A Hybrid Approach for Computing Visual Hulls of Complex Objects

Edmond Boyer and Jean-Sébastien Franco

Gravir-Inria Rhône-Alpes

In Proceedings of Computer Vision and Pattern Recognition, 2003

Abstract

- **이미지 윤곽선**으로부터 **Visual Hull** 을 계산하는 문제
- 혼합된 접근방식
  o Surface 기반 방식의 장점을 취함
  → Voxel 기반 방식이 가지는 정확도/복잡도의 Trade-off 를 극복
- **공간의 이산화(Space Discretization)** 도입
  o 대부분의 Cell 이 무효화되는 정규격자 기반 방식이 아니라,
  o Visual Hull 의 표면에 놓인 점들을 샘플링할 수 있는 **비정규격자** 기반 방식
  ; 샘플링된 점들로 **Delaunay 삼각화**를 수행, 사면체 형태의 Cell 로 격자를 구성
  o 격자를 구성한 후 이미지 윤곽선 정보에 따라 Cell 을 삭제(Carving)
- 현저하게 정확도를 증가시키고, 시간과 공간의 복잡도를 줄이면서, Voxel 기반 방식의 안정성을 유지
- 복잡한 형태의 물체 복원 가능

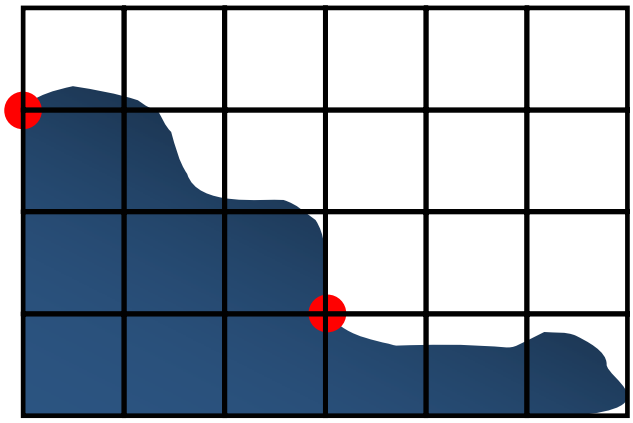
1. Introduction

- 여러 개의 다른 카메라 시점에 대하여 윤곽선 정보가 주어졌다고 가정
- Visual Hull: 물체의 윤곽선 정보와 일치하는 최대의 형상
- 이 논문의 목표
  ; Visual Hull 을 계산하기 위해 윤곽선 정보를 효율적으로 사용하는 방법을 서술
- 동기
  ; 복잡한 물체의 정확한 모델을 계산하기 위한 실제적인 해결책을 새로이 제안
  ; 시간과 공간의 복잡도를 적당한 수준에서 유지

- "The Visual Hull Concept for Silhouette-Based Image Understanding"
  ; A. Laurentini
  ; IEEE Transactions on PAMI (Pattern Analysis and Machine Intelligence), 1994
  o Visual Hull 의 개념을 처음으로 도입
  o 무한대의 시점이 있다는 가정하에 순수 이론적인 논리 전개

- Visual Hull 을 계산하는 알고리즘의 대표적인 두 가지 접근방식
  - o Visual Hull 로 둘러싸는 **Volume** 을 고려
    ; **공간의 이산화**에 기반
  - o Visual Hull 의 **표면** 자체를 고려
    ; 개별적인 **점과 다면체** 각각을 고려

- Volume 기반 방식
  - o 기본적인 Cell 단위(**Voxel**)로 공간을 이산화
    → 윤곽선에 대해 각 Cell 의 이미지 상에서의 위치에 따라 삭제여부를 결정
- "A Theory of Shape by Space Carving"
  ; K. Kutulakos and S. Seitz
  ; International Journal of Computer Vision, 2000
  - o Voxel 이 각각 다른 이미지에 투영된 픽셀의 **색 일치도(Color Consistency)**에 따라 삭제 여부를 결정
- 대부분의 기존 Volume 기반 방식의 특징
  - o 정규 Voxel 격자 기반
  - o 복잡한 형상의 물체 복원 가능
  - o 3 차원 공간 이산화의 문제점
    ; 막대한 계산시간 소요
    ; 불안정한 정확도 ← 대부분의 격자점이 Visual Hull 의 표면상에 있지 않기 때문



- 표면 기반 방식
  - o Visual Hull 의 외곽경계부분(점, 면)을 측정
    ; Viewing Cone 의 표면들에 교차(Intersection) 연산을 적용하여 생성
- "Image-Based Visual Hulls"
  ; W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan
  ; SIGGRAPH '00
  - o 물체의 새로운 시점에서의 이미지를 생성하는 연산을 Visual Hull 을 이용하여 2 차원 계

산만으로 수행

o 기하학적인 모델을 얻어내지는 않음 → Image-based Rendering

- 표면 기반 방식의 특징

o 정확성을 보장

o 생성된 모델이 종종 불완전하거나 형태가 망가지는 경우 존재

; Viewing Cone 의 경계부위를 교차시키는 연산이 제대로 정의되지 않는 경우, 형상이 복잡한 물체를 다룰수록 복원이 잘못될 가능성이 높아짐

→ 수치해석상의 불안정성에 매우 민감한 결과를 보임

- 이 논문의 접근 방식

o 양쪽 접근방식의 장점들을 동시에 취함

; Volume 기반 방식의 안정성 유지

→ 공간의 이산화를 그대로 사용

; 표면 기반 방식의 정확도 유지

→ 샘플링하는 점들의 간격이 일정하지 않고 Visual Hull 의 표면에서 계산됨

o 공간을 구성하는 Cell 의 단위는 샘플링된 점들의 Delaunay 삼각화로 구성한 사면체

o 각 Cell 들이 이미지에 투영된 위치에 따라 삭제여부를 결정

o 최종 생성된 결과물은 다면체 모델

o 특징

; 모델을 구성하기 위해 사용된 점들이 Visual Hull 의 표면 위에 존재

→ 높은 수준의 정확도

; Delaunay 삼각화를 통해 기하학적인 모델 생성

2. Definitions

- Pinhole 카메라를 가정
- 물체의 표면은 회전이 가능한 닫힌(Closed) 모델

; 부드럽거나 다면체

- 물체의 Genus 는 0 이 아닐 수도 있음
- 테두리(Rims)

o 시선방향이 표면에 접하는 점들의 물체 표면상에서의 궤적

o 테두리가 이미지 상에 투영된 곡선들을 **Occluding Contours** 라 함

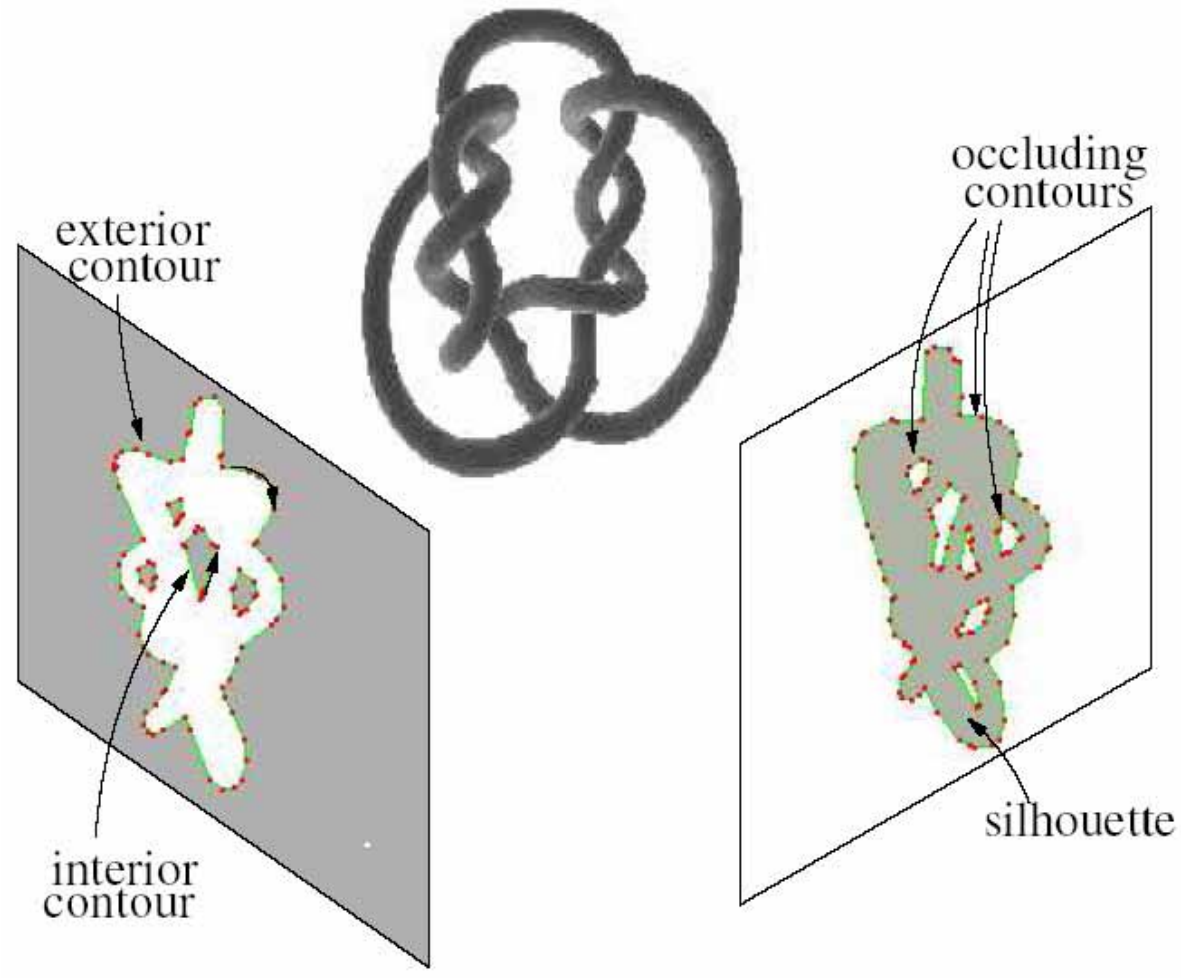
- Occluding Contours

o 이미지 평면상에서 물체의 외곽선을 형성

o O_j^i: 이미지 i 의 j 번째 Occluding Contour

o 물체가 오른쪽에 위치하도록 그 회전방향을 결정

; **외부/내부 외곽선(Exterior/Interior Contours)**: **반시계/시계** 방향



<Occluding Contour 의 예>

- Viewing Cone V_j^i
  - o Occluding Contour O_j^i에 연결
    ; 정점(Apex)은 이미지의 중심, 밑면은 Occluding Contour 의 내부로 구성
  - o O_j^i로 투영되는 테두리를 따라 해당 물체의 표면에 대해 접함

- Visual Hull
  - o 다른 시점에서 얻어낸 모든 Viewing Cone 들의 교차형상으로 정의

$$VH(I,C) = \bigcap_{i \in I,\, j \in C} V_j^i$$

; V_j^i: 이미지 i 에서 테두리 j 의 Viewing Cone

; I: 이미지 집합, C: 테두리 집합

o 유한개의 이미지 집합 I 를 가정할 때, Visual Hull 은 위상적으로 다면체를 구성
; 실제의 경우, Occluding Contour 는 2 차원 다각곡선으로 근사됨
→ Viewing Cone 도 다면체
→ Visual Hull 도 다면체
o 위의 정의는 이미지 집합 I 의 모든 이미지에서 단일 객체만 관찰될 때에 유효

- 정의 (1)
o 각 객체에 대한 Visual Hull 들의 합집합을 고려

$$VH(I,K) = \bigcup_{k \in K} VH(I, C_k)$$

$$= \bigcup_{k \in K} \left(\bigcap_{i \in I_k, j \in C_k} V_j^i \right) \quad (1)$$

; K: 대상 물체들의 집합, C_k: 물체 k의 윤곽선 집합
; I_k: 물체 k가 나타난 이미지 집합

o 이 정의를 바로 적용하기 위해서는 C_k와 I_k의 집합을 알아야 함
→ 모든 이미지 집합에서 각 물체들의 Occluding Contour 를 구별해야 함
; 간단한 연산이 아님
; 윤곽선이 하나의 이미지에서 겹쳐질 경우, 물체의 구별이 더욱 어려워짐

- 정의 (2)
o Visual Hull을 모든 이미지에서 보이고 윤곽선 내부로 투영되는 R^3상의 점집합으로 정의

$$VH(I,K) = \bigcap_{i \in I} \left(\bigcup_{k \in S^i} \left(\bigcap_{j \in C_k^i} V_j^i \right) \right) \quad (2)$$

; S^i: 이미지 i에서 윤곽선들의 집합, C_k^i: 이미지 i의 윤곽선 k에 연관된 외곽선 집합

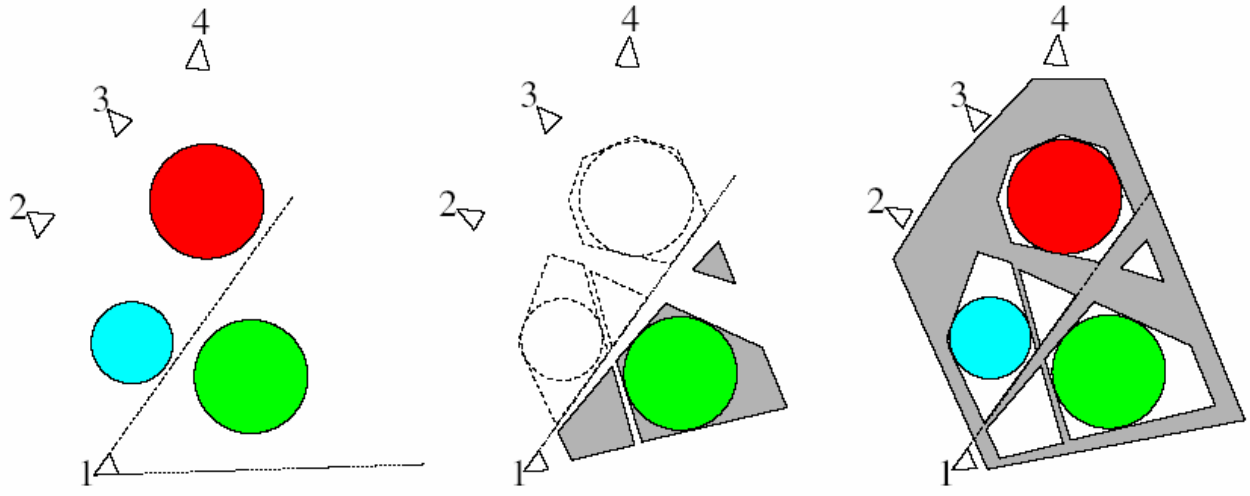
o 이미지 집합 내에 있는 외부 외곽선을 이용하여 쉽게 적용 가능
; 어떤 윤곽선이든 하나의 외부 외곽선과 여러 개의 내부 외곽선을 가지고 있음
o 물체가 하나 또는 여러 장의 이미지에서 보이지 않을 수도 있다는 가능성을 배제

- 정의 (3)
o **Visual Hull 의 여집합**을 고려

$$VH^c(I,K) = \bigcup_{i\in I}\left(\bigcap_{k\in S^i}\left(\bigcup_{j\in C_k^i} D^i \setminus V_j^i\right)\right)$$

; D^i: R^3에서의 이미지 i의 가시성 도메인
; $D^i \setminus V$: D^i 도메인에 대한 V의 여집합

o Visual Hull 이나 그 여집합을 계산하는 것은 **Dual** 연산관계
; 그 표면이 양쪽의 경계를 둘러싸고 있기 때문



<원래 장면> <정의 (2) 결과> <정의 (3) 결과>

- 정의 (2)와 (3)의 특징
o 원래 장면에서 나타나지 않는 가상의 물체가 복원될 수 있음
; 가시성 도메인 자체의 제약 때문
; 시점을 증가시켜서 이러한 문제를 줄일 수 있음

3. Visual Hull Surface Points

3.1 Algorithm Outline

- 이미지 집합으로부터 Occluding Contour 가 추출되고, O_j^i 가 다각 외곽선이라 가정
- 아래의 조건이 만족되면 Viewing Cone V_j^i 에 연관된 점들은 Visual Hull 의 표면을 구성
  - o 이미지 i 의 O_j^i 위에 투영되고,
  - o 다른 어떤 이미지에서도 윤곽선 여집합의 교차영역에 투영되지 않으면...
- 위의 조건을 만족하는 점들을 찾기 위해서
  - o O_j^i 위의 점들이,
  - o 다른 시점의 Viewing Cone 과 해당 점들의 시선이 교차하는 부분을 조사
    ; 대부분의 연산이 2 차원 이미지 상에서 수행됨

<Pseudocode>

Algorithm 1 Visual hull surface points

```
1:  for all contours O_j^i in all images: do
2:    for all images k such that k ≠ i: do
3:      for all points p_j^i in O_j^i : do
4:        compute the epipolar line l of p_j^i in image k,
5:        for all contours O_l^k in image k: do
6:          compute the intersections of l with O_l^k,
7:          update depth intervals along the viewing line
            of p_j^i,
8:        end for
9:      end for
10:     compute the 3D points delimiting intervals along
        the viewing line of p_j^i.
11:   end for
12: end for
```

3.2 Updating Depth Intervals along the Viewing Line

- 다른 외곽선이나 이미지로부터 두 개의 깊이 정보에 대한 리스트를 조합할 것인가?
  - o 먼저 각 이미지로부터 윤곽선을 구별 → 외곽선을 그룹별로 구분
    ; 외부 외곽선을 이용 ← 하나하나가 개별적인 물체를 가리킴
  - o 내부 외곽선은 완전하게 Visual Hull 의 여집합에 포함됨

→ 외부 외곽선들만 교차를 검사

$$VH^c(I,K) = \bigcup_{i \in I} \left[\left(\bigcap_{j \in Ext^i} D^i \setminus V_j^i \right) \bigcup \left(\bigcup_{j \in Int^i} D^i \setminus V_j^i \right) \right] \quad (4)$$

; Ext^i/Int^i: 이미지 i의 외부/내부 외곽선 집합

<원본 모델>

<40 개의 시점에서 찾아낸 Visual Hull Surface Points>

3.3 Complexity

- $O(n^2m^2qr)$ 시간동안 $\theta(nmq)$개의 3D 점들을 계산
  ; n: 이미지 개수, m: 이미지 당 외곽선의 개수, q: 외곽선 당 점의 개수
  ; r: Line-Contour 교차 연산의 상한 복잡도
- 순수한 교차연산의 복잡도 r=O(q)
  ; q 개의 점을 지닌 외곽선을 가정

→ 총 복잡도는 $O(n^2m^2q^2)$ or $O(N^2)$
; N: 계산된 3D 점들의 개수

- 교차연산의 최적화를 통해 복잡도 r 이 O(1)으로 감소 가능
  o Epipolar Rectification
  ; Epipolar Line 이 수평선이 되도록 이미지를 수정
  ; 교차연산은 단지 Lookup 값으로 세로좌표만 이용
  o Epipole 과 각 외곽선점을 연결하는 선의 각도를 Lookup 값으로 사용
  ; 이미지 수정이 불필요

o 위의 두 가지 방법 모두 r 을 O(1)으로 변경
; 이미지를 수정하거나 외곽선점의 각도를 계산하는데 $\theta(n^2mq)$ 개의 연산을 추가
o 최적화된 교차연산을 사용하여 총 복잡도는 $O(n^2m^2q)$ 가 됨

4. Visual Hull Surface

- 전통적인 Voxel 기반 접근 방식
; 정규격자 Cell (Voxel)로 이루어진 3D 공간의 부분을 깎아서(Carving) 구성
- 이 논문의 접근 방식
; 계산된 Visual Hull 점들을 이용해 **Delaunay 사면체**를 구성한 비정규격자 Cell 로 구성
→ 시간과 공간 복잡도가 합리적인 수준으로 정확성을 유지할 수 있음

4.1 Point Triangulation

- Visual Hull 표면의 점들을 이용한 Delaunay 사면체를 구성
; 비조직적인 3D 점들로부터 표면을 복원하는 알고리즘
- Delaunay 삼각화의 두 가지 장점
o 외접하면서 비어있는 부분(?)과 같은 성질을 만족시키는 Cell 구성이 가능
o 빠르고 안정적인 구현방법이 존재

- Delaunay 삼각화의 부분집합을 탐색
o 2D 이미지 정보를 고려
- Delaunay 삼각화의 복잡도
o 알려진 바로, 최악의 경우 $O(n^2)$, n은 점의 개수 → $O(n^2m^2q^2)$
o 몇몇 최근 연구들은 Delaunay 삼각화가 점의 개수에 대해 선형복잡도를 가짐을 증명

4.2 Surface Extraction

- Delaunay 삼각화를 이용하여 입력된 점들의 Convex Hull 을 구성하는 사면체 집합을 생성
- 생성된 사면체 집합으로부터 Visual Hull 의 여집합에 속하는 사면체들을 분류하고 삭제
o 간단한 접근 방식
; 사면체의 중심을 이미지에 투영시킨 후 윤곽선의 내부인지 여부를 판단
; 배경과 전경에 대한 정보를 표현할 수 있는 이진 이미지가 주어지면 빠른 속도를 보임
- 중심이 윤곽선의 내부에 투영되는 사면체의 집합은 Visual Hull Cell 로 볼 수 있음
- 최종 Surface 모델은 모든 점이 Occluding Contour 에 투영된 삼각형 면으로 구성된 다면체

- 3D 점들뿐만 아니라 각 점의 쌍을 연결하는 선분 또한 Visual Hull 을 구성
→ Conforming Delaunay 삼각화
; 삼각화된 결과가 미리 정의된 선형 혼합물(모서리, 면)을 포함
; 제약조건을 위해 입력 데이터에 많은 수의 점을 추가 → 매우 느린 수행 속도

5. Experimental Results

- 결과 #1: Knots 모델

o 전체적으로 상당히 낮은 복잡도에 기하학적으로 향상된 결과 생성

o 이 논문의 방법

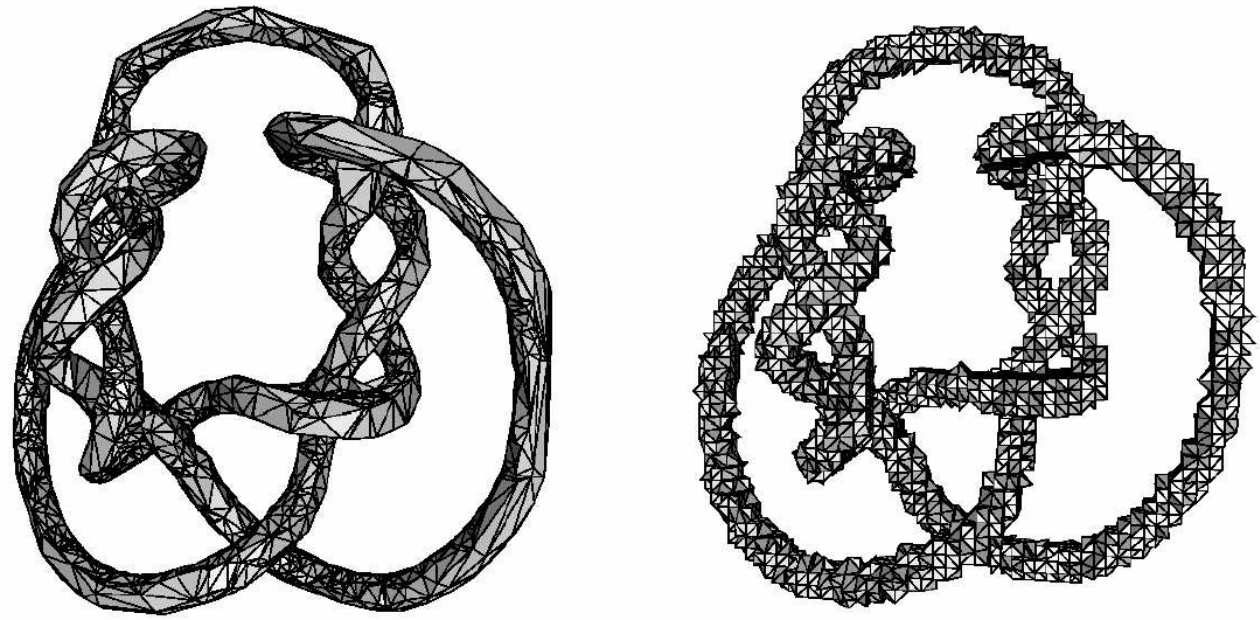
; 3772 개의 점을 복원

; 이미지 개수에 대한 상한 복잡도가 2 차

o Voxel 기반 방법

; 60^3개의 Voxel을 검사하여 생성

; 이미지 개수에 대한 상한 복잡도가 선형 ← 이미지가 공간 분할에 영향을 미치지 않음



<이 논문의 방식>

<Voxel 기반 방식>

- 결과 #2: Torus 모델

o 임의로 분포된 카메라로 찍은 이미지를 입력으로 받아들임

o 외곽선에 점을 추가하는 것이 수행시간에 많은 영향을 끼치지 않음 ← $O(n^2m^2q)$

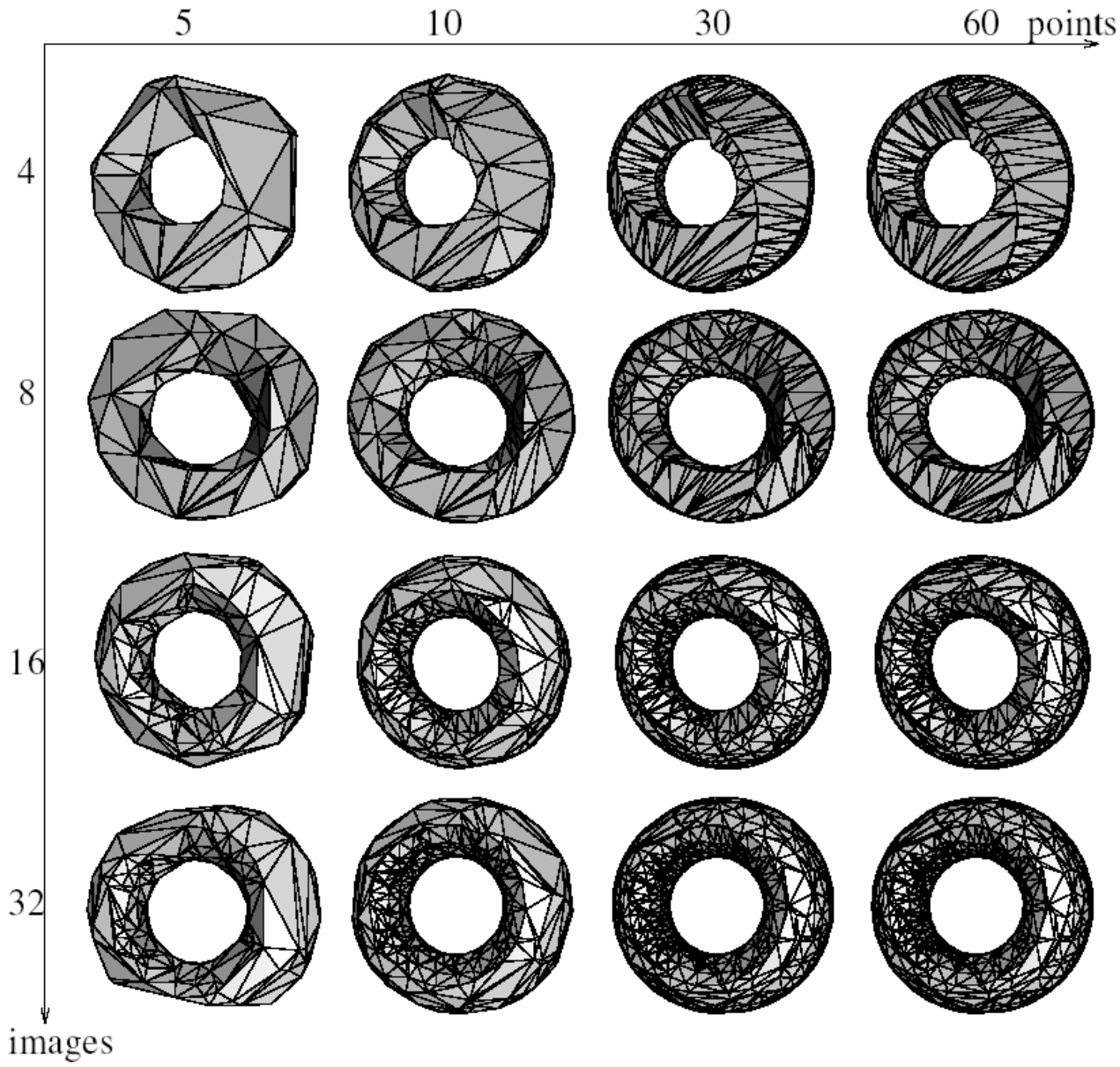
o 이미지의 개수가 일정 수준에 이르면 이미지 추가에 따른 정확도 향상이 많지 않음

; 이미지를 추가하면 물체 표면에 더 가까운 Visual Hull 점들이 추가됨

→ 몇 개의 Delaunay 사면체들이 표면에 더 가깝게 접근

- 결과 #3: 사람 모델

o 외곽선은 [Rennesson95]의 방법으로 추출


5
10
30
60 points
4
8
16
32
images

6. Conclusion

- 윤곽선이 주어졌을 때, 복잡한 장면의 Visual Hull 을 계산하는 방법 제안
- Voxel 기반 방식과 표면 기반 방식의 혼합 알고리즘
  - o Visual Hull 의 표면에 있는 점들을 계산
    ; Visual Hull 의 여집합을 구성하는 점들을 계산
  - o Delaunay 삼각화로 Visual Hull 의 표면을 추출
    ; 윤곽선 내부에 투영되는 다면체로 구성되는 표면 생성
- 기존 Voxel 기반 방식보다 낮은 시간과 공간 복잡도로 정확도면에서 나은 결과를 보임
- 향후 연구
  - o 다른 사면체 제거의 기준을 연구
  - o Delaunay 삼각화의 수정
  - o PC 클러스터링을 통한 실시간 모델링 방법 구현이 도전적인 과제