論文 86-23-4-17

입체도형의 표현을 위한 隱面除去

(Hidden Surface Elimination for the Representation of 3-D Objects)

南 局 鎭*, 崔 炳 旭*

(Kook Jin Nam and Byung Uk Choi)

要 約

本 論文에서는 scan line을 이용한 隱面除去 algorithm을 제시한다. 이 algorithm은 서로 貫通하는 물체들에 대한 隱面除去를 포함하고, 이를 위한 segment들의 優先順位 판정방법을 최소한의 비교로써 최적인 span으로 나누어 처리함으로 계산시간을 단축시킨다. 또한 복수개로 正義된 모델끼리의 boolean 뺄셈을 허용함으로써 복잡한 물체의 斷面表現을 가능하게 했고, 이에 대한 隱面除去問題에 대해 論한다.

Abstract

In this paper, a hidden surface removal algorithm that generates a perspective view of three-space plane-faced objects using scan line coherence is proposed. In particular, the proposed algorithm deals with hidden surface elimination of penetrating objects by using the optimum span division method with minimum test for priority test of segments. Also, a hidden surface removal method for representing sectional view of solids by using boolean difference is described.

I. 序 論

건축, 토목, 선박, 기계부품 등의 CAD 시스템과 같이 복잡한 설계의 자동화에 있어서 컴퓨터 그래픽의 역할은 중요한 비중을 차지하고 있다. 특히 최근에는 CAD 시스템의 3次元化가 주목받고 있어 이에 대한 硏究가 활발히 진행되고 있다.[1,2] 이러한 3 차원 CAD 시스템에서의 입체도형 표현방법은 工學的인 이용외에도 映畵製作에서의 動畵像作成 또는 flight simulator의 模擬視界發生裝置와 같은 군사적인 목적으로도 이용되고 있다. 그러나 3次元 물체를 보다 사실적으로 表現하기 위해서는 다른 물체에 의해 가려져 보이지 않는 部分을 찾아 이를 처리해 주어야 하는 隱面除去 과정의 어려움이 뒤따른다. 그림 1 (a)와 같이 隱面處理를 해주지 않았을 경우에는 보는 사람에 따라 (b), (c), (d)로 각각 다르게 이해될 수도 있다.

이러한 隱面除去 문제는 計算量이 많아 處理에 상당한 시간이 소요되는 관계로 컴퓨터 그래픽스의 중요한 課題로 다루어져 왔다.

隱面除去方法으로는 크게 둘로 나누어 object space algorithm[3,4]과 image space algorithm[5,6]으로 나눌 수 있다.[7] Object space algorithm은 도형을 screen에 투영시키기 전에 visibility를 계산하여 priority 가 높은 부분만을 투영시켜 표현해 주는 것이고, image space algorithm은 도형을 screen에 투영시킨후 투영된 data에 대해 screen의 어느 부분이 visible 인가를 계산하는 것이다. 일반적으로 object space algorithm은 algorithm이 간단하고 計算의 정확도가 높은 반면

*正會員, 漢陽大學校 電子工學科

(Dept. of Elec. Eng., Hanyang Univ.)

接受日字 : 1985年 12月 21日

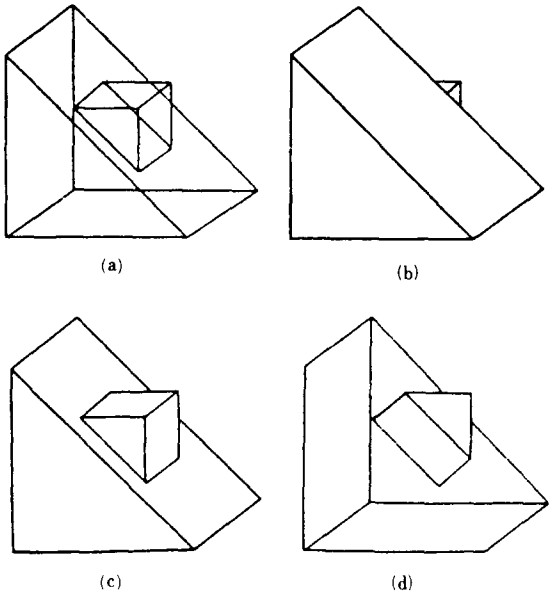


그림 1. 隱面除去의 효과
Fig. 1. Effect of hidden surface elimination.

表現하려는 물체가 복잡해 짐에 따라 계산시간이 증가되고, image space algorithm은 계산의 정확도는 다소 떨어지지만 表現하려는 물체가 복잡해져도 處理時間에 크게 영향을 주지 않는다. 이들 隱面除去 algorithm은 서로 貫通하는 물체들의 隱面除去에 있어서 多角形을 分割하거나 coherence를 적용하기 어려운 문제들이 있다.[8,9]

本 論文에서는 image space algorithm의 일종인 scan line algorithm에 기초를 두고, 이 algorithm으로는 해결이 어려웠던 서로 貫通하는 물체들의 隱面除去 방법과 이를 위한 segment들의 優先順位(priority) 判定方法을 최소한의 비교로써 최적인 span으로 나누어 행하는 방법을 제안하고, 아울러 복수개로 정의된 모델끼리의 뺄셈을 허용함으로써 복잡한 물체의 斷面表現을 가능하게 했고, 이에 대한 隱面除去 문제를 論하였다.

Ⅱ. Data의 構成

1. 모델의 正義

모든 물체는 多面體로 구성되어 있다고 볼 수 있고, 多面體는 多角形의 조합으로 表現될 수 있으므로 원하는 모델을 표현하기 위해서는 그 多面體의 꼭지점 좌표와 함께 그 多面體를 구성하고 있는 多角形을 정의해 주면 된다. 예를 들면 정육면체는 8개의 꼭지점 좌표와 6개의 多角形을 정의해줌으로써 表現될 수 있다. 自由曲面(free form surface)의 경우에도 이와같은 형식의 입력 data를 만들어 줄 수 있다.[10] 또 각각의 다각형은 꼭지점의 순서를 반시계 방향으로 正義해 주고, 多角形의 normal vector와 視線 vector의 내적에 의해 後方面(back face)과 前方面(front face)을 구분해 준다. 그리고 그림 2 와 같이 구멍(hole)이 있는 多角形의 경우에는 'O'을 사용하여 外部 多角形과 구분하여 표현하였다.

이렇게 하여 構成된 각각의 primitive들을 원하는 크기로 X, Y, Z에 대해 각각 확대 또는 축소시키고, 회전시킨 후 적절한 位置로 平行移動시켜 원하는 모델을 완성시킨다. 그림 2 모델의 data 구조를 표1에 나타내었다.

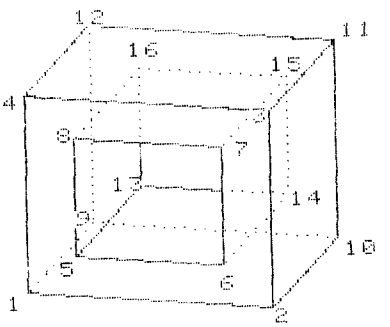


그림 2. 구멍이 있는 다면체의 표현
Fig. 2. Representation of polyhedron containing hole.

표 1. 그림 2 모델의 입력 data
Table. 1. Input data of Fig.2 model.

```
V 1=(0,0,0)   F 1: 9-10-11-12- 9- 0-13-14-15-16
V 2=(0,5,0)   F 2: 2- 3-11-10
V 3=(0,5,5)   F 3: 8- 7- 6- 5- 8- 0- 4- 3- 2- 1
V 4=(0,0,5)   F 4: 4- 1- 9-12
V 5=(0,1,1)   F 5:12-11- 3- 4
V 6=(0,4,1)   F 6: 1- 2-10- 9
V 7=(0,4,4)   F 7:13-14- 6- 5
V 8=(0,1,4)   F 8:14-15- 7- 6
V 9=(5,0,0)   F 9:15-16- 8- 7
V10=(5,5,0)   F10:16-13- 5- 8
V11=(5,5,5)
V12=(5,0,5)
V13=(5,1,1)
V14=(5,4,1)
V15=(5,4,4)
V16=(5,1,4)
```

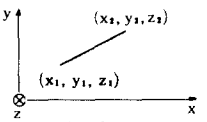
2. Edge block

Edge block(이하 EB라 함)은 모델을 視點座標系로 변환시켰을 때의 edge에 관한 여러가지 정보를 수록하고 있다. n번째 EB은 아래와 같이 構成된다. 만일 edge 중에서 x軸에 平行하거나 後方面에 속해 있는 edge는

EB에서 除外시킨다.

En	X_{min}	Y_{min}	Y_{max}	F_1	F_2	$\triangle x$	$\triangle y$	pointer
	0	1	2	3	4	5	6	7

X_{min} : Edge의 최소 x좌표값
Y_{min} : Edge의 최소 y좌표값
Y_{max} : Edge의 최대 y좌표값



F_1 : Edge가 속해 있는 面(face)의 번호
F_2 : Edge가 輪廓線(contour edge)이 아닌 경우 가질 수 있는 또 다른 面의 번호
$\triangle x$: y의 변화에 대한 x의 변화율
$\triangle x = (x_2 - x_1) / (y_2 - y_1)$
$\triangle y$: y의 변화에 대한 z의 변화율
$\triangle y = (z_2 - z_1) / (y_2 - y_1)$
pointer : 동일한 y_{min}값을 갖는 또 다른 edge 번호에 대한 pointer.

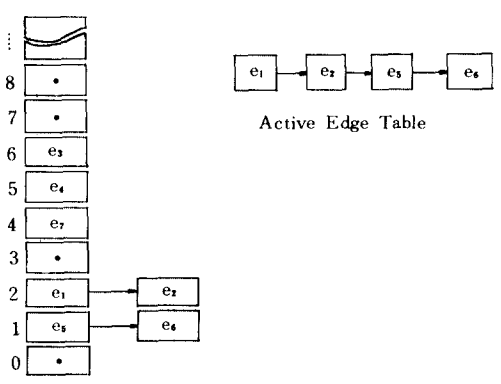
3. Edge Table과 Active Edge Table

Edge Table(이하 ET라 함)은 EB의 Y_{min}값에 의해 bucket sort된 table이다. 이때 같은 Y_{min} 값을 갖는 edge는 EB의 pointer에 의해 서로 연관된다. ET 의 크기는 screen의 크기에 비례해서 커지게 된다. Active Edge Table(이하 AET라 함)은 현재 scan line이 지나가고 있는 edge에 대한 table이다. 그림3에 대한 ET과 AET을 표2에 나타내었다.

Ⅲ. 隱面除去

그림4에 4개의 육면체로 構成된 모델에 대해 XZ平面에 平行한 임의의 한 scan plane이 만들어내는 segment를 표시했다. 이들 16개의 segment중에서 front segment만을 表示해 보면 그림4 (b)와 같이 8개가 된

표 2. Edge table과 active edge table
Table. 2. Edge table and active edge table.



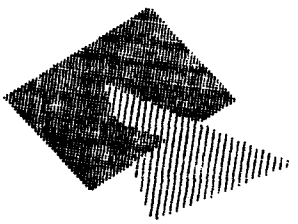
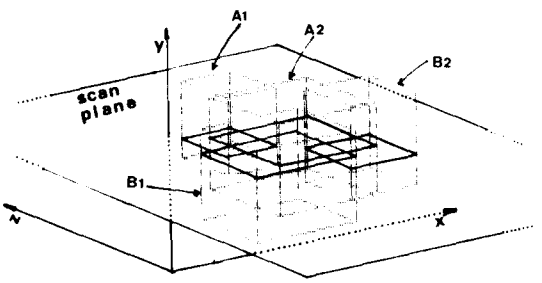
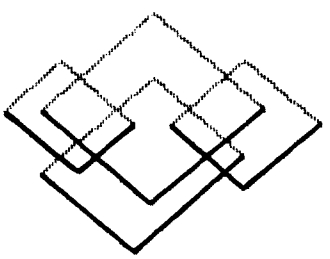


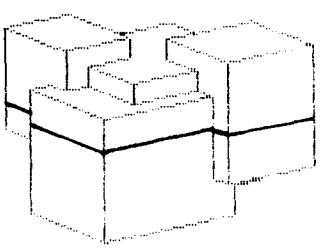
그림 3. 교차하는 두 평면의 표현
Fig. 3. Representation of two planes that are intersected.



(a)



(b)



(c)

그림 4. $A_1+A_2+B_1+B_2$의 경우
(a) Scan plane이 만든 segments
(b) Front segments
(c) 결 과

Fig. 4. A case of $A_1+A_2+B_1+B_2$.
(a) Scan plane making segments.
(b) Front segments.
(c) Result.

다. 이들 8 개의 segment에 대해 優先順位를 test하여 줌으로써 그림 4 (c)와 같이 원하는 결과를 얻을 수 있다.

1. 隱面除去 algorithm

전체적인 algorithm은 아래와 같이 構成된다. Step 9의 방법은 3 절에 기술한다.

$\left\{\begin{array}{l} SWY_{min} : \text{screen의 최소Y값} \\ SWY_{max} : \text{screen의 최대Y값} \end{array}\right\}$

step 1 : AET의 초기화
step 2 : $Y = SWY_{min}$
step 3 : $Y = Y + 1$
step 4 : if $ET(Y) = 0$ then step 3
step 5 : AET의 作成
step 6 : scan line과 edge와의 交點 (SX_n, SZ_n)을 계산한다.
step 7 : SX_n 값에 따른 AET의 sorting
EB의 F_1이나 F_2가 같은 edge를 AET의 순
step 8 : 서에 따라 segmentation한다.
step 9 : segment들의 可視部分 결정 (priority test)
step10 : $Y = Y + 1$
step11 : $(Y_{max} < Y)$인 edge를 AET에서 除外시킨다.
step12 : $SX_n = SX_n + \triangle X$, $SZ_n = SZ_n + \triangle Z$
step13 : if $ET(Y) = 0$ then step16
step14 : AET에 새로운 edge를 추가시킨다.
step15 : scan line과 추가된 edge와의 (SX_n, SZ_n)을 계산한다.
step16 : if AET≠empty then step 7
step17 : if $Y < SWY_{max}$ then step 3
step18 : END.

Algorithm을 간략히 설명하면 다음과 같다. 우선 AET를 초기화시킨 후, step 6~step 8 에서 AET을 이용하여 그림 4 (b)와 같은 segments를 구한다. 그리고 step 9 에서 가시부분을 결정한 후, scan line을 한 pixel 더 이동시켜 새로운 AET을 만든 후, scan line이 SWY_{max}에 도달할 때까지 위의 과정을 반복한다.

2. 두 segment의 교차[11]

그림 5 에서 두 선분 **L**, **L′**를 segment라 하고, 이에 대한 수직 vector를 각각 **M**, **M′**라 하면 S_0, S_1, S_0', S_1'에 각각 비례하는 값들은 式 (1)~(4)와 같이 表現된다.

$$S_0 = \mathbf{M} \cdot (\mathbf{P}_0' - \mathbf{P}_0) \quad (1)$$
$$S_1 = \mathbf{M} \cdot (\mathbf{P}_1' - \mathbf{P}_0) \quad (2)$$
$$S_0' = \mathbf{M}' \cdot (\mathbf{P}_0 - \mathbf{P}_0') \quad (3)$$
$$S_1' = \mathbf{M}' \cdot (\mathbf{P}_1 - \mathbf{P}_0') \quad (4)$$

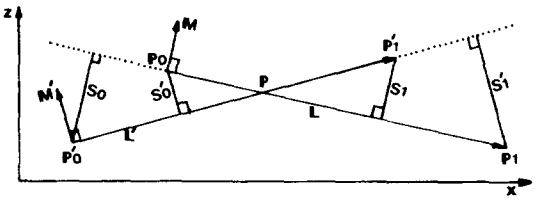


그림 5. 두 segment의 교차
Fig. 5. Intersection of two segments.

이때 두 선분이 교차하기 위해서는 式 (5), (6)을 만족하여야 한다.

$$S_0 \cdot S_1 < 0 \quad (5)$$
$$S_0' \cdot S_1' < 0 \quad (6)$$

式 (5), (6)을 만족하는 선분이라면 이들의 교점 **P**는 式(7)과 같이 구할 수 있다.

$$\mathbf{P} = \mathbf{P}_0 + \frac{S_0'}{S_0' - S_1'} (\mathbf{P}_1 - \mathbf{P}_0) \quad (7)$$

두 선분의 優先順位는 아래와 같다.

$S_0' < 0$, $S_1' > 0$: **L′**가 높다.

$S_1' < 0$, $S_1' < 0$: **L**이 높다.

$S_0' > 0$, $S_1' < 0$: X<**P**인 영역에서는 **L′**가 높고, X>**P**인 영역에서는 **L**이 높다.

$S_0' < 0$, $S_1' > 0$: X<**P**인 영역에서는 **L**이 높고, X>**P**인 영역에서는 **L′**가 높다.

3. 여러 segment들의 優先順位 결정

여러개의 segment들이 서로 복잡하게 얼켜있는 경우의 優先順位 決定도 두 segment의 비교로 한정지어 생각할 수 있다. 이 절에서는 X_{min} 값으로 sort되어있는 segment를 둘씩 비교해 가며 處理해 줌으로써 比較回數를 최소한으로 줄이는 동시에 span을 최적으로 나누는 방법을 기술한다.

두 segment의 관계는 그림 6 과 같이 8 가지로 나뉘어진다. 세 segment $\overline{p_1q_1}$, $\overline{p_2q_2}$, $\overline{p_3q_3}$를 각각 L_1, L_2, L_3라 하면 각 경우는 P_3의 위치에 따라 處理가 달라진다. 즉 P_3가 위치한 바로 전 영역까지 優先順位 問題를 처리해 주고 그 이후의 영역은 P_3를 포함한 두 segment로 하여 다시 처리하여 준다. 이때 處理할 수 없는 部分의 segment는 stack에 저장시킨다. Stack에 저장된 segment는 P_3의 X座標값과 比較하여 작은 값을 가질때 L_3로 代置되어 사용된다.

(e)의 경우를 예를 들면 P_3가 위치한 영역에 따라 3 가지 경우로 나뉘어진다.

A : $\overline{p_1'q_1}$과 $\overline{p_0q_2}$를 stack에 저장시키고 $\overline{p_1p_0}$를 L_1, L_3를 L_2로 한다.

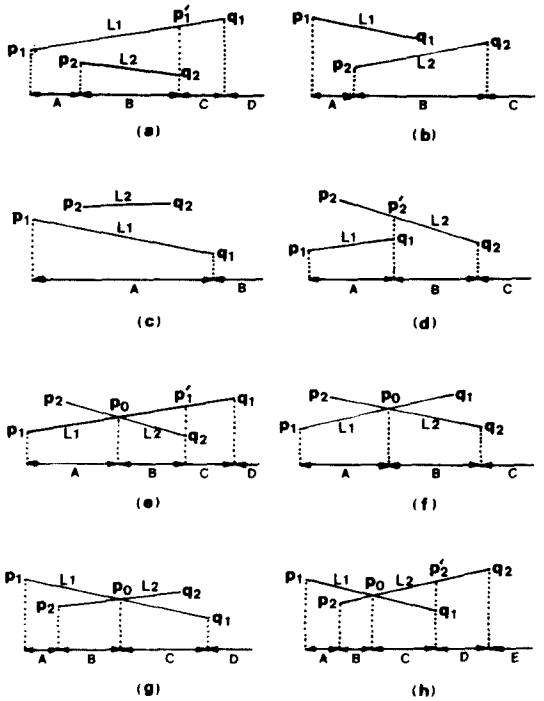


그림 6. 두 segment의 관계
Fig. 6. Relation of two segments.

B : A영역을 處理해 주고, $\overline{p_1'q_1}$을 stack에 저장시킨 후 $\overline{p_0q_2}$를 L_1, L_3를 L_2로 한다.

C : A, B 영역을 처리해 주고, $\overline{p_1'q_1}$을 L_1, L_3를 L_2로 한다.

Ⅳ. 모델의 boolean뺄셈

正義된 모델에서 斷面을 表示하기 위해서는 표시하고자 하는 斷面을 포함하는 多角形으로 구성된 또다른 모델을 정의해 주어 그 모델을 원래의 모델에서 제거함으로써 가능하게 된다. 이때 빼주는 도형을 복수개로 하고 이들 도형이 共有部分을 갖는 것을 허용한다면 좀 더 복잡한 斷面表現이 가능하게 되고 단면표현 외에도 새롭고 복잡한 물체의 표현이 쉽게 될 뿐만 아니라 주물의 형틀설계 등의 새로운 분야에도 이용될 수 있다.

1. 필요한 segment의 조건

그림 4 (a)에 나타낸 16개의 segment는 4 개의 多角形으로 構成되어 있고 각 다각형은 반시계 方向으로 正義되어 있다. 이들 16개의 segment중에서 $(A_1+A_2)-(B_1+B_2)$의 경우 필요로 하는 segment를 그림 7 (a)에 굵은 선으로 表示했고, 이때의 결과를 그림7 (b) 에 나타내었다.

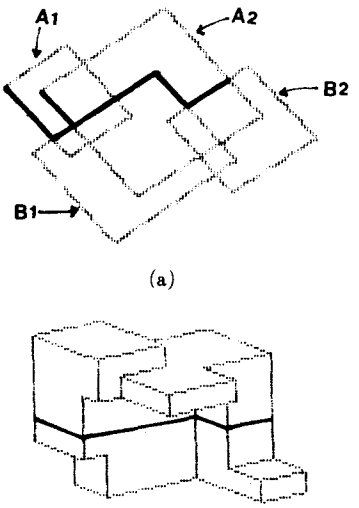


그림 7. $(A_1+A_2)-(B_1+B_2)$의 경우
(a) 필요한 segments
(b) 결 과

Fig. 7. A case of $(A_1+A_2)-(B_1+B_2)$.
(a) Needed segments.
(b) Result.

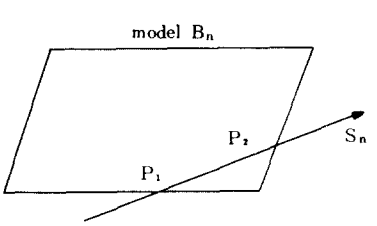
이들 segment를 구하는 조건은 아래와 같다.

- 모델 A에 속한 segment의 경우
  1) Front segment이어야 한다.
  2) 모델 B에 속한 多角形의 外部에 있어야 한다.
- 모델 B에 속한 segment의 경우
  1) Back segment이어야 한다.
  2) 모델 A에 속한 多角形의 內部에 있어야 한다.
  3) 모델B에 속한 多角形의 外部에 있어야 한다.

2. In-point와 out-point의 결정

두 segment가 만났을 때 이들 교점의 성질은 in-point와 out-point로 나눌 수 있다.

- In-point : P_1점과 같이 多角形의 內部로 들어가는 點. 이때 P_1점은 segment S_n에 대해 IB_n으로 表現된다.



- Out-point : P_2점과 같이 多角形의 外部로 나오는 點. 이때 P_2점은 segment S_n에 대해 OB_n으로 表現된다.

Segment의 交點이 in-point인지 out-point인지의 여부는 이들 segment가 front segment인지 back segment인지에 따라 그림 8과 같이 8가지 경우로 나누어 생각할 수 있다.

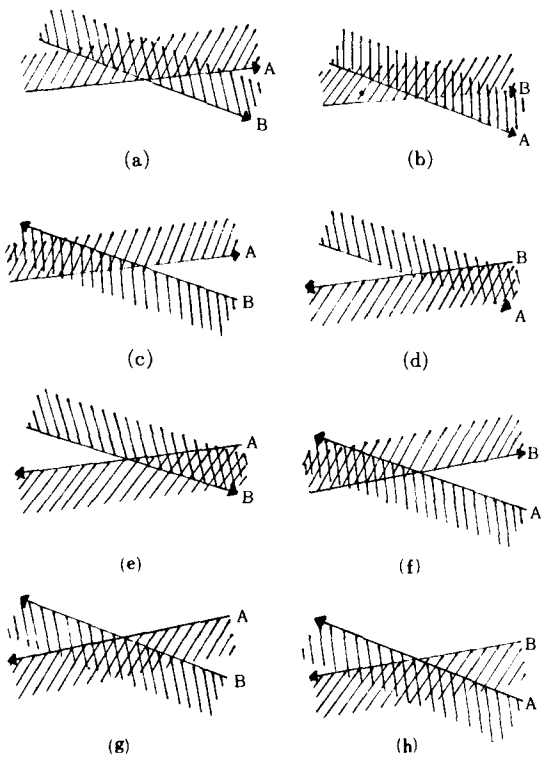


그림 8. In-point와 out-point의 관계
Fig. 8. Relation of in-point and out-point.

Front segment의 경우는 segment의 진행방향이 왼쪽에서 오른쪽으로의 방향이고 back segment의 경우는 오른쪽에서 왼쪽으로의 방향이다. 이때 진행방향의 왼쪽部分이 多角形의 內部가 된다. 각 경우에 대한 in-point와 out-point는 아래와 같다.

(a) (d) (f) (g) : A에 대해서는 IB, B에 대해서는 OA
(b) (c) (e) (h) : A에 대해서는 OB, B에 대해서는 IA

3. Algorithm의 구성

조건에 맞는 segment만을 추출하기 위해서는 segment로 構成된 多角形의 시작점과 다른 多角形과의 포함관계를 알아야 한다. 이를 위해서는 多角形을 반시계 방향으로 따라가며, in-point를 거치지 않은 out-point수를 check함으로써 알 수 있다. 전체적인 구성은 아래와 같다.

1) 多角形을 이루고 있는 segment를 반시계 方向으로 정의한다.

2) 각 segment의 in-point와 out-point를 계산한다.

3) 모델 A, B 각각의 多角形에 대해 in-point 와 out-point를 이용하여 조건에 맞는 segment를 구한다.

4) 구해진 segment를 X_{min}값으로 sorting한다.

5) 隱面除去 routine(Ⅲ-1절의 step9)으로 넘어간다.

4. 예외의 경우

지금까지는 A, B多角形이 交點을 갖는 경우에 대해서만 論하였지만, 그림 9 (a), (b)와 같이 서로 교점을 갖지 않는 경우에 대해서는 두 多角形의 포함관계를 test하여 이에 맞는 處理를 해주어야 한다. 즉 A多角形이 B多角形의 內部에 있거나 B多角形이 다른 B多角形의 內部에 있으면, 그 多角形을 구하려는 segment에서 除外시킨다. 또한 B多角形이 A多角形의 內部에 있으면, 그 多角形의 시작점 포함관계에 1을 더해 준다. 그러고 그림 9 (c), (d)와 같이 두 segment가 겹쳐있을 경우에는 아직 處理가 곤란한 問題로 남아있다.

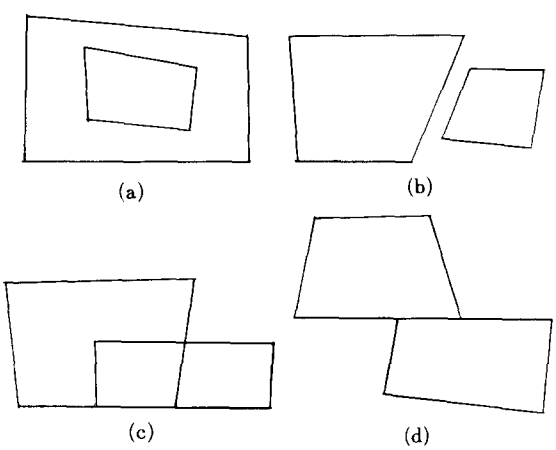


그림 9. 예외의 경우
Fig. 9. Situation of exception to the rule.

V. 실험 및 고찰

本 論文은 shading을 전제로하여 3次元 물체의 隱面除去를 수행하였다. 실험은 apple Ⅱ computer를 사용하였고, 결과는 X축에 대해 한번 더 scan하여 wire-frame으로 나타내었다.

Segment의 비교횟수는 그림 4 (b)와 같이 8개의 segment로 構成된 경우 n^2-n, 즉 56번의 優先順位 test가 필요하고, span의 分割은 segment의 양끝점과 交點으로 나눌 경우 14개의 span으로 분할된다. 그러나 本 論文의 Ⅲ-3절에 있는 方法을 사용하면 9번의

優先順位 test와 5개의 span으로 나뉘어지므로 계산 효율의 증가를 가져온다.

그림10(a)는 36개의 六面體로 구성되어 있고 아직 隱面處理가 되지 않은 상태이다. 이에 대한 隱面處理 결과를 그림10(b)에 나타내었다. 그림11(a)는 2개의 六面體와 2개의 원기둥으로 構成되어 있다. 이 상태에서 그림11(b)와 같은 모델을 빼주면 그림11(c)와 같이 원하는 모델을 얻을 수 있다. 이때 원하는 斷面을 잘라내면 그림11(d)와 같이 斷面을 表示할 수 있다. 즉, 그림11(a)에서 큰 육면체의 data를 A_1, 작은 육면체를 A_2, 2개의 원기둥을 각각 B_1, B_2라 하면, 그림11(a)는 $A_1+A_2+B_1+B_2$, 그림11(b)는 $A_2+B_1+B_2$, 그림11(c)는 $A_1-(A_2+B_1+B_2)$에 의해 각각 얻을 수 있다. 또 단면을 포함하는 모델C_1을 정의해 주면 그림11(d)는 $A_1-(A_2+B_1+B_2+C_1)$에 의해 얻을 수 있다. 그림12, 13의 경우도 마찬가지로 설명할 수 있다. 그림14는 형틀설계를 위한 simulation의 例로 그림14(a) 모델에 대한 형틀을 그림14(b)에 나타내었다.

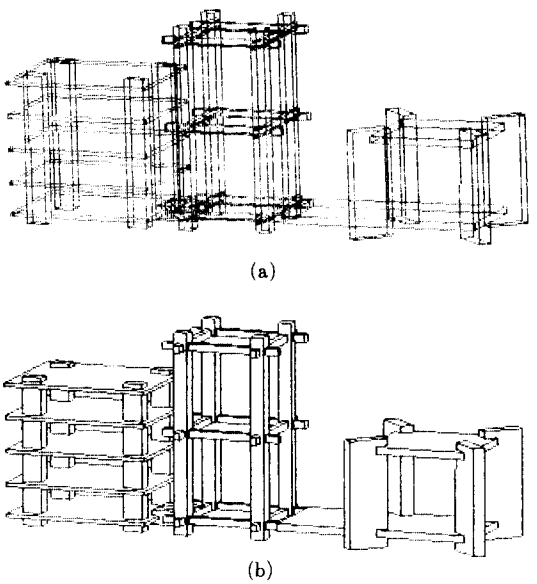


그림10. 은면제거의 예
Fig. 10. Example of hidden surface elimination.

Ⅵ. 結　　論

本 論文에서는 최소한의 比較로 segment들의 優先順位를 決定해 주는 방법을 제안했고, 이 방법을 사용하여 서로 貫通하는 물체에 대한 隱面處理를 하여 주었다. 또한 복수개로 正義된 모델끼리의 뺄셈을 허용함으로써 복잡한 물체의 斷面表現을 가능하게 했고, 이

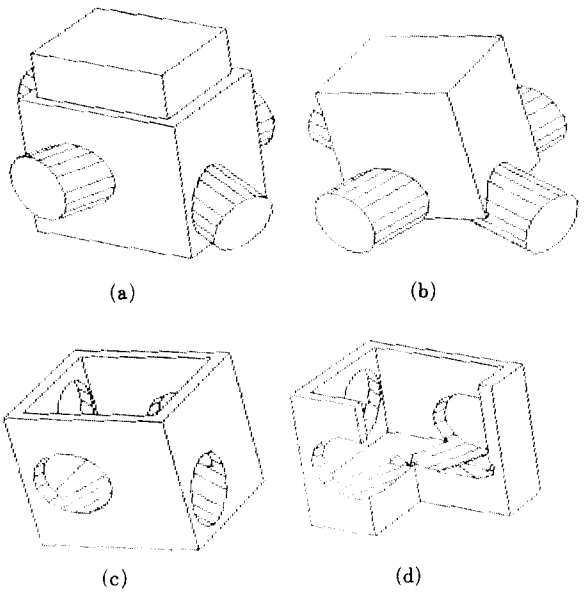


그림11. 단면 표현의 예 1
Fig. 11. Example 1 of representing section view.

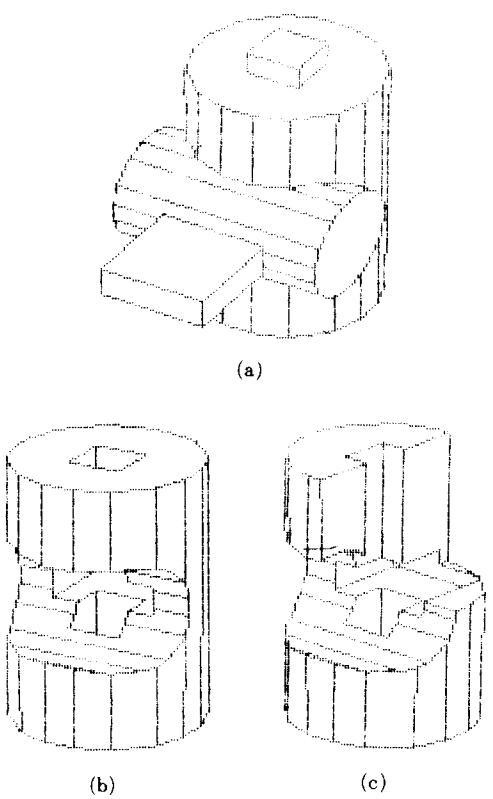
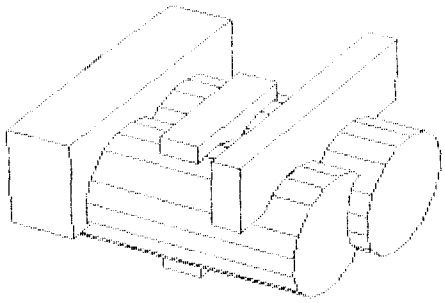
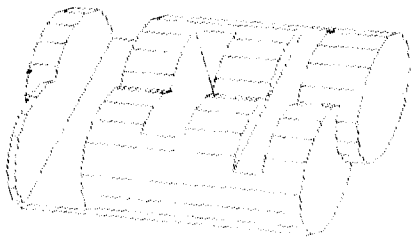


그림12. 단면표현의 예 2
Fig. 12. Example 2 of representing section view.

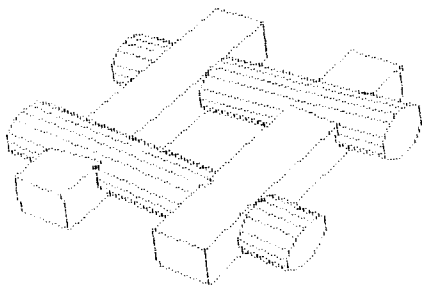


(a)

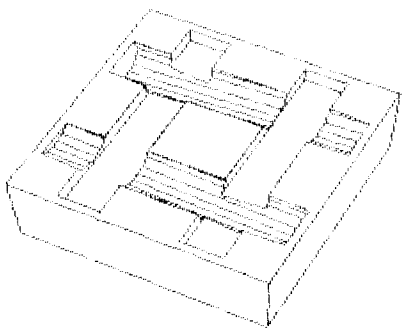


(b)

그림13. Boolean 빨셈의 예
Fig. 13. Example of boolean difference.



(a)



(b)

그림14. 형틀설계의 예
Fig. 14. Example of mold design.

에 대한 隱面除去 問題를 다루었다.

앞으로의 연구는 CSG(Constructive Solid Geometry) tree[6,12]를 사용한 보다 완전한 대화식 그래픽 시스템의 개발이 進行되어야 할 것이다. 또한 3次元 물체의 입력을 위한 입력기기[13]의 연구와 free-form surface에 대한 표현방법들의 연구가 계속 진행되어야 할 것으로 생각된다.

參 考 文 獻

[1] Ware Myers, *"An Industrial Perspective on Solid Modeling"*, IEEE CG & A pp. 86-97, March 1982.

[2] C.B. Besant, *"Computer-Aided Design and Manufacture"*, Ellis Horwood, 1983.

[3] M.E. Newell, R.G. Newell and T.L. Sancha, *"A Solution to the Hidden Surface Problem"*, proceedings, ACM Nat. meeting, 1972.

[4] S. Sechrest and D.P. Greenberg, *A Visible Polygon Reconstruction Algorithm*. Computer Graphics, 15 (3), pp. 17-27, August, 1981.

[5] G. Hamlin and C.W. Gear, *"Raster-Scan Hidden Surface Algorithm Techniques"*, Computer Graphics, 11 (2), Summer 1977.

[6] P.R. Atherton, *A Scan-Line Hidden Surface Removal Procedure for Constructive Solid Geometry"*, Computer Graphics, 17 (3), pp. 73-82, July 1983.

[7] Surtherland. I.E., Sproull. R.F. and Schumacker. R.A., *"A Characterization of Ten Hidden-Surface Algorithm"*, Computing Surveys, 6(1), pp. 1-55, March 1974.

[8] J.D. Foley and A. Van Dam, *"Fundamentals of Interactive Computer Graphics"*, Addison Wesley, 1982.

[9] William M. Newman and Robert F. Sproull, *"Principles of Interactive Computer Graphics"*, McGRAW-HILL, 1979.

[10] 김경배, 최병욱, "연속구를 이용한 곡면 조절점의 결정," 대한전자공학회 하계종합학술대회 논문집, 8(1), pp. 310-313, 1985.

[11] 山口富士夫, "コンピュータデトスプレトによる圖形處理工學," 日刊工業新聞社, 1981.

[12] John W. boyse and jack E. Gilchrist, *"GMSolid: Interactive Modeling for Design and Analysis of Solids"*, IEEE CG & A pp. 27-40, March 1982.

[13] Ivan E. Sutherland, *"Three-Dimensional Data Input by Tablet"*, proceeding IEEE, 62 (4), pp. 453-461, April 1974. *