 검출된 윤곽석을 효율적으로 표현, 관리, 검색, 조작하기 위한 자료클래스를 설계구현 함으로서 윤곽선검출 알고리즘의 표준화와 재사용을 촉진시키고 검출된 다양한 응용을 가능하게 한다.

1. 서 론

영상에서 추출된 윤곽선은 물체의 위치, 형태, 크기 정보 뿐만 아니라 Texture 정보까지도 포함하고 있어서 영상인식과 분석에 핵심적인 역할을 한다. 따라서 영상에서 정확한 윤곽선 검출은 집중적인 연구대상이 되어 왔으며 다양한 알고리즘이 존재한다[1]. 윤곽선을 응용한 연구도 여러 분야에서 진행되고 있다[2,3]. 하지만 이러한 연구들은 윤곽선의 효율적인 표현이나 검색을 위한 표준화된 방법 고려하지 않고 자신들의 목적에 특화된 자료표현 방법을 사용하였기 때문에 다른 연구자들이 활용하는데 어려움이 있다. 윤곽선 정보의 표현과 검색 방법은 응용분에 따라서 많은 차이가 난다. Hough 변환처럼 화소단위의 윤곽선 정보를 사용하는 응용분야에서는 배열과 같은 단순한 자료구조가 효율적인 반면에 물체의 구조적인 분석을 수행하는 응용 분야에서는 윤곽선의 구조적인 정보를 표현하고 신속하게 검색할 수 있는 자료구조가 필요하다. 이러한 윤곽선 정보를 계층적으로 표현하는 방안에 대한 연구는 영상처리분야와 컴퓨터 그래픽스 분야에서 이루어졌다. 먼저 컴퓨터 그래픽스 분야에서 가시화 측면에서 윤곽선을 기반으로 하는 물체의 표현과 가시화를 위한 연구가 이루어졌다[4]. 영상처리 분야에서 윤곽선의 표현방법으로는 ISO 표준이 PIKS[5]와 미국의 ARPA를 중심으로 정의된 IUE[6]가 중심축을 형성하고 있다. PIKS[10]는 영상자료와 윤곽선자료를 모두 배열로 표현하고 있으며 Khoros[7]와 Wit[8]에서도 같은 패턴을 따르고 있다. 이 방법은 윤곽선을 화소단위로 표현할 때는 효율적이나 윤곽선의 구조적인 정보를 표현하는 데는 어려움이 많다. IUE는 컴퓨터그래픽스의 윤곽선 자료표현 방법을 바탕으로 설계하고 객체지향 개념을 적용하여 표현한 방법으로 확장성과 표현력이 우수하다. 하지만 다양한 분야의 요구를 모두 수용하려다 보니 구조가 복잡해 지고 사용 방법도 어려워 일반 사용자들이 사용하기에는 어려움이 많다. 따라서 본래 취지와는 달리 널리 활용되지 않고 있다.대부분의 윤곽선 기반 영상분석 알고리즘들은 화소나 세그먼트 단위로 윤곽선 정보를 표현하고 사용하고 있다. 본 연구에서는 이러한 알고리즘을 보다 효율적으로 표현하고 활용할 수 있는 윤곽선 클래스를 설계구현하고 응용 방법을 소개한다.

2. 연구 배경

윤곽선을 이용한 영상 분석이나 영상인식 알고리즘에서 윤곽선은 주로 윤곽선단위나 세그먼트 단위로 사용된다. 그러나 구조적인 인식 방법이나 모델기반 인식 등에서는 세그먼트 사이의 관계, 면과 윤곽선 세그먼트의 관계가 필요하다. 이러한 관계를 잘 표현하고 있는 것이 IUE의 윤곽선 클래스 구조라 할 수 있다[9]. IUE의 윤곽선 클래스의 구조는 <그림 1>과 같다.



<그림 1> Topology Structure

이러한 IUE의 클래스 구조는 1차원의 Vertex에서부터 3차원의 Volume에 이르기까지 다양한 차원의 데이터를 효율적으로 관

리하는 것이 가능하다. 또한 상속관계는 Vertex, edge, line, region 등과 같은 기본 요소들 간의 관계 파악을 용이하게 해준다. 그 관계를 이용하여 영상을 분석할 경우 많지 않은 정보로도 다양한 정보를 추출할 수 있다. 또한 IUE는 객체지향을 기반으로 하여 설계되어 있다. 이는 영상 분할이나 인식할 때 영상의 중요한 정보를 포함하는 기본 요소들을 object단위로 관리하는 것을 가능하게 한다. 그리고 object단위이기 때문에 다양한 분야에 적용하거나 검출된 정보를 재사용 하는 것이 용이하다. 그러나 이 방법은 앞에서 언급한 것처럼 구조가 복잡하고 대부분의 윤곽선 기반 알고리즘에서 불필요한 부분을 많이 내포하고 있어서 널리 사용되지 않고 있다. 본 연구에서는 IUE의 윤곽선 클래스의 기본구조를 유지하면서 단순화된 윤곽선 클래스를 제안한다.

3. 제안된 윤곽선 클래스의 설계

윤곽선을 이용한 영상분석에서 효율적인 윤곽선 정보의 표현과 활용을 위한 윤곽선 자료 클래스 설계를 위해서는 다음과 같은 사항이 고려되어야 한다. 첫째, 윤곽화소를 기반으로 하는 많은 응용 시스템을 효율적으로 지원해야 한다. 이를 위해서 오버헤드 없이 윤곽선 화소 정보를 표현하고 신속하게 검색할 수 있는 구조를 지원해야 한다. 둘째, 물체를 구성하는 윤곽선의 구조적인 정보를 신속하게 구성하고 손쉽게 사용 할수 있어야 한다. 이를 위해서는 각 윤곽선 정보가 계층적으로 표현되어야 하고 윤곽선 세그먼트 사이의 관계정보를 쉽게 파악할 수 있는 정보와 구조를 가져야 한다. 셋째, 클래스의 개념에 익숙치 않은 대부분의 영상처리 전문가들이 손쉽게 사용할 수 있도록 간단한 구조를 가져야 한다. 꼭 필요한 구조와 기능으로 구조 자체를 간소화하여 표현 할 필요가 있다. 넷째, 표준화된 윤곽선 검출 및 응용 소프트웨어를 수정 없이 바로 사용할 수 있는 컴포넌트로 생성할 수 있어야 한다. 다섯째 추출된 윤곽선 정보를 독립적인 객체로 수정하고 조작하고 출력할 수 있어야 한다. 이와 같은 사항을 고려하여 본 연구에서는 그림 2와 같은 구조의 윤곽선 클래스를 제안한다.먼저 윤곽선 정보를 저장하기 위한 기본 클래스로 <그림 2>에 제시된 PointType, SegmentType, VertexTye으로 구성된다. PointType은 한 점을 표시한다. SegmentType은 각 윤곽선 세그먼트를 저장하는 클래스로 세그먼트의 시작 과 끝 Vertex의 포인터와 세그먼트를 구성하는 윤곽선 화소들을 저장하는 리스트를 갖는다. 윤곽선 화소를 배열에 저장할 경우에는 세그먼트의 시작과 끝 인덱스를 저장함으로서 구성 화소정보를 나타낼 수 있다. 어떤 방법을 취하느냐는 응용분야 따라서 달라질 수 있다. 예를 들어 각 세그먼트에 새로운 화소가 추가되거나 삭제되는 응용분야에서는 독립적인 리스트로 표현하는 것이 유리하다. VertexType은 각 분기점이나 단점에서 연결되는 세그먼트들의 리스트를 포함하며 이 정보와 세그먼트에 연결된 시작과 끝 Vertex 정보를 이용하여 윤곽선에 대한 한 차원 높은 구조적인 정보를 추출할 수 있다.

EdgeListType은 <그림 3>에 제시된 것처럼 위에서 정의한 기본적인 윤곽선 정보를 체계적으로 유지관리하기 위한 상위 클래스이다. 이것은 여러 계층의 윤곽선 정보를 생성, 관리, 검색, 삭제하는 역할을 담당한다. 윤곽선 검출과정에서 윤곽선을 추적하면서 이 클래스의 멤버함수를 이용하여 윤곽선 정보를 생성할 수 있다. 윤곽선을 구성하는 화소는 EdgeArray에 저장하거나 각 세그먼트의 화소 리스트에 저장할 수 있다. 만약 각 세그먼트별로 윤곽선 화소들을 독립적으로 표현되어 있을 경우 이들을 멤버함수를 통해서 하나의 배열에 표현할 수 있다. 이러한 배열은 윤곽선화소를 주로 이용하는 알고리즘에 효율적으로 사용될 수 있다.

<그림 4>는 위에서 정의된 클래스들로 표현된 윤곽선 자료의

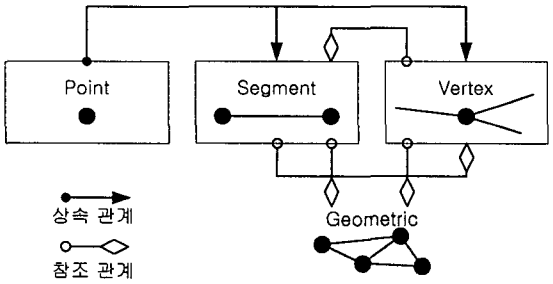
```
class PointType
{
//member data
public:
    int x, y;
//operator
bool operator == (PointType p)
{
 if(x==p.x && y==p.y)  return true ;
 else                  return false;
}
...
};

class EdgeSegmentType
{
//member data
private:
  PointType   m_Start , m_End;
  int         m_SegSize     ;
  PointType  *m_Point       ;
  int         m_SegIndex    ;
//member function
bool AllocPoint(int size);
bool SetPoint(list<PointType> *point);
...
};

struct ConnectType
{
  PointType         pt       ;//Vertex에 연결된 좌표
  int               direction ;//중심에서의 방향
  EdgeSegmentType *Seg     ;//연결된 Segment의 Memory
};

struct VertexType
{
  PointType     pt          ;// vertex의 좌표
  ConnectType  connected[8] ;// 연결된 segment.
  int           numSeg      ;// 연결된 segment 수
};
```

<그림 2> 기본 클래스의 ADT

```
class EdgeListType
{
// member data
private:

// edge pixel들에 관한 정보

// 전체 윤곽선화소를 저장하는 배열.
PointType *m_Points;
// 전체 윤곽선화소수
int m_NumOfPoints;

//segment들에 관한 정보
list<EdgeSegmentType> *m_EdgeList;// 윤곽선 세그먼트의 List

//Vertex정보. 각 segment들 간의 관계를 저장
list<VertexType> *m_Vertices;
// 각 세그먼트의 시작과 끝 윤곽선화소 정보를 담은 리스트
...

// member function
int   CreatePoints(int);//m_Point의 메모리를 할당하는 함수
//m_Point에 Edge Point를 저장하는 함수
void  SetPoints(int start, list<PointType> *point);

//Segment를 추가하는 함수
void  AddNewSegment(int size, list<PointType> *point);

//Vertex를 추가 하는 함수
void  AddNewVertex(int,int,int*);

...
void FindClosedSegment();
void          RotateSegment(PointType          center,          float
angle,EdgeSegmentType *sg);
void TransportSegment(PointType p, EdgeSegmentType *sg);

};
```

<그림 3> EdgeListTye Class

구조이다. 그림에서 보는 바와 같이 윤곽선 클래스에는 크게 세 가지의 데이터 즉, 각 픽셀 좌표의 배열과 기본단위 Segment의 리스트, 정점의 정보의 리스트로 저장된다. 배열에 저장된 데이터는 Hough 변환과 같이 윤곽선의 화소만 사용하는 알고리즘의 오버헤드를 줄인다. Segment의 리스트에 저장된 데이터는 폐곡선을 찾는 등의 알고리즘에서 매우 효율적으로 사용될 수 있다. 또한 Vertex List는 윤곽선의 구조적 해석을 쉽게 하고, Vertex에 연결된 Segment를 링크 함으로서 빠르게 직접적인 데이터를 액서스 할 수 있다. 이러한 구조 영상처리에 적합한 계층구조를 가짐으로써 불필요한 메모리 사용을 줄여서 메모리 효율성을 높여주고, 검색속도를 향상시킬 수 있다. 또한 윤곽선 간의 관계정보의 추출과 모델기반 인식을 위한 Face나 Volume로의 표현이 가능하다.



<그림 4> 제안된 윤곽선 클래스의 구조

4. 윤곽선검출 및 응용

본 논문에서는 Canny의 알고리즘[1]을 구현하여 제안된 윤곽선 클래스로 표현 하여 특정 알고리즘이 아니라 범용적으로 쓰일수 있는 예를 제시한다. <그림 6>에 제시된 Pseudo code는 추출된 윤곽선이 Thinning까지 처리 되었을 때 제안된 윤곽선 클래스의 맴버함수를 이용하여 맴버 데이터를 생성 하는 과정을 보여준다.

```
edge.CreatePoints(Edge Pixel Number);
list point;
Label=0;

Scan thinImg
  if : thinImg[current point];
    point.Add(Current Point)
    Scan 8 Direction
    Edgefollowing(!junctionImg);
    if : point.GetLength() != Noise)
      edge.SetPoints(point,Label);
      //Save Segment And Vertex Information
    Label++;
```

<그림 5> 맴버 함수를 이용한 맴버 데이터의 생성

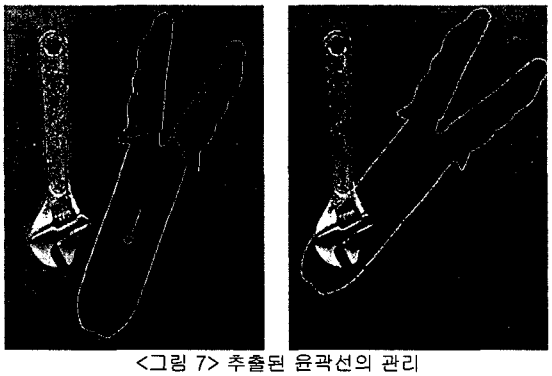
생성된 맴버데이터는 각 세그먼트의 Merge, Add, Delete등의 기본 연산 및 Transport, Rotate등의 클래스 맴버 함수를 지원한다. <그림 6>은 연결된 Close Loop을 찾아서 하나의 Object boundary를 추출하는 과정과 그 결과를 변환(회전)하여 출력하는 과정을 Pseudo code로 보여준다.

5. 결론

본 논문에서 제시한 윤곽선 클래스를 이용하여 쉽게, 그리고 효과적으로 윤곽선 정보를 검출하고 관리할 수 있도록 하였다. 클래스의 배열구조는 화소 정보만 사용하는 알고리즘에서 오버해드를 줄여주었고, 맴버함수 하나의 호출로 복잡한 과정 없이 윤곽선의 정보를 손쉽고 신속하게 구성할 수 있었다. 또한 윤곽선의 간단한 구조적 정보를 사용하여 Close Loop의 세그먼트를 찾아 이동, 회전하는 예를 보였다. 따라서 추출된 윤곽선 정보를 독립적인 객체로 수정하고 조작하고 출력할 수 있음을 보여 주었다. 제안된 윤곽선 클래스의 사용은 윤곽선을 이용하는 다양한 알고리즘에서 적은 메모리로, 빠른 속도로, 쉽게 사용 할 수 있다.

```
Scan Segment List
curLinkEdge = 현 Segment 의 끝점이 시작점(or 끝점)인 Segment;
while(curLinkEdge)
  if : Start Point OR End Point Of curLinkEdge == Start Point OR
End Point Of current Edge
    현 Segment 를 Close Segment로 설정
  curLinkEdge Update

등록된 Close Segment 와 연결된 Segment Merge
Transfort : current Edge;
Rotate : current Edge;
```

<그림 6>연결된 Close Loop를 찾아 이동, 회전하는 예



<그림 7> 추출된 윤곽선의 관리

Reference

[1] John Canny, A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, NO. 6, November 1986.

[2] F.Bartollini, V.Cappellini, "Autometic detection of the intrusions by image processing", Proceedings of the International Conference on Digital Signal Processing, 2:468~471,1993.

[3] M.B.Wilson, S.Dickson, "A Robust Road Boundary Detection and Tracking Algorithm", BMVC99, p352-361, 1999.

[4] Konstantinos, John R. Rasure, "The Khoros Software Development Environment for Image and Signal Processing", IEEE Trans. On Image Processing, No.1, pp. 506-510, 1994.

[5] John Rasure, et al, Teaching Image Processing with Khoros, International Conference on Image Processing, No.1, pp. 506-510, 1995.

[6] Keith Price, IUE Committee, "The Image Understanding Environment: Image Feature, Proceedings" IMAGE UNDERSTANDING I, pp. 311-315, 1993.

[7] Khoros Program Service, Khoral Research INC, http://www.khoral.com

[8] T.Arden, J.Poon, "WiT Uswe's Guide.", Logical Vision Ltd., Gilmore Way Burnaby, B.C., Canada, 1993.

[9] J.L. Mundy, IUE Committee, "Spatial objects in the Image Understanding Environment, Proceedings" IMAGE UNDERSTANDING I, pp. 317-331, 1993.

[10] Willam K. Pratt, "PIKS Foundtion C Programmers Guide", Prentice Hall, pp. 18-26, 1995.