(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2022년12월09일
(11) 등록번호 10-2475218
(24) 등록일자 2022년12월02일

(51) 국제특허분류(Int. Cl.)
G06T 17/20 (2006.01) *G06N 20/00* (2019.01)
G06T 7/90 (2017.01) *G06V 10/40* (2022.01)
(52) CPC특허분류
G06T 17/205 (2013.01)
G06N 20/00 (2021.08)
(21) 출원번호 10-2022-0016285
(22) 출원일자 2022년02월08일
심사청구일자 2022년02월08일
(56) 선행기술조사문헌
JP2020115336 A

(73) 특허권자
한발대학교 산학협력단
대전광역시 유성구 동서대로 125 (덕명동)
(72) 발명자
이승호
대전광역시 유성구 엑스포로123번길 26-30 스마트시티리버뷰오피스텔 102동 601호
김정윤
대전광역시 유성구 학하서로121번길 150
(74) 대리인
이은철

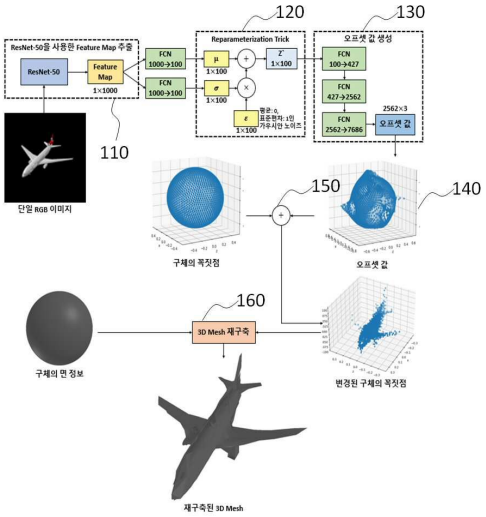
전체 청구항 수 : 총 5 항 심사관 : 강석제

(54) 발명의 명칭 **딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 방법**

(57) 요 약

본 발명은 3D Mesh 재구축 방법에 관한 것으로, 딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 방법에 관한 것이다. 본 발명에 따른 딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 방법은 3D 모델 데이터를 얻기 위해 사용되던 기존의 방식에 비해 단일 이미지만을 사용하여 3D 모델 데이터를 생성할 수 있어 가격이 저렴하고 휴대성, 확장성이 높고 소형화에 유리하며 무엇보다도 단일 이미지로 데이터를 획득하기 쉽다는 큰 장점이 있다.

대 표 도 - 도1



(52) CPC특허분류
G06T 7/90 (2017.01)
G06V 10/40 (2022.01)
G06T 2207/20081 (2013.01)

이 발명을 지원한 국가연구개발사업
과제고유번호 1711171377
과제번호 2022R1F1A1066371
부처명 과학기술정보통신부
과제관리(전문)기관명 한국연구재단
연구사업명 개인기초연구(과기정통부)
연구과제명 메타버스 컨텐츠에서 사용할 수 있는 음성에 따른 자연스러운 감정 표현이 가능한 3D 얼굴 자동 생성 솔루션 개발
기 여 율 1/1
과제수행기관명 한밭대학교
연구기간 2022.06.01 ~ 2025.02.28

공지예외적용 : 있음

명 세 서

청구범위

청구항 1

(a)입력 RGB 이미지에 대한 특징점인 Feature Map을 추출하는 단계,

(b)상기 추출된 Feature Map를 Latent Vector로 변환하는 단계,

(c)상기 Latent Vector를 FCN의 입력 데이터로 사용하여 오프셋 값을 생성하는 단계,

(d)상기 오프셋 값을 적용한 구체의 형태 변형 및 3D Mesh 재구축 단계,

(e)재구축된 3D Mesh를 학습 데이터셋의 3D Mesh와 비교하여 두 데이터 간의 점유공간의 유사도, 상기 재구축된 3D Mesh의 면 연결정도, 3D Mesh를 샘플링하여 생성하는 3D 포인트 클라우드 간의 거리 오차를 산출하는 단계 및

(f)상기 (e)단계의 각 항목별로 가중치를 적용하고 더한 값을 이용하여 상기 오차를 줄이도록 학습하는 단계를 포함하며,

상기 (d)단계에서 단일 RGB 이미지로부터 추출한 Feature Map을 Latent Vector로 변환하여 FCN을 통해 출력된 값을 구체의 꼭짓점의 위치를 변경시키기 위한 오프셋 값을 이용하며, 오프셋 값과 구체의 꼭짓점에 사이에 덧셈 연산을 수행하여 구체의 꼭짓점의 위치를 변경시킨 후, 구체의 면 정보를 적용하여 3D Mesh를 재구축하는 것을 특징으로 하는 딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 방법.

청구항 2

제1항에 있어서,

상기 (a)단계의 Feature Map의 내용은 입력된 RGB 이미지가 어떤 모양의 사물을 촬영한 것인지, 어떤 방향으로 촬영되었는지, 어떤 색상인지로 인해 달라지는 것을 특징으로 하는 딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 방법.

청구항 3

제1항에 있어서,

상기 (b)단계는 Feature Map을 크기가 축소된 Latent Vector로 변환하는 것을 특징으로 하는 딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 방법.

청구항 4

제1항에 있어서,

상기 (e)단계는

재구축된 3D Mesh를 학습 데이터셋의 3D Mesh와 비교하여 두 데이터 간의 점유공간의 유사도를 나타내는 IoU Loss와 재구축된 3D Mesh의 면의 연결 정도를 나타내는 Laplacian Smoothing Loss를 계산하고, 이후, 3D Mesh를 샘플링하여 생성한 2개의 3D 포인트 클라우드 간의 거리 오차를 나타내는 Chamfer Distance Loss를 계산하는 것을 특징으로 하는 딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 방법.

청구항 5

제1항에 있어서,

상기 (f)단계는

IoU Loss와 Laplacian Smoothing Loss, Chamfer Distance Loss에 각각 가중치를 적용하고 더한 값을 이용하여 ResNet-50과 FCN을 학습하되, ResNet-50과 FCN은 3개의 Loss를 모두 0에 가까워지도록 만드는 방향으로 학습하

는 것을 특징으로 하는 딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 3D Mesh 재구축 방법에 관한 것으로, 보다 상세하게는 딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 방법에 관한 것이다.

배 경 기 술

[0002] 기존의 3차원 CNC 가공 분야를 포함하여 4차 산업혁명을 시작으로 크게 성장하고 있는 가상현실(VR), 증강현실(AR) 및 3D 프린팅 분야 등에서 3D 모델 데이터는 활용 가치가 매우 높은 필수적인 정보라 할 수 있다.

[0003] 대표적인 3D 모델 데이터의 예로는 특정 사물의 표면을 따라 일정 간격으로 점을 찍듯이 스캔하여 무수히 많은 점들의 x, y, z 좌표들로 이루어진 3D 포인트 클라우드가 있으며, 이 점들을 그물처럼 엮어서 하나로 이어진 그물망처럼 표현하는 기법인 3D Mesh 방식이 있다.

[0004] 또한, N^3 크기의 3D 배열을 이용하여 수많은 정육면체의 조합으로 3D 모델 데이터를 표현하는 3D 복셀 방식이 있다. 이러한 3D 모델 데이터를 얻기 위하여 여러 대의 카메라를 부착한 부스를 이용한 3D 스캐너나 빛을 이용하는 LiDAR, 스테레오 카메라 등을 이용한 3D 스캐너 등이 개발되고 사용되어왔다. 그러나 많은 비용이 필요하고 넓은 공간이 필요하며 스캔하고자하는 물체의 크기에 많은 제약이 있었다. 최근에는 딥러닝 네트워크를 활용한 단일 이미지로부터 3D 모델 데이터를 재구축하는 연구가 꾸준히 진행되고 있다.

선행기술문헌

특허문헌

[0005] (특허문헌 0001) 공개특허 10-2021-0073442 (공개일 2021.06.18.)

발명의 내용

해결하려는 과제

[0006] 본 발명은 상술한 문제를 해결하고자 고안한 것으로, 기존에 여러 대의 카메라를 부착한 부스에서 촬영한 여러 장의 RGB 이미지나 빛을 이용하는 LiDAR, 스테레오 카메라 등을 이용한 3D 스캐너 등을 필요로 했던 기존의 3D 모델 획득 방식에서 딥러닝과 구체의 형태 변형 방법을 사용한 3D Mesh 재구축 방법을 제안하였다.

[0007] 또한 딥러닝을 이용하면 한 방향에서 사물을 촬영한 단편적인 정보만으로도 보이지 않는 부분의 형태를 예측할 수 있고 3D Mesh를 생성할 때 가장 어려운 문제인 면 생성 문제를 구체의 형태 변형 방법을 이용하여 해결하고자 하였다.

과제의 해결 수단

[0008] 본 발명에 따른 딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 방법은 (a)입력 RGB 이미지에 대한 특징점인 Feature Map을 추출하는 단계, (b)상기 추출된 Feature Map를 Latent Vector로 변환하는 단계, (c)상기 Latent Vector를 FCN의 입력 데이터로 사용하여 오프셋 값을 생성하는 단계, (d)상기 오프셋 값을 적용한 구체의 형태 변형 및 3D Mesh 재구축 단계, (e)재구축된 3D Mesh를 학습 데이터셋의 3D Mesh와 비교하여 두 데이터 간의 점유공간의 유사도, 상기 재구축된 3D Mesh의 면 연결정도, 3D Mesh를 샘플링하여 생성하는 3D 포인트 클라우드 간의 거리 오차를 산출하는 단계 및 (f)상기 (e)단계의 각 항목별로 가중치를 적용하고 더한 값을 이용하여 상기 오차를 줄이도록 학습하는 단계를 포함한다.

[0009] 상기 (a)단계의 Feature Map의 내용은 입력된 RGB 이미지가 어떤 모양의 사물을 촬영한 것인지, 어떤 방향으로 촬영되었는지, 어떤 색상인지로 인해 달라진다.

[0010] 상기 (b)단계는 Feature Map을 크기가 축소된 Latent Vector로 변환한다.

[0011] 상기 (e)단계는 재구축된 3D Mesh를 학습 데이터셋의 3D Mesh와 비교하여 두 데이터 간의 점유공간의 유사도를 나타내는 IoU Loss와 재구축된 3D Mesh의 면의 연결 정도를 나타내는 Laplacian Smoothing Loss를 계산하고, 이후, 3D Mesh를 샘플링하여 생성한 2개의 3D 포인트 클라우드 간의 거리 오차를 나타내는 Chamfer Distance Loss를 계산한다.

[0012] 상기 (f)단계는 IoU Loss와 Laplacian Smoothing Loss, Chamfer Distance Loss에 각각 가중치를 적용하고 더한 값을 이용하여 ResNet-50과 FCN을 학습하되, ResNet-50과 FCN은 3개의 Loss를 모두 0에 가까워지도록 만드는 방향으로 학습한다.

발명의 효과

[0013] 본 발명의 일 실시예에 따르면, 3D 모델 데이터를 얻기 위해 사용되던 기존의 방식에 비해 단일 이미지만을 사용하여 3D 모델 데이터를 생성할 수 있어 가격이 저렴하고 휴대성, 확장성이 높고 소형화에 유리하며 무엇보다도 단일 이미지로 데이터를 획득하기 쉽다는 큰 장점이 있다.

[0014] 기존의 근처의 가까운 점들을 연결하여 모서리 또는 면을 구축하는 방식과 다르게 딥러닝 네트워크을 통하여 구체의 꼭짓점의 위치를 사물의 3D 포인트 클라우드와 매우 유사하게 수정한다. 3D 포인트 클라우드를 이용하므로 메모리가 적게 필요하며 구체의 꼭짓점에 오프셋 값 사이에 덧셈 연산만을 수행하기 때문에 연산량이 적다.

[0015] 수정한 꼭짓점에 구체의 면 정보를 씌워 3D Mesh를 재구축한다. 구체의 꼭짓점의 위치를 수정하여 생성한 3D 포인트 클라우드의 점들의 간격이 일정하지 않을 때에도 이미 점들 사이의 연결 여부를 나타내는 구체의 면 정보라는 3D Mesh의 면 정보를 가지고 있어 표현의 단순화나 결손을 방지할 수 있다.

도면의 간단한 설명

[0016] 도 1은 본 발명의 일 실시예에 따른 딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 방법의 전체 개요도이다.

도 2는 본 발명의 일 실시예에 따른 딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 방법의 학습과정에 대한 흐름도이다.

도 3은 CNN과 ResNet에서 사용되는 Residual learning의 구조의 차이를 나타낸다.

도 4 내지 도 6은 ResNet-50을 사용한 Feature Map의 추출과정을 설명하기 위한 도면이다.

도 7 내지 도 8은 Reparameterization Trick을 사용한 Latent Vector 변환과정을 설명하기 위한 도면이다.

도 9 내지 도 10은 FCN을 사용한 오프셋 값 생성과정을 나타낸 도면이다.

도 11 내지 14는 본 실시예에 따른 3D Mesh 재구축을 설명하기 위한 도면이다.

도 15 내지 16은 IoU Loss, Laplacian Smoothing Loss, Chamfer Distance Loss 계산과정을 설명하기 위한 도면이다.

도 17은 Laplacian Smoothing Loss가 3D Mesh 재구축에 미치는 영향을 나타낸 도면이다.

도 18은 두 3D 포인트 클라우드 간의 가장 가까운 점들의 예시를 나타낸다.

도 19는 본 발명의 일 실시예에 따른 시스템의 실행과정을 나타낸 흐름도이다.

도 20 내지 24는 본 발명에서 실험결과를 설명하기 위한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0017] 본 발명의 실시예에서 제시되는 특정한 구조 내지 기능적 설명들은 단지 본 발명의 개념에 따른 실시예를 설명하기 위한 목적으로 예시된 것으로, 본 발명의 개념에 따른 실시예들은 다양한 형태로 실시될 수 있다. 또한, 본 명세서에 설명된 실시예들에 한정되는 것으로 해석되어서는 아니 되며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경물, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.

[0018] 한편, 본 발명에서 제1 및/또는 제2 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 상기 구성요소들은 상기 용어들에 한정되지는 않는다. 상기 용어들은 하나의 구성요소를 다른 구성요소들과 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않는 범위 내에서, 제1 구성요소는 제2 구

성요소로 명명될 수 있고, 유사하게 제2 구성요소는 제1 구성요소로도 명명될 수 있다.

[0019] 이하 첨부된 도면을 참조하여 본 발명의 실시예를 설명한다. 본 발명의 실시예를 설명함에 있어서, 관련된 공지 기능 혹은 구성에 대한 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우 그 설명을 생략하였다.

[0020] 도 1은 본 발명의 일 실시예에 따른 딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 방법의 전체 개요도이다.

[0021] 본 발명에서 제안하는 기법의 개요는 도 1과 같다. 본 발명에서 제안하는 기법은 단일 RGB 이미지만을 사용하여 3D Mesh를 재구축한다. 제안된 기법은 딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 시스템의 추출부(110), 변환부(120), 생성부(130), 재구축부(140), 계산부(150), 학습부(160) 구성을 포함한다. 이러한 구성으로 다음의 과정으로 구성되며, 추출부(110)의 ResNet-50을 사용한 Feature Map 추출 과정, 변환부(120)의 Reparameterization Trick을 사용한 Latent Vector 변환 과정, 생성부(130)를 통한 FCN을 사용한 오프셋 값 생성 과정, 재구축부(140)를 통한 오프셋 값을 적용한 구체의 형태 변형 및 3D Mesh 재구축 과정, 계산부(150)의 IoU Loss, Laplacian Smoothing Loss, Chamfer Distance Loss 계산 과정, 계산한 Loss에 가중치를 적용한 학습부(160)의 ResNet-50, FCN 학습 과정으로 구분된다.

[0022] ResNet-50을 사용한 Feature Map 추출 과정에서는 입력 RGB 이미지에 대한 특징점인 Feature Map을 추출한다. Feature Map의 내용은 입력된 RGB 이미지가 어떤 모양의 사물을 촬영한 것인지, 어떤 방향으로 촬영되었는지, 어떤 색상인지 등으로 인해 달라진다.

[0023] Reparameterization Trick을 사용한 Latent Vector 변환 과정에서는 추출된 Feature Map를 Latent Vector로 변환한다. Feature Map을 크기가 축소된 Latent Vector로 변환하는 것의 이점은 각 RGB 이미지의 고차원의 데이터를 저차원으로 변환하여 불확실성을 줄이고 유사한 데이터의 유사성을 더욱 높일 수 있다는 것이다. 결과적으로 의자가 촬영된 모든 RGB 이미지의 Latent Vector는 유사해진다. 이는 같은 카테고리의 사물이 가진 공통적인 특징을 빠르게 파악하도록 하여 딥러닝 네트워크의 학습 횟수를 단축시켜준다.

[0024] FCN을 사용한 오프셋 값 생성 과정에서는 Latent Vector를 FCN의 입력 데이터로 사용하여 오프셋 값을 생성한다. 본 발명에서 형태를 변형시키고자 하는 구체의 꼭짓점 배열의 크기가 2562, 3이기 때문에 2562 x 3 크기인 7686까지 증가를 시킨다. 이후, 최종출력을 2562, 3 형태로 변환하여, 이를 구체의 꼭짓점 위치를 변경시키기 위한 오프셋 값으로 활용한다.

[0025] 오프셋 값을 적용한 구체의 형태 변형 및 3D Mesh 재구축 과정에서는 오프셋 값을 구체의 꼭짓점에 적용하여 꼭짓점의 위치를 변경시킨 후, 구체의 면 정보를 그대로 적용하여 3D Mesh를 재구축한다.

[0026] IoU Loss, Laplacian Smoothing Loss, Chamfer Distance Loss 계산 과정에서는 재구축된 3D Mesh를 학습 데이터셋의 3D Mesh와 비교하여 두 데이터 간의 점유공간의 유사도를 나타내는 IoU Loss와 재구축된 3D Mesh의 면의 연결 정도를 나타내는 Laplacian Smoothing Loss를 계산한다. 이후, 3D Mesh를 샘플링하여 생성한 2개의 3D 포인트 클라우드 간의 거리 오차를 나타내는 Chamfer Distance Loss를 계산한다.

[0027] 계산한 Loss에 가중치를 적용한 ResNet-50, FCN 학습 과정에서는 1-IoU Loss와 Laplacian Smoothing Loss, Chamfer Distance Loss 등에 각각 가중치를 적용하고 더한 값을 이용하여 ResNet-50과 FCN을 학습한다. ResNet-50과 FCN은 3개의 Loss를 모두 0에 가까워지도록 만드는 방향으로 학습된다.

[0028] 본 발명에서 제안된 기법의 객관적인 신뢰성을 평가하기 위해 공개된 표준 데이터베이스인 ShapeNet 데이터셋을 이용하여 다른 발명에서 발표한 기법들과 비교 평가하였다. 실험결과, 본 발명에서 제안하는 기법의 IoU 값이 0.581로 다른 발명에서 발표한 기법들보다 낮은 수치를 나타내었다. IoU 값은 수치가 높을수록 우수한 3D Mesh 재구축 기법을 나타내므로, 다른 발명에서 발표한 기법들보다 높은 수치를 나타내어 그 효율성이 입증되었다. 또한, 본 발명에서 제안하는 기법의 Chamfer Distance 값이 0.212로 다른 발명에서 발표한 기법들보다 낮은 수치를 나타내었다. Chamfer Distance 값은 수치가 낮을수록 우수한 3D Mesh 재구축 기법을 나타내므로, 다른 발명에서 발표한 기법들보다 낮은 수치를 나타내어 그 효율성이 입증되었다.

[0029] 도 2는 본 발명의 일 실시예에 따른 딥러닝과 구체의 형태 변형을 이용한 단일 이미지에서의 3D Mesh 재구축 방법의 학습과정에 대한 흐름도이다. 이러한 도 2에 도시된 바와 같이, 재구축한 3D Mesh와 학습 데이터셋의 원본 3D Mesh 사이의 차이를 계산하여 학습을 조정하고 단일 이미지로부터 생성된, 구체의 꼭짓점 위치를 변경하기 위한 오프셋 값을 최적화하여 구체가 원본 3D Mesh에 가깝게 변형되도록 한다.

[0030] ResNet-50을 사용한 Feature Map 추출단계에서, ResNet은 ImageNet 2012 데이터셋의 분류를 위해 만들어진 딥러닝 네트워크로 입력 층의 특징이 유실되지 않고 심층까지 효과적으로 전달되는 특징이 있다. 도 3은 CNN과 ResNet에서 사용되는 Residual learning의 구조의 차이를 나타낸다.

[0031] CNN은 x를 입력받고 2개의 Weight layer를 거친 후 H(x)라는 출력을 내며, 학습을 통해 최적의 H(x)를 찾기 위해 Weight layer의 값들을 수정한다. 그러나 ResNet은 도 3의 Residual learning의 구조처럼 H(x) = F(x) + x가 되면 F(x)= H(x) - x가 된다. 이때의 최적의 F(x) = 0이다. 그 이유는 F(x)가 거의 0이 되는 방향으로 학습하게 되면 입력의 작은 차이도 쉽게 검출할 수 있게 되기 때문이다. 작은 차이(Residual)를 학습한다는 의미에서 Residual learning이라고 한다. 이런 특징으로 인해 입력 데이터의 중요한 특징을 추출하는 것에 높은 성능을 발휘할 수 있다.

[0032] 본 발명에서는 50층으로 구성된 ResNet-50을 사용하여 128^{X}128 크기의 입력 RGB 이미지로부터 1^{X}1000 크기의 Feature Map을 추출한다. 1^{X}1000 크기인 이유는 ResNet이 1000개의 카테고리를 분류하도록 설계되어 출력 크기가 1^{X}1000인 이유도 있으나, 후술할 Reparameterization Trick으로 크기를 축소시키기 위해서는 충분히 많은 특징들이 필요하기 때문에 그대로 사용하였다.

[0033] 도 4는 ResNet-50을 사용한 Feature Map의 추출과정을 나타낸다.

[0034] ResNet-50의 각 층의 출력 값을 비선형으로 만드는 활성화함수로는 ReLU를 사용하였다. 입, 출력 값이 선형이면 1층만으로도 계산이 가능하여 깊은 층을 사용할 필요가 없으며 표현의 한계가 있다. 따라서 층을 깊게 구성하고 복잡한 표현이 가능하도록 하며 음수 값을 0으로 만들어 학습을 빠르게 만드는 ReLU를 활성화함수로 사용하였다. ReLU는 도 5와 같다.

[0035] 도 5는 ResNet-50의 활성화 함수로 사용되는 ReLU를 나타낸 그래프이다.

[0036] [표 1]은 본 발명에서 사용한 ResNet-50의 구조를 나타내며, 도 6은 본 발명에서 사용된 ResNet-50의 구조도이다.

[0037] (표1)

Group	Contents	Count	Output Size
conv1	7x7 conv, 64, stride2	x1	64, 64
conv2	3x3 max pool, stride2	x1	32, 32
	1x1 conv, 64	x3	32, 32
	3x3 conv, 64		
	1x1 conv, 256		
conv3	1x1 conv, 128	x4	16, 16
	3x3 conv, 128		
	1x1 conv, 512		
conv4	1x1 conv, 256	x6	8, 8
	3x3 conv, 256		
	1x1 conv, 1024		
conv5	1x1 conv, 512	x3	4, 4
	3x3 conv, 512		
	1x1 conv, 2048		
	average pool	x1	1, 1000

[0038]

[0039] 다음으로 Reparameterization Trick을 사용한 Latent Vector 변환을 설명하기로 한다.

[0040] Reparameterization Trick은 VAE에서 사용되는 기법으로 딥러닝 네트워크를 통해 추출한 특징을 저차원의 확률분포 형태의 Latent Vector로 변환하는 기법이다. 본 발명에서는 입력 데이터로 고차원의 데이터인 RGB 이미지를 사용하며, 추출된 Feature Map으로는 가능성이 너무 많아 불확실성이 높다. 그러나 저차원의 확률 분포로 변환하게 되면 가능성을 줄여, 데이터를 보다 분명하게 판단할 수 된다. Feature Map을 1 x 100 크기의 Latent Vector로 변환하기 위해 FCN을 이용하여 1 x 100 크기의 특징 2개를 추출하고 각각을 Latent Vector Z`를 생성하기 위한 평균 μ와 표준편차 σ로 사용한다. σ에 대하여 평균이 0, 표준편차가 1인 가우시안 노이즈 ε를 곱해서 크기를 제한하여 불확실성이 높은 Feature Map을 의미 있는 데이터로 만든다. Feature Map에서 추출한 평

균 μ와 $\varepsilon\sigma$를 더하여 확률 분포인 Latent Vector Z`를 생성한다. [식 1]은 Latent Vector Z`를 생성하는 식을 나타내며 도 7은 Reparameterization Trick을 사용한 Latent Vector 변환과정을 나타낸다.

[0041] (식1)

$$\varepsilon = N(0,1)$$
$$Z` = \mu + \varepsilon\sigma$$

[0042]

[0043] Reparameterization Trick을 사용한 Latent Vector변환에서 고차원의 단일 이미지로부터 추출된 Feature Map은 여전히 가능성이 무수히 많아 불확실성이 높은 데이터이다. 따라서, Reparameterization Trick을 사용하여 Feature Map을 저차원의 확률 분포 형태인 Latent Vetor Z'로 변환하여 판별이 용이하도록 하며, 가우시안 노이즈 ε로 표준편차 σ를 제한되어 가능성이 줄어들어 의미있는 데이터가 된다.

[0044] 다음으로 FCN을 사용한 오프셋 값 생성을 설명하기로 한다.

[0045] Feature Map을 저차원의 확률 분포 형태의 Latent Vector Z`로 변환한 뒤, 이를 FCN의 입력 데이터로 사용하여 오프셋 값으로 변환한다. 오프셋 값은 임의의 3D 포인트 클라우드의 점의 위치를 변경하기 위한 값으로써, 크기가 2562 x 3인 이유는 본 발명에서 3D Mesh 재구축을 위해 사용한 구체의 꼭짓점 개수가 2562개이며, 꼭짓점 하나마다 x, y, z 좌표를 갖고 있기 때문이다. 본 발명에서는 Latent Vector Z`를 FCN의 입력 데이터로 사용하여 2562 x 3 크기인 7686으로 크기를 증가시킨다. 여러 층에 걸쳐 크기를 증가시켜야 비선형성을 높여 성능을 향상시킬 수 있다. 생성된 7686개의 값 중에서 앞의 2562개는 x 좌표이며, 중간의 2562개는 y 좌표, 마지막 2562개를 z 좌표로 출력 크기를 2562 x 3로 변환한다. 오프셋 값은 학습과정에서 계속 모양이 수정되며 결과적으로 구체를 변형하여 재구축된 3D Mesh의 정확도가 가장 높은 모양으로 출력된다. 도 8은 FCN을 사용한 오프셋 값 생성과정을 나타낸다.

[0046] 도 9는 FCN을 사용한 오프셋 값 생성과정을 나타낸 도면이다. 본 발명에서 사용된 FCN의 구조는 도 9와 같다. 1개의 데이터를 정수 배로 크기를 증가시키기 위해 7686의 약수로 크기를 서서히 증가시켰다.

[0047] FCN의 활성화함수로는 LeakyReLU를 사용하였다. LeakyReLU는 양수 값에 대해서는 ReLU와 같지만 다른 점은 음수에 약간의 기울기를 곱하여 0이 아닌 작은 음수를 출력하여 세부적인 특징을 살린다. 출력 층은 0.5, -0.5로 정규화되어있는 학습 데이터 셋의 크기에 맞게 Tanh로 출력 크기를 제한하여 출력 값이 너무 커지거나 작아지는 것을 방지하여 학습 속도를 향상시킨다. 도 10은 FCN의 활성화 함수로 LeakyReLU와 Tanh를 나타낸 그래프이다.

[0048] 다음으로 오프셋 값을 적용한 구체의 형태 변형 및 3D Mesh 재구축을 설명하기로 한다.

[0049] 오프셋 값을 구체의 꼭짓점에 적용하여 구체의 형태를 변형시켜 3D Mesh를 재구축하는 기법은 본 발명에서 새롭게 제안하는 기법이다. 여러 가지 형태 중에 구체를 이용한 가장 큰 이유는 중심으로부터 모든 점이 똑같은 거리에 위치하고 있어 다양한 사물의 모양에 대응하기에 적합하기 때문이다.

[0050] 본 발명에서 제안하는 기법은 다음과 같은 장점이 있어 사용하였다. 3D Mesh 표면을 재구축하는 부분에 있어서 어느 꼭짓점끼리 연결하여 모서리, 혹은 면을 생성할지 결정하는 것은 굉장히 어려운 문제이다. 그러나 본 발명에서 제안하는 기법은 구체를 변형시키는 방법으로 이미 구체의 면 정보를 가지고 있다. 따라서 해당 면 정보에 최적화된 방향으로 꼭짓점의 위치를 수정하는 것이 가능하다면 면 생성에 관련된 문제를 해결할 수 있다. 본 발명에서는 구체가 가지고 있는 면 정보에 최적화된 꼭짓점 위치를 찾아내기 위한 방법으로 단일 이미지로부터 생성하는 오프셋 값을 이용하여 면 생성의 문제를 해결하였다. 또한, 오프셋 값과 구체의 꼭짓점 사이에 덧셈 연산만을 수행하기 때문에 적은 연산량으로 효과적인 3D Mesh 재구축이 가능하다. 이러한 이유로 제안하는 기법으로 3D Mesh 재구축을 수행하였다.

[0051] 본 실시예에 따른 시스템은 반지름이 0.5인 구체 생성의 경우, 중심의 좌표가 0, 0, 0이고 반지름이 0.5인 구체를 생성한다. 구체는 꼭짓점 개수가 2562개, 면의 개수가 5120개인 3D Mesh 모델을 사용한다. 도 11은 반지름이 0.5인 구체를 시각화한 것을 나타낸다. 본 발명에서는 3D Mesh 표면 생성에 대한 문제를 회피하기 위해 구체 3D Mesh의 형태를 변형하는 방법을 채택하였으며, 구체를 선택한 이유는 중심으로부터 모든 점이 같은 거리에 위치하기 때문이다.

[0052] 본 실시예에 따른 시스템은 오프셋 값을 이용한 구체의 꼭짓점 위치 수정에서, 해당 구체의 꼭짓점 배열의 n번째 x, y, z와 학습과정에서 생성된 오프셋 값 배열의 n번째 x, y, z 값 사이에 덧셈 연산을 하여 꼭짓점의 위치가 변경된 구체를 생성하게 된다. 도 12는 오프셋 값을 적용시켜 구체의 꼭짓점 위치를 변경하는 과정을 나타내며, 괄호 안의 수치는 점의 x, y, z 좌표이다. 좌표 값은 소수점 두 번째 자리에서 반올림하였다. 도 12에 도시된 바와 같이, 오프셋 값을 적용한 구체의 형태 변형 및 3D Mesh 재구축에서 단일 이미지로부터 생성된 오프셋 값을 구체의 꼭짓점에 적용하여 위치를 변경한다.

[0053] 본 실시예에 따른 시스템은 위치가 바뀐 꼭짓점에 면 정보를 씌워 3D Mesh를 재구축에 있어서, 꼭짓점과 면 정보를 이용한 Mesh 표면을 생성한다. 면 정보에는 꼭짓점 배열의 n번째 꼭짓점 3개를 이어서 삼각형이 구성된다는 정보가 들어있다. 예를 들어 면 정보가 7, 1, 28인 경우 꼭짓점 배열의 7번, 1번, 28번 점 3개를 이어 삼각형을 구성한다. 도 13은 꼭짓점과 면 정보를 이용한 Mesh 표면 생성의 예시를 나타낸다.

[0054] 도 14와 같이 오프셋 값을 적용한 변경된 구체의 꼭짓점에 면 정보를 적용하여 3D Mesh를 재구축한다.

[0055] 다음으로 본 실시예에 따른 IoU Loss, Laplacian Smoothing Loss, Chamfer Distance Loss 계산과정을 설명하기로 한다.

[0056] IoU Loss는 2개의 3D 모델 간의 점유공간의 유사도를 평가하기 위한 방법이다. 2개의 3D Mesh를 유사도를 비교할 때 꼭짓점의 개수와 면의 개수가 일치하지 않으면 둘 사이의 오차를 계산하는 기준이 모호해질 수 있다. 따라서 3D 격자 안의 3D Mesh가 차지하는 점유공간을 비교하여 재구축된 3D Mesh의 점유공간이 학습 데이터셋의 3D Mesh의 점유공간과 유사해지도록 만든다. [식 2]는 IoU Loss의 계산식을 나타내며 $Mesh$ 와 $Mesh$ 는 각각의 3D Mesh가 차지하는 점유공간들을 나타낸다.

[0057] (식2)

$$IoU_{Loss} = \frac{\sum Mesh_1 * Mesh_2}{\sum Mesh_1 + Mesh_2}$$

[0058]

[0059] 도 15는 IoU 계산을 도식화한 것을 나타낸다.

[0060] 본 발명에서 사용한 Laplacian Smoothing Loss의 정의는 다음과 같다. 만약 임의의 꼭짓점 i와 j가 같은 꼭짓점일 경우, -1, i와 j가 모서리를 이룰 경우, $1/\deg(i)$, 둘 다 해당되지 않을 경우에는 0을 출력한다. 이때의 deg(i)는 임의의 꼭짓점 i에 연결된 모서리의 개수를 의미한다. 모든 꼭짓점에 대해 Laplacian Smoothing Loss를 계산하여 평균값을 학습에 이용한다. Laplacian Smoothing Loss를 학습에 이용하는 이유는 꼭짓점 배열의 Index를 고려하지 않고 면 정보를 적용하면 3D Mesh가 엉망으로 재구축되기 때문에 꼭짓점 배열의 Index를 정렬하기 위함이다. 도 16은 두 꼭짓점을 이용한 Laplacian Smoothing Loss 계산의 예시를 나타낸다.

[0061] 도 17은 Laplacian Smoothing Loss가 3D Mesh 재구축에 미치는 영향을 나타낸 도면이다. 도 17에 도시된 바와 같이, IoU Loss, Chamfer Distance Loss, Laplacian Smoothing Loss를 학습에 이용하여 비행기의 단일 이미지를 3D Mesh로 재구축한 (a)와 Laplacian Smoothing Loss만을 제외하고 학습하여 재구축한 3D Mesh (b)의 차이를 나타낸다. (a)는 꼭짓점 배열의 Index가 면 정보를 적용하기에 최적의 순서로 정렬되어 정상인 3D Mesh가 재구축된 반면, (b)는 꼭짓점 배열의 Index가 제대로 정렬되지 않았기 때문에 구체의 면 정보를 적용하였을 때 전혀 다른 모양이 출력된다.

[0062] Chamfer Distance Loss는 2개의 3D 포인트 클라우드 간의 유사도를 평가하기 위한 방법이다. 본 발명에서 제안하는 기법으로 재구축한 3D Mesh와 학습 데이터셋의 3D Mesh를 일정간격으로 똑같은 개수의 점을 샘플링하여 3D 포인트 클라우드를 얻는다. 이후, 두 3D 포인트 클라우드 간의 Chamfer Distance Loss를 계산한다. Chamfer Distance Loss를 학습에 이용하는 이유는 2개의 3D Mesh에서 일정간격으로 샘플링한 3D 포인트 클라우드의 거리 오차가 0에 가까워지도록 만들어 윤곽을 유사하게 만들기 위함이다. 본 발명에서는 3D Mesh의 꼭짓점에서 Chamfer Distance를 계산하지 않고 샘플링한 3D 포인트 클라우드의 Chamfer Distance를 계산한다. 이와 같은 방식의 장점은 3D Mesh의 윤곽의 유사도를 높이는 방향으로 학습되어 변경된 구체의 꼭짓점의 간격과 밀도가 일정하지 않아도 정확도 높은 3D Mesh를 재구축할 수 있다. [식 3]은 Chamfer Distance Loss의 계산식을 나타내며, v1과 v2는 재구축한 3D Mesh와 학습 데이터셋을 일정간격으로 샘플링하여 얻은 3D 포인트 클라우드를

나타낸다.

[0063] (식3)

$$ChamferDistance(v_1, v_2) = \sum_{x \in v_1} \min_{y \in v_2} \| x - y \|_2^2 + \sum_{y \in v_2} \min_{x \in v_1} \| x - y \|_2^2$$

[0064]

[0065] Chamfer Distance Loss는 비교할 2개의 3D Mesh의 윤곽을 따라 일정 간격으로 샘플링한 점들의 거리 오차를 숫자로 나타낸 값으로 0에 가까울수록 오차가 적으므로, 재구축된 3D Mesh의 윤곽을 원본과 유사하게 만들기 위해 사용한다.

[0066] 도 18은 두 3D 포인트 클라우드 간의 가장 가까운 점들의 예시를 나타낸다. 괄호안의 값은 Chamfer Distance Loss의 수치이다.

[0067] 다음으로 계산한 Loss에 가중치를 적용하여 ResNet-50, FCN 학습을 설명하면 다음과 같다.

[0068] 본 발명에서는 IoU Loss, Laplacian Smoothing Loss, Chamfer Distance Loss가 학습에 미치는 영향을 최적화하기 위해 각각의 Loss에 weight를 곱하여 사용하였다. [식 4]는 본 발명의 ResNet-50, FCN 등을 학습할 때 사용되는 $Loss_{total}$을 나타낸다. 이때 α, β, γ 등은 학습과정을 반복적으로 수행하여 최적의 결과를 얻기 위하여 계속 변경하며, 최적의 결과를 얻었던 weight 값으로 최종 결정된다.

[0069] (식4)

$$Loss_{total} = \alpha*(1 - IoU_{Loss}) + \beta* ChamferDistance_{Loss} + \gamma* LaplacianSmoothing_{Loss}$$

[0070]

[0071] 본 실시예에 따른 시스템은 계산한 Loss에 가중치를 적용하여 ResNet-50, FCN 학습에 있어서, 계산한 Loss에 가중치 α, β, γ를 부여하여 학습에 어떤 Loss가 더 많이 반영될지를 결정한다. 본 발명에서는 점유 공간의 유사도인 IoU Loss의 반영 비율을 높이기 위해 α를 최대치로 설정한 뒤에 β와 γ의 조정을 수행하였다. α=0.1, β=1.0, γ=0.009로 최적화된 수치를 얻는다.

[0072] IoU_{Loss}의 경우 $ChamferDistance_{Loss}$와 $LaplacianSmoothing_{Loss}$와 다르게 1에 가까울수록 좋으며 최댓값이 1이기 때문에 $1-IoU_{Loss}$로 IoU_{Loss}를 최대화하는 방향으로 학습되도록 했다. 본 발명에서는 재구축된 3D Mesh와 원본 3D Mesh 사이의 점유공간의 유사도를 높이는 것에 중점을 두어 $1-IoU_{Loss}$의 영향이 가장 크게 나타나도록 하였다. 하나의 Loss가 지나치게 큰 경우 작은 Loss들은 학습에 영향을 주지 못하기 때문에, IoU_{Loss}의 크기를 다른 Loss들보다 큰 값으로 유지하면서 정확도 또한 개선시킬 수 있는 α의 최적의 크기를 알아내기 위해 α값을 1.0으로 초기화하고 서서히 줄이며 학습했다. 이때의 $ChamferDistance_{Loss}$와 $LaplacianSmoothing_{Loss}$가 비슷한 크기를 갖도록 β는 1.0, γ는 0.01로 초기화했다. 이렇게 도출된 α값 0.1을 고정하고 $ChamferDistance_{Loss}$와 $LaplacianSmoothing_{Loss}$의 조정을 수행했다. $ChamferDistance_{Loss}$의 크기를 증가시킨 경우, $LaplacianSmoothing_{Loss}$의 영향이 줄어들어 꼭짓점 배열의 Index가 제대로 정렬되지 않아 3D Mesh가 정상적으로 재구축되지 않았다. $ChamferDistance_{Loss}$의 크기를 감소시킬 경우, 학습에 걸리는 시간이 늘어났으며 더 많은 횟수를 학습해도 전보다 작은 값으로 수렴하지 못하고 0.003 부근에서 멈췄다. $LaplacianSmoothing_{Loss}$의 크기를 증가시킬 경우, 1/deg(i)의 값이 최소화되도록 각각의 꼭짓점에 최대한 많은 모서리를 연결하려는 방향으로 학습되어 비행기 날개의 끝부분과 같이 연결되는 면이 적게 필요한 부분의 표현이 제대로 되지 않는 문제가 나타났다. $LaplacianSmoothing_{Loss}$의 크기를 감소시킬 경우, 각각의 꼭짓점에 불필요하게 연결된 모서리가 올바른 짝을 찾아가는 양상을 보였다. γ가 0.01 ~ 0.008 정도일 때 좋은 결과를 얻어 중간 값인 0.009로 설정했다. 따라서 최적의 결과를 얻은 weight 값인 $\alpha = 0.1$, $\beta = 1.0$, $\gamma = 0.009$를 IoU_{Loss}, $ChamferDistance_{Loss}$, $LaplacianSmoothing_{Loss}$에 각각 곱하여 본 발명에서 사용되는 ResNet-50, FCN을 학습시켰다. 각각의 Loss에 weight를 적용하면 IoU_{Loss}는 0.058, $ChamferDistance_{Loss}$는 0.0021,

$LaplacianSmoothing_{Loss}$는 0.00135로 IoU_{Loss}를 제외하면 비슷한 크기이다. 2000 Epoch 학습을 수행한 뒤 IoU와 Chamfer Distance가 목표한 값에 도달하지 못하면 weight를 수정한 후에, 2000 Epoch 학습하는 것을 반복하여 최적의 weight를 도출했다. 1 Epoch이란 학습 데이터셋의 전체 데이터에 대해 1번 학습하는 것을 의미한다. 한편, ResNet-50과 FCN은 $Loss_{total}$의 크기와 관계없이 $Loss_{total}$을 0에 가깝게 최소화하는 방향으로 학습이 되기 때문에 $Loss_{total}$을 정규화하여 1로 만들지 않아도 된다.

[0073] 학습 과정은 IoU와 Chamfer Distance가 목표 값에 도달할 때까지 반복된다.

[0074] 도 19는 본 발명의 일 실시예에 따른 시스템의 실행과정을 나타낸 흐름도이다.

[0075] 실행 과정은 학습 과정에서 학습된 ResNet-50과 FCN을 이용하여 수행된다. 실행 과정에서는 학습 과정에서 학습된 ResNet-50과 FCN을 이용하여 단일 RGB 이미지로부터 오프셋 값을 생성하고 오프셋 값을 구체에 적용하여 형태를 변형시켜 3D Mesh 재구축을 완료한다.

[0076] 본 실시예에 따른 실험 환경은 다음과 같다.

[0077] <하드웨어>

[0078] ○ Intel(R) Core(TM) i7-10700 3.80GHz CPU

[0079] ○ RAM 16GB

[0080] ○ NVIDIA GeForce RTX 3070(VRAM 8GB)

[0081] <운영체제>

[0082] ○ Window 10 Pro 64비트

[0083] <개발도구>

[0084] ○ Visual Studio Code

[0085] ○ Python 3.6

[0086] <라이브러리>

[0087] ○ Pytorch 1.9.1

[0088] ○ CUDA 11.1

[0089] ○ cuDNN 8.2.1

[0090] 본 발명에서는 제안하는 구체의 형태 변형 방법을 사용하여 RGB 이미지로부터 재구축된 3D Mesh의 정확도를 평가하기 위해 ShapeNet 데이터셋을 이용하여 ResNet-50과 FCN을 학습했다.

[0091] ShapeNet 데이터셋은 Princeton, Stanford 및 TTIC의 연구원들이 협업하여 56개의 카테고리의 사물들로 구축하여 공개된 표준 데이터셋으로서, 기존의 논문들에서 비교 벤치마크로 사용되었다. ShapeNet 데이터셋은 56개의 카테고리를 가진 444,723개의 3D 모델로 이루어져 있다. 본 발명에서는 비교 논문들에서 수행한 결과를 바탕으로 똑같은 환경을 구성하기 위해 [표 2]와 같이 13개의 카테고리를 사용하였다. 또한, 데이터셋을 80 : 20으로 분할하여 학습을 위한 Train Set을 80으로 구성하고 딥러닝 네트워크의 유효성을 검사하기 위해 Validation Set을 20으로 구성하였다. 도 20은 본 발명에서 실험에 사용한 ShapeNet 데이터셋의 예시 이미지를 나타내고 있다.

[0092] (표2)

[0093]

분류	내용
사물	벤치, 의자, 램프, 스피커, 총기, 탁자, 수납장, 모니터, 긴 의자, 휴대폰
탈것	선박, 비행기, 자동차

[0094] 성능평가를 위한 값으로는 IoU, Chamfer Distance를 사용한다. IoU는 비교할 2개의 3D Mesh 사이의 점유 공간의

유사도를 숫자로 나타낸 값으로 1에 가까울수록 오차가 적다.

[0095] Chamfer Distance는 비교할 2개의 3D Mesh의 윤곽을 따라 일정 간격으로 샘플링한 점들의 거리 오차를 숫자로 나타낸 값으로 0에 가까울 수록 오차가 적다.

[0096] 도 21은 ShapeNet 데이터셋의 단일 이미지에서의 3D Mesh 재구축 결과를 나타낸다. 학습에 사용한 Training sets 이외의 약 8762개의 사물에 대한 Validation sets을 이용하여 3D Mesh 재구축을 수행한다.

[0097] 본 발명에서 제안하는 딥러닝과 구체의 형태 변형 방법을 이용한 단일 이미지에서의 3D Mesh 재구축 기법의 객관적인 성능을 평가하기 위하여 공개된 표준 데이터셋인 ShapeNet 데이터셋을 이용하여 다른 논문에서 발표한 기법들과 비교 평가했다. 정확도의 평가 기법으로는 IoU, Chamfer Distance를 사용하였다. [표 3]은 본 발명에서 제안하는 기법과 다른 논문에서 발표한 기법들의 ShapeNet 데이터셋 비교 결과를 나타내며 Chamfer Distance는 100을 곱하여 소수점 첫째자리로 조정하였다. 본 발명에서 제안하는 기법은 구체의 형태를 변형한 3D Mesh를 재구축을 수행하였으며 IoU는 0.581, Chamfer Distance 0.212로 우수한 결과를 산출하였다. 단, AtlasNet은 다른 논문들과 다르게 RGB 이미지뿐만 아니라 3D 포인트 클라우드를 입력 데이터로 사용하여 학습하였기 때문에 단일 이미지만을 사용한 방법들 중 본 발명에서 제안하는 기법이 가장 우수하다.

[0098] (표 3)

ShapeNet Dataset		
Method	**IoU ↑**	**Chamfer Distance ↓**
3D-R2N2	0.493	0.278
PSGN	-	0.215
Pix2Mesh	0.480	0.216
AtlasNet	-	**0.175**
OccupancyNet	0.571	0.215
The Proposed Method	**0.581**	0.212

[0099]

[0100] 도 22는 IoU를 비교 평가한 결과를 그래프로 나타내고 있다. 본 발명에서 제안하는 기법의 IoU 값이 다른 논문에서 발표한 기법들보다 높은 수치를 나타내었다. IoU 값은 수치가 1에 가까울수록 우수함을 나타내므로, 다른 논문에서 발표한 기법들보다 우수한 것이 확인되었다.

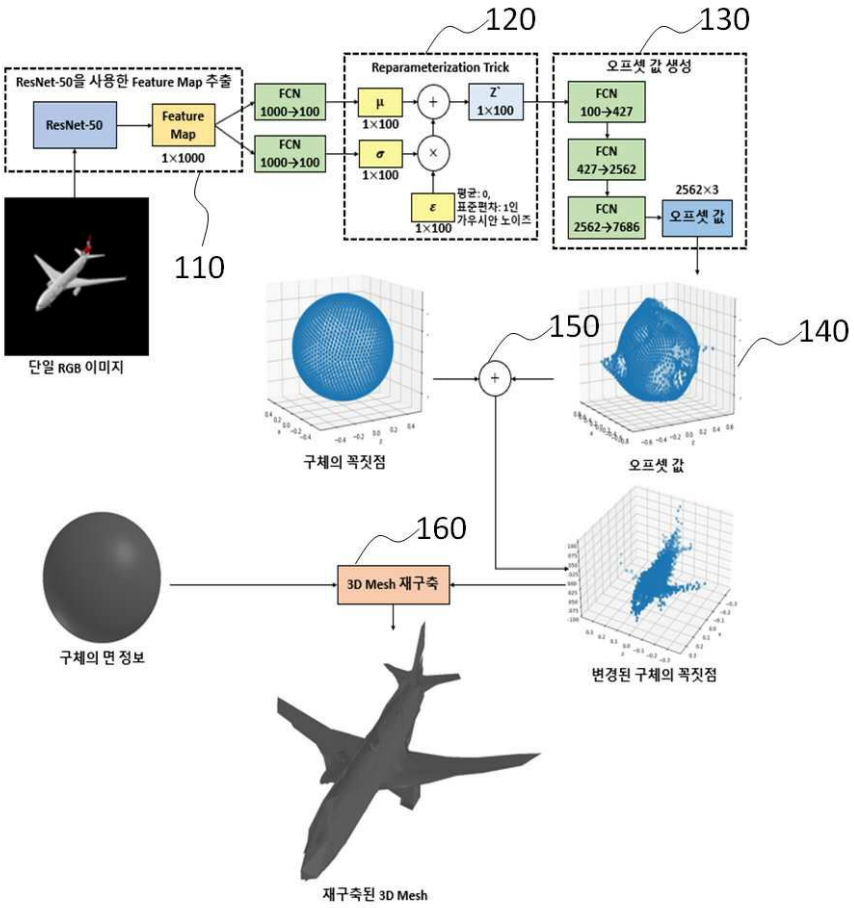
[0101] 도 23은 Chamfer Distance를 비교 평가한 결과를 그래프로 나타내고 있다. 본 발명에서 제안하는 기법의 Chamfer Distance 값이 다른 논문에서 발표한 기법들보다 근소하게 높은 수치를 나타내었다. Chamfer Distance 값은 수치가 0에 가까울수록 우수함을 나타내므로, 다른 논문에서 발표한 기법들보다 우수한 것이 확인되었다. 단, AtlasNet은 다른 논문들과 다르게 RGB 이미지뿐만 아니라 3D 포인트 클라우드를 입력 데이터로 사용하여 학습하였기 때문에 Chamfer Distance 값이 가장 우수하다.

[0102] 변경된 구체의 꼭짓점의 Chamfer Distance Loss를 학습에 이용하지 않고, 재구축된 3D Mesh와 학습 데이터셋의 3D Mesh를 샘플링한 3D 포인트 클라우드 간의 Chamfer Distance Loss를 학습에 이용하는 것은 재구축된 3D Mesh가 변경된 구체의 꼭짓점의 간격과 밀도에 상관없이 사물의 윤곽을 잘 표현하도록 한다. 도 24는 변경된 구체의 꼭짓점과 재구축된 3D Mesh를 샘플링한 3D 포인트 클라우드를 나타낸다. 똑같은 3D Mesh에 대한 꼭짓점과 윤곽이다. 변경된 구체의 꼭짓점의 밀도가 균일하지 않고 점 간격이 일정하지 않아도 사물의 윤곽이 잘 표현되는 장점이 있다.

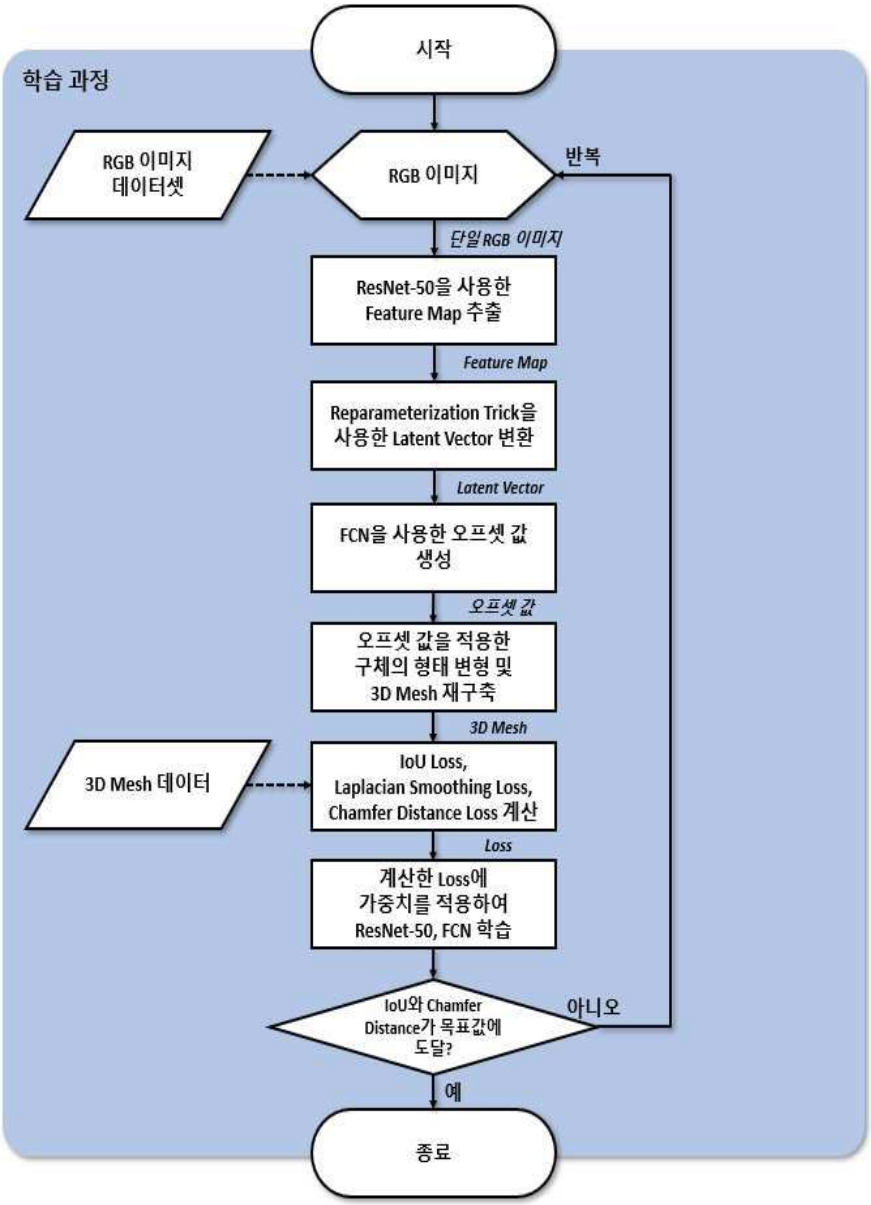
[0103] 이상에서 설명한 본 발명은 전술한 실시예 및 첨부된 도면에 의해 한정되는 것이 아니고, 본 발명의 기술적 사상을 벗어나지 않는 범위 내에서 여러 가지 치환, 변형 및 변경이 가능함은 당업자에게 명백할 것이다.

도면

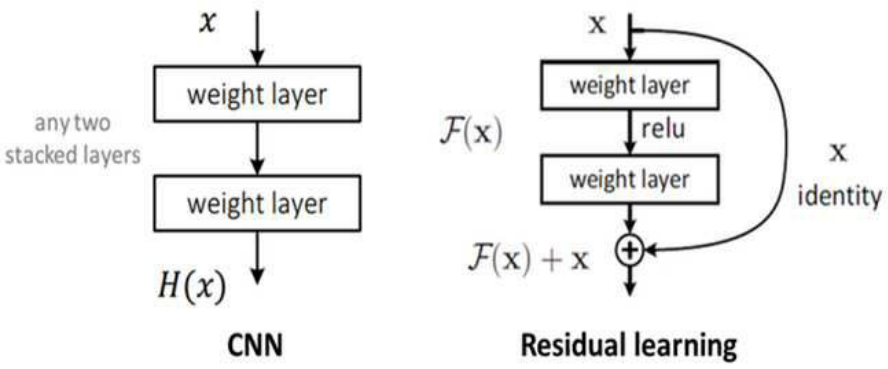
도면1


120
130
ResNet-50을 사용한 Feature Map 추출
ResNet-50
Feature Map
1×1000
FCN 1000→100
FCN 1000→100
Reparameterization Trick
μ
1×100
σ
1×100
ε
1×100
평균: 0, 표준편차: 1인 가우시안 노이즈
Z`
1×100
오프셋 값 생성
FCN 100→427
FCN 427→2562
FCN 2562→7686
2562×3
오프셋 값
110
단일 RGB 이미지
구체의 꼭짓점
150
140
오프셋 값
160
3D Mesh 재구축
구체의 면 정보
변경된 구체의 꼭짓점
재구축된 3D Mesh

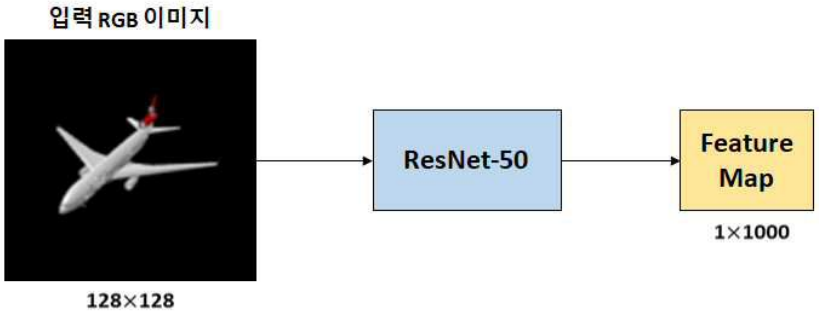
도면2


시작
학습 과정
RGB 이미지 데이터셋
RGB 이미지
반복
단일 RGB 이미지
ResNet-50을 사용한 Feature Map 추출
Feature Map
Reparameterization Trick을 사용한 Latent Vector 변환
Latent Vector
FCN을 사용한 오프셋 값 생성
오프셋 값
오프셋 값을 적용한 구체의 형태 변형 및 3D Mesh 재구축
3D Mesh
3D Mesh 데이터
IoU Loss, Laplacian Smoothing Loss, Chamfer Distance Loss 계산
Loss
계산한 Loss에 가중치를 적용하여 ResNet-50, FCN 학습
IoU와 Chamfer Distance가 목표값에 도달?
아니오
예
종료

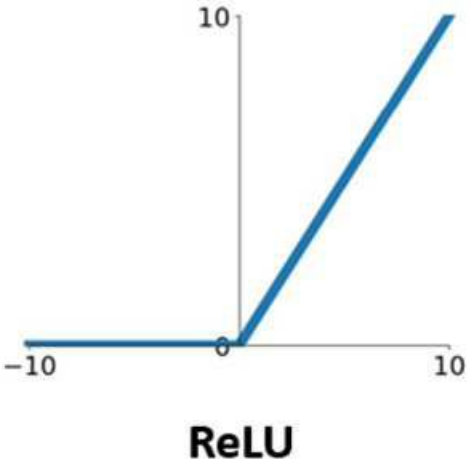
도면3


x
weight layer
any two stacked layers
weight layer
H(x)
CNN
x
weight layer
relu
weight layer
F(x)
x
identity
F(x) + x
Residual learning

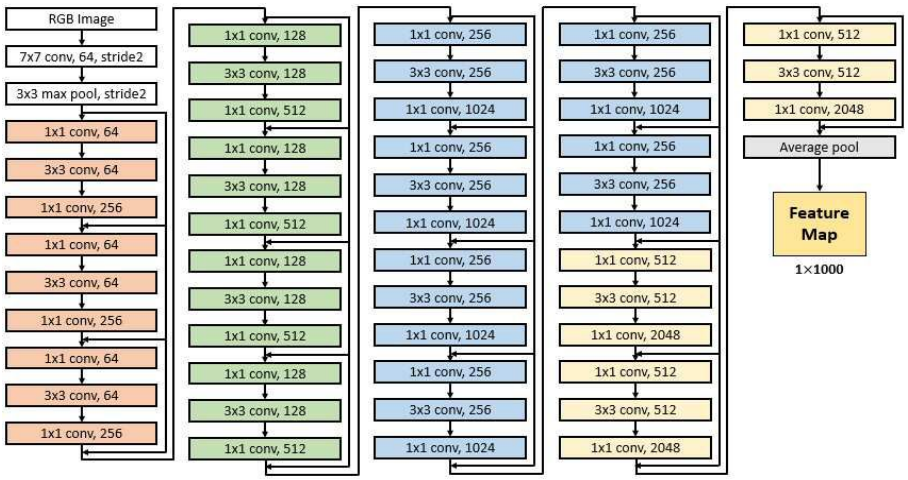
도면4


입력 RGB 이미지
128×128
ResNet-50
Feature Map
1×1000

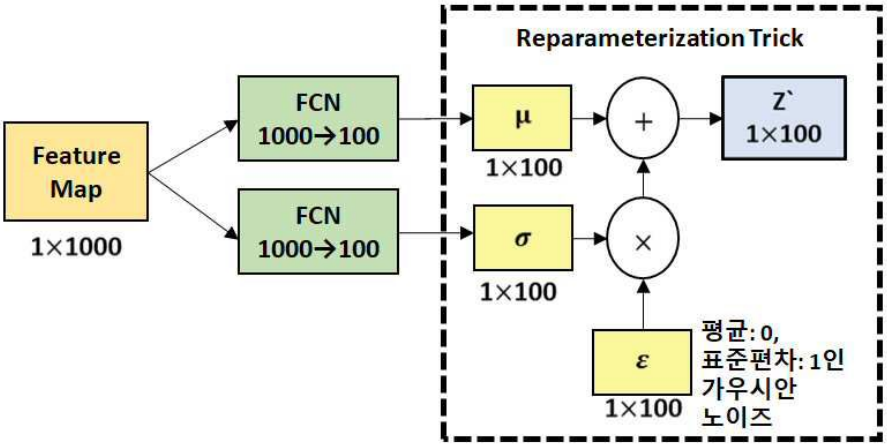
도면5


10
−10
0
10
ReLU

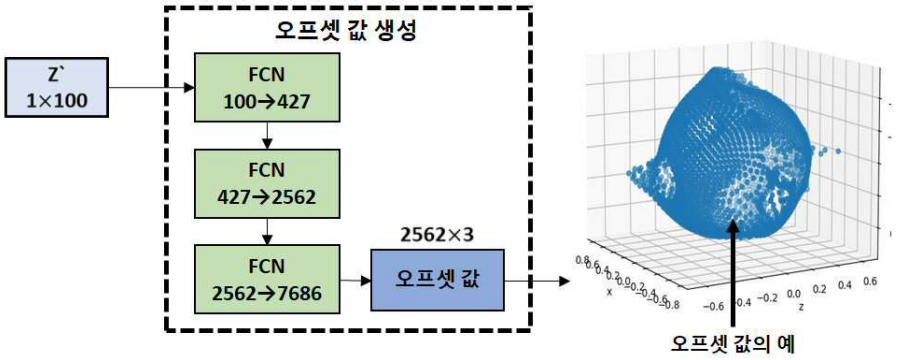
도면6


RGB Image
7x7 conv, 64, stride2
3x3 max pool, stride2
1x1 conv, 64
3x3 conv, 64
1x1 conv, 256
1x1 conv, 64
3x3 conv, 64
1x1 conv, 256
1x1 conv, 64
3x3 conv, 64
1x1 conv, 256
1x1 conv, 128
3x3 conv, 128
1x1 conv, 512
1x1 conv, 128
3x3 conv, 128
1x1 conv, 512
1x1 conv, 128
3x3 conv, 128
1x1 conv, 512
1x1 conv, 128
3x3 conv, 128
1x1 conv, 512
1x1 conv, 256
3x3 conv, 256
1x1 conv, 1024
1x1 conv, 256
3x3 conv, 256
1x1 conv, 1024
1x1 conv, 256
3x3 conv, 256
1x1 conv, 1024
1x1 conv, 256
3x3 conv, 256
1x1 conv, 1024
1x1 conv, 256
3x3 conv, 256
1x1 conv, 1024
1x1 conv, 512
3x3 conv, 512
1x1 conv, 2048
1x1 conv, 512
3x3 conv, 512
1x1 conv, 2048
1x1 conv, 512
3x3 conv, 512
1x1 conv, 2048
Average pool
Feature Map
1×1000

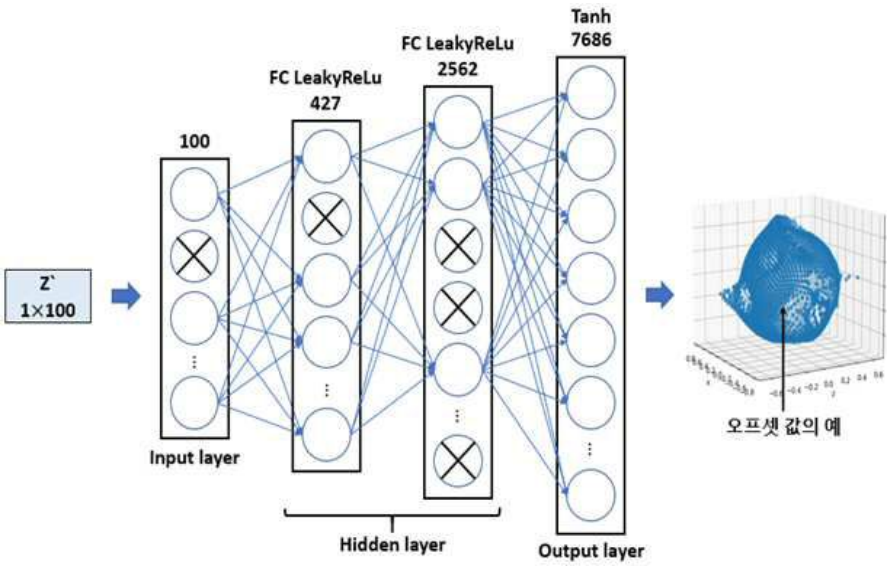
도면7


Reparameterization Trick
Feature Map
1×1000
FCN 1000→100
FCN 1000→100
μ
1×100
σ
1×100
+
×
Z`
1×100
ε
1×100
평균: 0, 표준편차: 1인 가우시안 노이즈

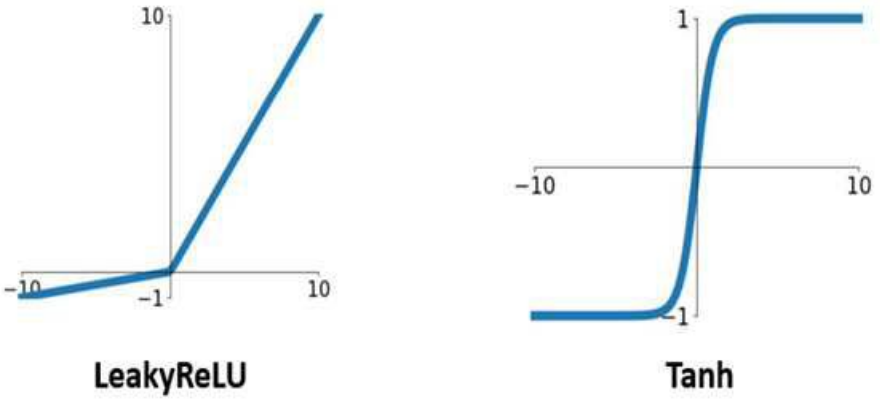
도면8


오프셋 값 생성
Z`
1×100
FCN 100→427
FCN 427→2562
FCN 2562→7686
2562×3
오프셋 값
오프셋 값의 예

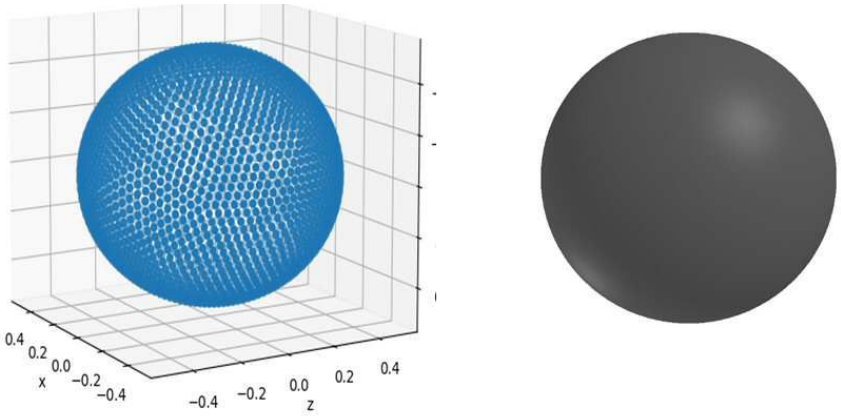
도면9


Z`
1×100
100
Input layer
FC LeakyReLu 427
FC LeakyReLu 2562
Hidden layer
Tanh 7686
Output layer
오프셋 값의 예

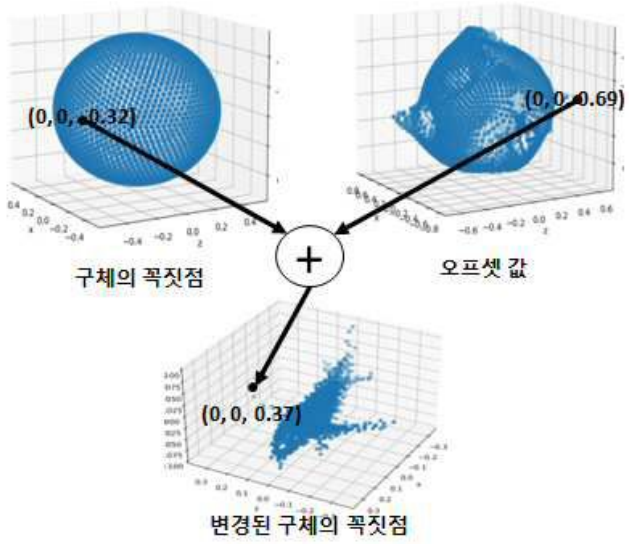
도면10



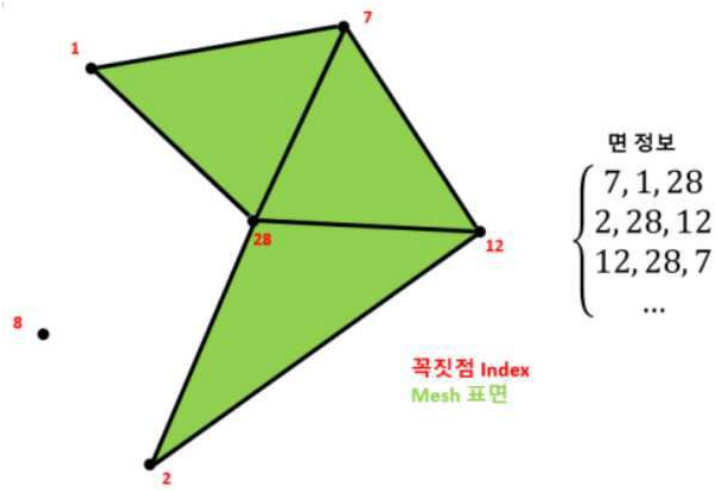
도면11



도면12



도면13

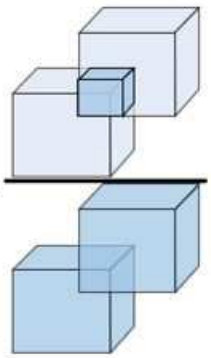


도면14

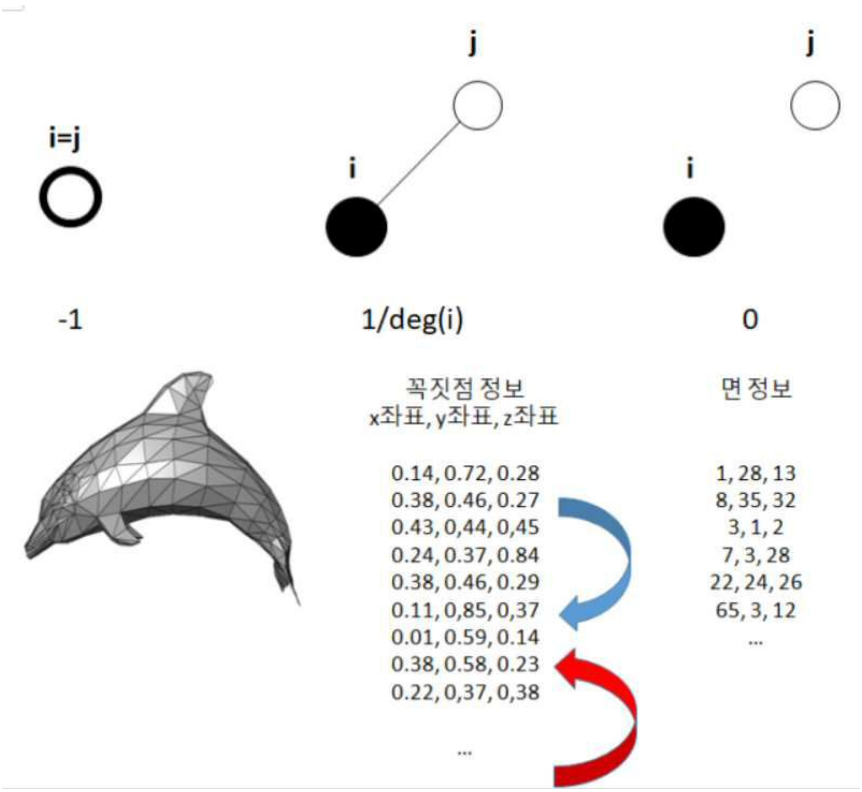


도면15

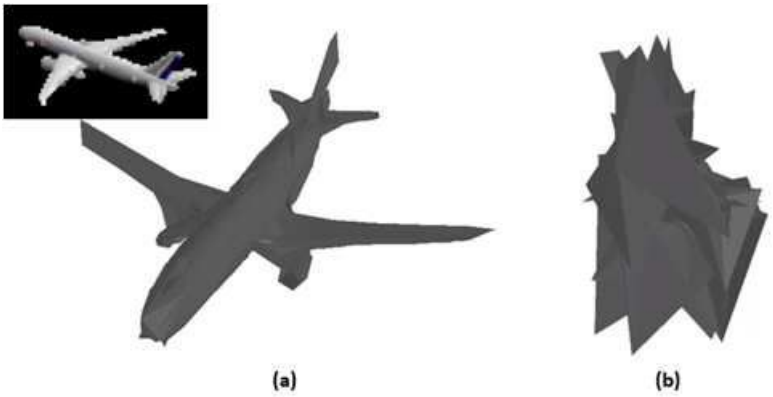
$$\mathrm{IoU}_{Loss} = \frac{Intersection}{Union} =$$



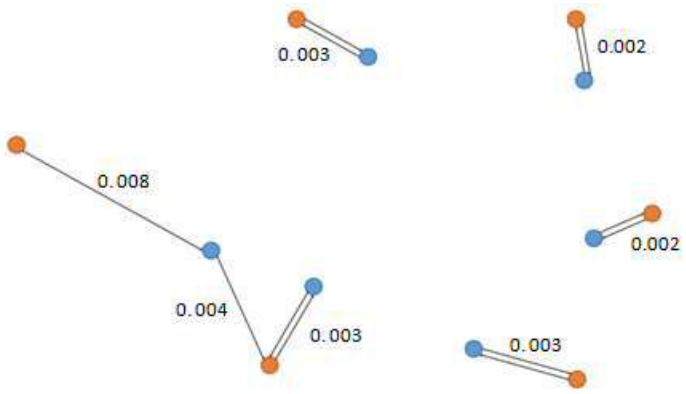
도면16



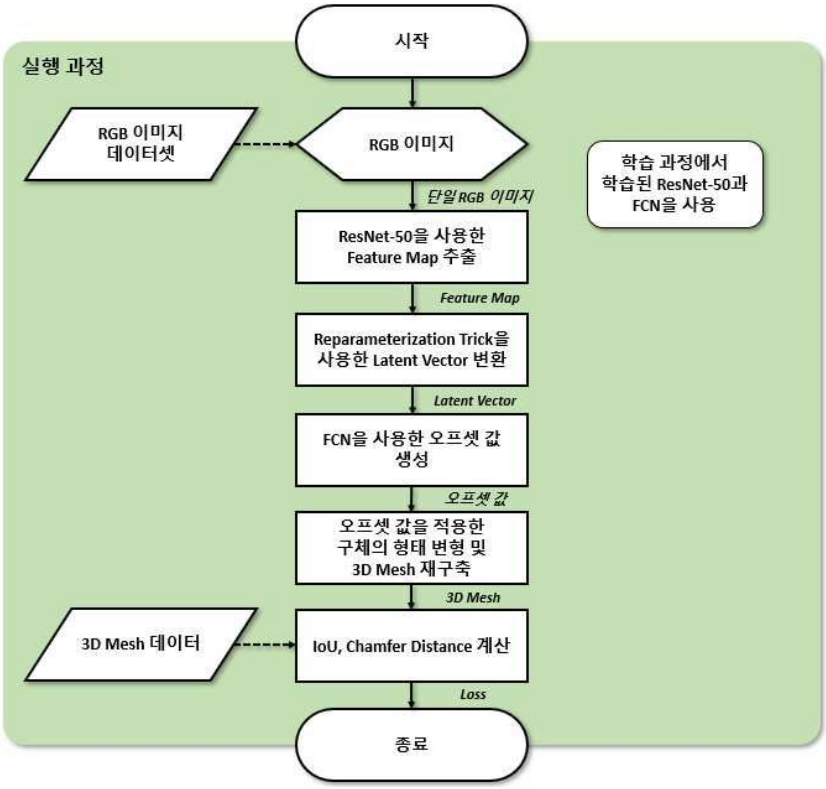
도면17



도면18



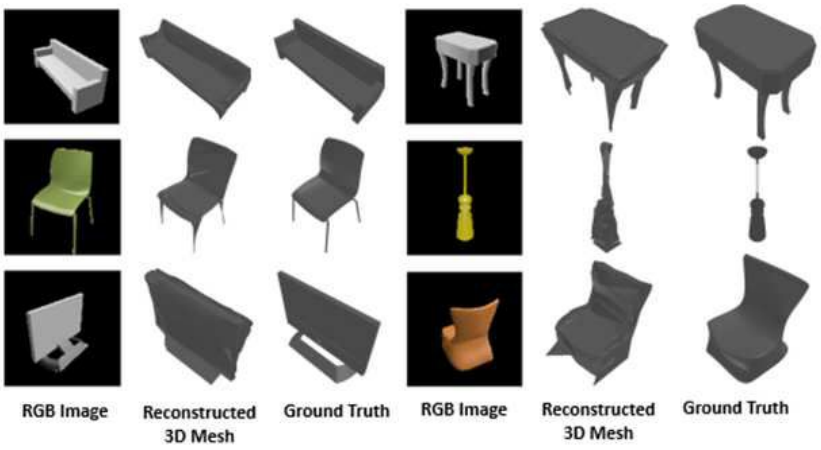
도면19


시작
실행 과정
RGB 이미지 데이터셋
RGB 이미지
학습 과정에서 학습된 ResNet-50과 FCN을 사용
단일 RGB 이미지
ResNet-50을 사용한 Feature Map 추출
Feature Map
Reparameterization Trick을 사용한 Latent Vector 변환
Latent Vector
FCN을 사용한 오프셋 값 생성
오프셋 값
오프셋 값을 적용한 구체의 형태 변형 및 3D Mesh 재구축
3D Mesh
3D Mesh 데이터
IoU, Chamfer Distance 계산
Loss
종료

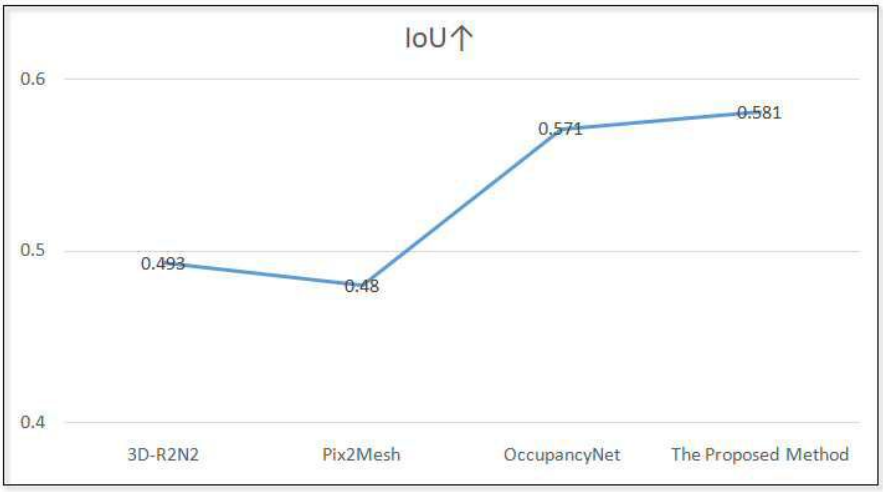
도면20



도면21



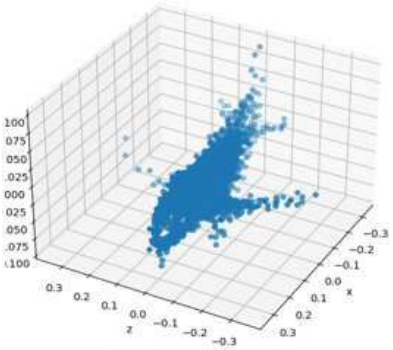
도면22



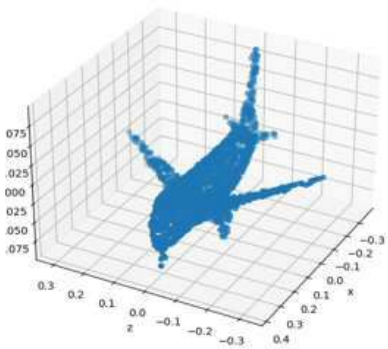
도면23



도면24



변경된 구체의 꼭짓점



재구축된 3D Mesh를 샘플링한
3D 포인트 클라우드