석사학위청구논문
2022 학년도

동적 환경에서 3D LiDAR SLAM을 위한 동적 물체 제거 및 포인트 클라우드 인페인팅

Moving Object Removal and Point Cloud Inpainting for 3D LiDAR SLAM in Dynamic Environments

광운대학교 대학원

로봇학과

한 창 완

동적 환경에서 3D LiDAR SLAM을 위한 동적 물체 제거 및 포인트 클라우드 인페인팅

Moving Object Removal and Point Cloud Inpainting for 3D LiDAR SLAM in Dynamic Environments

지도교수 오 정 현

이 논문을 공학 석사학위 청구논문으로 제출함.

2022년 12월 일

광운대학교 대학원

로봇학과

한 창 완

한창완의 공학 석사학위논문을 인준함.

심사위원장 _____ 인

심사위원 _____ 인

심사위원 _____ 인

광운대학교 대학원

2022 년 12 월 일

감사의 글

학부 과정을 공부하면서 더 많은 분야를 알게 되고 공부가 부족하다는 것을 느껴서 대학원을 진학하게 되었습니다. 많은 배려와 도움으로 무사히 2년 동안의 석사 과정을 끝내고 졸업을 하게 되었습니다. 프로젝트를 진행하며 협업에 대해 배울 수 있었고 연구를 진행하면서 많은 지식을 쌓을 수 있었습니다. 연구실 석사과정의 첫 기수여서 걱정을 많이 했지만 오정현 지도 교수님의 배려로 편하게 진행할 수 있었습니다. 특히 연구 주제를 선택하는데 많은 도움과 조언을 주신 오정현 지도 교수님께 항상 감사 드립니다. 또한 공개 발표 및 초심에서 적절한 조언과 심사를 진행해주신 박광현 교수님, 정문호 교수님 덕에 연구에 많은 진행이 있었으며 감사드립니다. 석사 과정을 진행할 수 있게 학부 시절에 학문 수행에 도움을 주신 조 황 교수님, 정문호 교수님, 박일우 교수님, 양우성 교수님께 감사드립니다. 그리고 학부 생황에 프로젝트 및 현장 경험을 알려주신 현재는 퇴직하신 최 익 교수님, 김진오 교수님께도 감사드립니다. 연구를 진행하며 같이 의견을 내주고 도움을 준 동기 형준이도 정말 감사합니다. 연구실 생활에 배려와 도움을 준 석사 2학기, 1학기를 진행하고 있는 후배 유동길, 정지훈 덕분에 더 의미 있는 석사 생활을 진행할 수 있었습니다. 마지막으로 언제나 뒤에서 응원해주시는 사랑하는 어머니 남정복, 아버지 한송석, 동생 한창우에게 항상 은혜를 느끼고 있습니다. 모든 분들의 정성 어린 도움덕에 석사 과정을 마치고 졸업을 할 수 있게 되었습니다. 사회에 진출해서도 받았던 도움을 생각하며 성실하게 임하겠습니다. 감사합니다.

2022년 12월

한창완 올림

국문 요약

최근에 LiDAR 센서를 통해 현재의 위치를 추정할 수 있는 다양한 SLAM 알고리즘이 연구되고 있다. 그러나 연구가 정적 환경을 기반으로 제안되었기 때문에 움직이는 물체가 많은 환경에서 취약하다. 움직이는 물체를 감지하는 것은 포즈를 추정하고 향후 작업을 수행하는 데 중요하다. 이 문제를 해결하기 위한 방법으로 움직이는 물체를 모두 제거하는 방법이 있다. 하지만 이 방법은 정보의 손실로 인해 지도에 많은 구멍을 생기게 해 신뢰성을 떨어뜨린다. 이를 해결하기 위해서는 상세한 물체의 움직임에 대한 상세한 분류 기준과 포인트 클라우드의 복원 방법이 필요하다. 본 논문에서는 움직이는 물체 분할 네트워크, 인페인팅 네트워크, 와 새로운 손실함수를 제안하였다. 우리는 정보 손실을 최소화하기 위해 이동 가능한 물체와 움직이는 물체를 분류한다. 분할 네트워크의 입력에 레지듀얼 이미지(Residual image)를 추가하여 움직이는 물체에 대한 정보를 학습하고 분류한다. 또한, 인페인팅 결과가 포인트 클라우드 간의 특성을 유지할 수 있도록 스무스니스 손실함수(Smoothness Loss)를 제안한다. 결과를 확인하기 위해 CARLA를 통해 데이터 세트를 직접 생성하고 제안된 방법의 타당성을 입증했다.

Abstract

There are various SLAM algorithms capable of estimating the current pose through a LiDAR. However, it is vulnerable to moving objects because it was proposed based on static environments. Detecting moving objects is important for estimating the pose and performing future tasks. Remove all moving objects as a way to solve the problem. However, this method creates many holes in the map and reduces reliability. To solve the problem, a detailed classification criteria and a method of restoring point cloud are needed. In this paper, we proposed the moving object segmentation network and inpainting network. To minimize lost information, we classify movable objects and moving objects. By adding a residual image to the input of the segmentation network, information on moving objects can be learned and classified. Also, smoothness loss is proposed so that the inpainting result can maintain the characteristics between point clouds. To check the results, a dataset was directly created through CARLA and the validity of the proposed method was proved.

차 례

그림 차례

표 차례

제 1 장 서론

LiDAR 를 이용해 simultaneous localization and mapping(SLAM)을 수행하는 방법은 다양하게 연구되어 왔다. 3D LiDAR SLAM은 각각의 포인트 클라우드에서 특징 포인트를 획득한 후, 매칭을 통해 지도를 생성 [1], [2]하며 최근에는 딥러닝(Deep Learning)을 도입해 특징 포인트를 생성하는 방법 [3]이 제안되었다. 제안된 방법은 대부분이 정적인 환경을 가정으로 연구되었으며 동적 환경에서 움직이는 물체 간의 특징 포인트 매칭으로 인해 위치 추정의 오차가 생성된다. 위 문제를 해결하기 위해서는 움직이는 물체를 구분하고 해당 포인트를 제거하는 방법 [4], [5]이 필요하다. 움직이는 물체를 구분 후 제거하기 위해서 레이더, 카메라 [6]를 이용하거나 딥러닝 [7]을 이용하는 방법이 제안되었다. 해당 방법을 통해 동적 물체가 많은 환경에서 위치 추정 오차가 줄어 성능의 향상을 보이지만 움직이는 물체가 많아서 제거된 포인트가 많아지는 경우, 많은 구멍들이 생성되며 특징 포인트 매칭 과정에서 오차가 생성된다.

구멍으로 인해 생성된 오차를 제거하기 위해서 학습 을 통한 빈 공간을 채우는 inpainting 네트워크인 SAM-Net [8]이 제안되었다. 움직이는 물체를 이진 마스크로 분류를 진행하고 해당 마스크를 통해서 지워진 부분을 inpainting 모듈을 통해서 복원하는 방법이다. 하지만

segmentation 모듈을 통해서 움직일 수 있는 가능성이 있는 물체들을 모두 제거해 특징 포인트 매칭에 유효할 수 있는 정보를 모두 제거한다. 또한 generator 학습에 포인트 클라우드의 특징을 반영하지 않아 복원된 정보가 일그러지는 문제가 있다.

우리는 그림 1-1 의 목적을 이루기 위해 기존 방법과 달리 최대한 많은 특징 포인트를 이용하기 위해서 움직이는 물체만을 분류하는 network 로 사용하고, 포인트 클라우드의 특징이 반영된 손실함수를 추가하는 inpainting 학습 방법을 제안한다. 제안 방법은 연속된 스캔의 포인트 클라우드를 이용해 residual image[7]를 생성한다. Residual image를 입력으로 사용해, 제안 모델은 스캔 간의 물체의 움직임 정보를 학습해 움직이는 물체를 분류할 수 있다. 또한 우리는 유효한 특징 포인트를 생성하기 위해서 inpainting 네트워크의 결과로 기존의 가려졌던 배경이 채워질 때, 채워진 배경과 실제 배경의 feature points 의 일관성을 유지하는 것에 집중했다. 그래서 우리는 일관성을 유지하기 위해서 포인트 클라우드의 특징 중 하나인 smoothness[1], [2]를 이용하는 손실함수를 추가했다. 제안한 손실함수는 여러 포인트 간의 연관성을 이용한 특징을 비교하기 때문에 모델이 주변 환경과 유사한 결과를 생성할 수 있게 한다. 실험 간에 smoothness 손실함수를 통해 기존 방법보다 ground truth 와 유사한 inpainting 결과와 특징점을

생성했다.

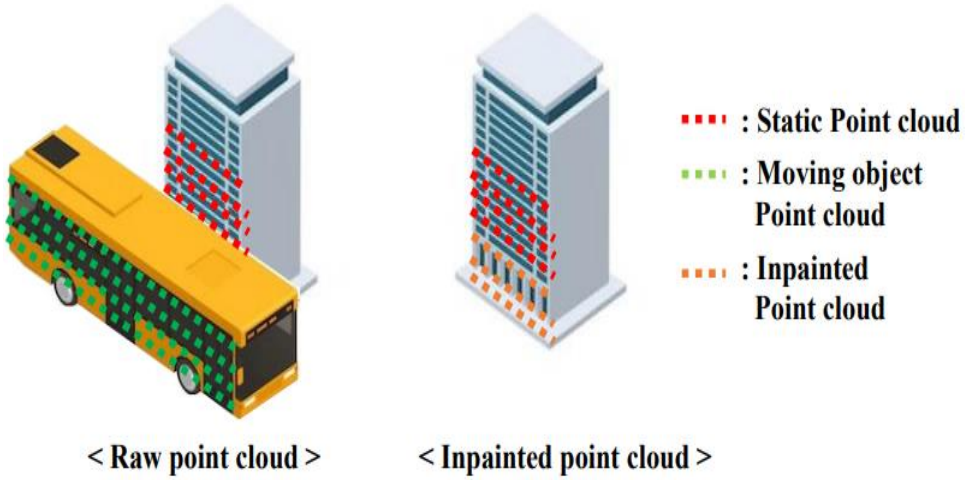


그림 1-1 제안 모델의 목적

Figure 1-1 Purpose of Model

기존의 데이터 셋(Dataset)에서는 움직이는 물체만을 표현하는 label 과 정적인 배경의 ground truth 가 존재하지 않았기 때문에 제안 방법의 학습을 위해서 우리는 CARLA[9] 시뮬레이션을 통해서 새로운 데이터 셋을 생성했다. CARLA 의 센서를 통해 움직일 수 있는 물체를 분류했고 속도정보를 기반으로 움직이는 물체의 label 을 생성했다. 이를 기반으로 학습을 진행했으며 아래와 같은 contribution 을 가진다.

• Residual images 를 이용하는 것으로 움직이는 물체의 label 을 생성하는 inpainting 모델을 제안한다. 기존 방법이 움직일 수 있는

물체만을 구별해 포인트의 손실이 생긴 것을 줄이는 것으로 위치 추정의 오차를 줄일 수 있다.

• 기존 학습 모델과 다르게 포인트 간의 연관성을 통해 획득한 planar, edge 특징 정보를 이용하는 smoothness loss 를 통해 보다 자연스러운 inpainting 결과를 얻을 수 있다.

• Inpainting 모델을 학습하기 위해 기존의 인위적인 데이터 생성 방법을 극복하기위해서 CARLA 시뮬레이션을 통해 현실과 유사한 dataset 을 생성한다. 이 데이터를 통해 모델은 움직일 수 있는 물체가 아닌 움직이는 물체를 분류할 뿐만 아니라 다양한 상황의 데이터를 학습해 분류의 성능을 높일 수 있다.

본 논문의 진행은 다음과 같다. 2 장에서 LiDAR 센서를 이용해 얻을 수 있는 다양한 특징점, SLAM 에 대해 설명한다. 동적 환경에서 SLAM 의 성능의 저하가 생긴 이유와 해결하기 위한 방법에 대해 설명한다. 그 중 기존 Inpainting 모델에 대해 설명하며 이를 개선하기 위한 방법을 제안한다. 3 장은 기존 학습 방법을 개선하기 위한 데이터 생성 방법, input 데이터 변경, 새로운 손실함수를 설명한다. 다음으로 4 장에서는 제안 방법과 기존 방법의 segmentation, inpainting 성능을 생성한 CARLA dataset 을 통해 비교한다. 다음으로 기존 SLAM 알고리즘과 제안 모델을 이용한 방법을 비교하는 것으로 제안 모델의

유용함을 증명한다. 이후, KITTI[10] benchmark dataset 을 이용해 결과를 비교하며 시뮬레이션 데이터 뿐만 아닌 실제 환경에서의 적합성을 확인한다. 마지막으로 5 장은 결론과 향후 계획을 설명한다.

제 2 장 배경 지식 및 관련 연구

제 2.1 절 라이더 특징점 추출(LiDAR Feature Extraction)

포인트 클라우드의 모든 정보를 사용해 필요한 결과를 생성하기 위해서는 많은 시간이 필요하기 때문에 특징을 잘 표현하는 특징 포인트를 선택해서 사용하는 방법이 연구되었다. 해당 방법에는 포인트 사이의 변화가 뚜렷한 환경에서 안정적으로 법선 벡터(normal vector)를 추출할 수 있는 NARF(Normal Aligned Radial Feature)[11]이 있고 다중 히스토그램을 이용해 연산 속도를 높인 FPFH(Fast point feature histogram)[12]이 있다. 하지만 위 방법은 RGB-D와 같은 밀도가 높은 포인트 클라우드를 제공하는 환경에서만 사용이 가능하며 외부 환경에서 이용하는 LiDAR의 포인트 클라우드의 경우 적용하기 어렵다. 실시간으로 진행하는 SLAM에 사용하기 위해서 포인트 간의 정보를 통해 surface와 edge를 추출하는 방법[1]이 제안되었다. 본 논문은 해당 이 방법을 loss 함수로 구현하는 것으로 inpainting의 결과와 기존 배경의 특징이 연속성이 유지될 수 있도록 한다.

제 2.2 절 라이더 슬램(LiDAR SLAM)

정확한 지도를 생성하기 위한 이전 연구로는 filter [13], [14]를 적용하는 방법이 있었다. 해당 방법은 2D points에서만 적용이 가능하고 입체적인 지도를 획득할 수 없으며 많은 시간이 필요하다. 해당 문제를 해결하기 위해서 특정 feature를 이용하거나 의미론적 정보를 이용하는 방법이 제안되었다. 그 중 LOAM[1]의 경우, 포인트의 smoothness 특징을 이용해 surface와 edge를 구분해 실시간으로 특징 매칭이 가능하다. 이후, 지면 포인트의 정보를 제거해 불필요한 정보를 제거하는 것과 Levenberg-Marquardt 방법을 이용한 최적화를 통해 성능을 향상 시킨 방법[2]이 제안되었다. 하지만 정적인 환경에서 연구를 진행했기 때문에 도시와 같은 움직이는 물체가 많은 복잡한 환경에서는 SLAM의 성능의 저하를 보인다.

위 문제를 해결하기 위해 움직일 수 있는 물체를 제거하는 방법[4], [15], [16]이 제안되었다. 딥러닝을 기반으로 물체의 label을 설정하고 연속된 스캔에서 물체에 포함된 surfel 정보를 기반으로 지도를 생성하는 방법 [17]이 제안되었다. 해당 방법의 경우, 생성된 지도에서 물체의 surfel의 변화를 인지하는 것으로 움직이는 물체를 지도에서 제거할 수 있다.

제 2.3 절 인페인팅 방법(Inpainting Method)

최근 inpainting 연구는 딥러닝을 기반 진행되며 특히 GAN을 통해 입력으로 주어진 정보를 기반으로 이미지의 구멍을 채우는 방법이 제안[18], [19]되었다. inpainting을 SLAM에 적용하기 위해서 외부 환경을 inpainting하는 방법 [20], [21]이 제안되었다. ORB를 기반의 손실함수를 생성해 inpainting 결과에서 특징점의 왜곡을 최소화하는 방법으로 진행되었고 SLAM에서 장소인식과 visual odometry에 성능의 향상을 가져오는 것을 확인할 수 있다.

LiDAR의 경우, 포인트 클라우드의 밀도를 높이기 위해서 inpainting 방법 [22]–[24]을 이용한다. RGB 정보와 딥러닝 모델을 이용해 부족한 정보를 보안해 포인트 클라우드의 밀도를 높이며 완성도 있는 지도를 만들 수 있다. 다른 inpainting 방법으로 물체에 가려진 포인트를 복원하는 방법[8]이 제안되었다. 학습 모델을 통해 움직일 수 있는 물체를 분류하고 해당 위치의 포인트를 정적인 포인트를 복원하기 위해 구조적 정보를 이용했다. 하지만 인위적인 데이터셋 생성, 많은 포인트의 유실, 복원 시 포인트 클라우드의 특징을 반영하지 않은 한계가 있다.

제 3 장 제안하는 방법

제 3.1 절 데이터 전처리 : 레지듀얼 이미지 (Data preprocessing : Residual image)

학습의 입력으로 포인트 클라우드를 이용하기 위해서 3차원 데이터를 2차원 데이터인 range image로 변환한다. 데이터를 2D으로 변환하는 것으로 3D 포인트 클라우드를 이용으로 인해 생기는 정보 해석의 불편함과 소요되는 시간을 단축할 수 있다. 또한 이미지를 처리를 통해 연구된 다양한 2D convolutional neural 네트워크를 이용해 유용한 결과를 획득할 수 있다. LiDAR 포인트 클라우드를 이용한 변환, 학습을 진행하는 연구에서 range image는 자주 사용되는 방법 [7], [25]이며 본 논문에서는 간단하게 설명할 것이다. Cartesian 좌표계 상의 하나의 LiDAR points를 p = {x, y, z}, range image의 구형 좌표를 (u, v)로 정의할 때, 구형 좌표 변환 $\Pi: R^3 \rightarrow R^2$ 을 통해 Cartesian에서 구형 좌표계로 변환하며 아래의 식과 같다.

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}[1 - arctan(x,y) \cdot \pi^{-1}] \cdot w \\ [1 - (arcsin(z \cdot r^{-1}) + f_{up}) \cdot f^{-1}] \cdot h \end{pmatrix} \qquad (3-1)$$

h, w는 생성할 range image의 수직과 수평의 크기를 뜻하며 $f = f_{up} + f_{down}$ 으로 LiDAR 센서가 측정할 수 있는 Field of view의 수직적 거리를 뜻한다. $r = \sqrt{x^2 + y^2 + z^2}$은 LiDAR 포인트의 각각의 거리를 뜻한다. 3차원의 정보를 (u, v) 인덱스에 r의 2차원 거리 정보를 매칭하여 이미지로 변경이 가능하다.

Residual image는 네트워크에 물체의 움직임 정보를 학습하기 위해서 두 개의 포인트 클라우드 간의 거리 차이 정보를 2차원으로 표현한 것이다. 그림 3-1과 같이 Residual image를 생성하기 위해서 먼저 현재 frame i의 좌표계에 이전 frame j의 포인트 클라우드 P_j를 좌표 변환 T_j^i 을 진행해야 한다. 좌표 변환된 포인트 클라우드를 $P_{j\to i}$로 정의하고 구형 좌표 변환을 통해서 각각의 인덱스에 맞는 거리 정보를 이용해 range image D_i, D_j^i 를 생성한다. 마지막으로 현재의 frame과 변환된 이전 frame의 각각의 range image 인덱스의 거리 차이를 이용해 residual image를 만들 수 있다. 현재 frame i의 한 개의 LiDAR 포인트인 p_k에서의 residual 정보를 $d_{j\to i,k}$ 정의하고 아래와 같은 방법으로 구할 수 있다.

$$d_{j\to i,k} = \begin{cases} \dfrac{\left|r_k - r_k^{j\to i}\right|}{r_k} & k \in vaild\ points \\ 0 & otherwise \end{cases} \qquad (3-2)$$

r_k는 현재 frame i에서 p_k의 (u_k, v_k)의 픽셀에 해당하는 값이며 $d_{j\to i,k}$는

두 range image의 정규화 된 거리 차이의 절대값이며 p_k 와 동일한 index를 가진다. 본 논문은 현재 frame i를 기준으로 이전 frame i−3까지 차이를 이용해 residual image를 생성하며 range image, x, y, z 값과 함께 7 channel의 input data를 생성한다.

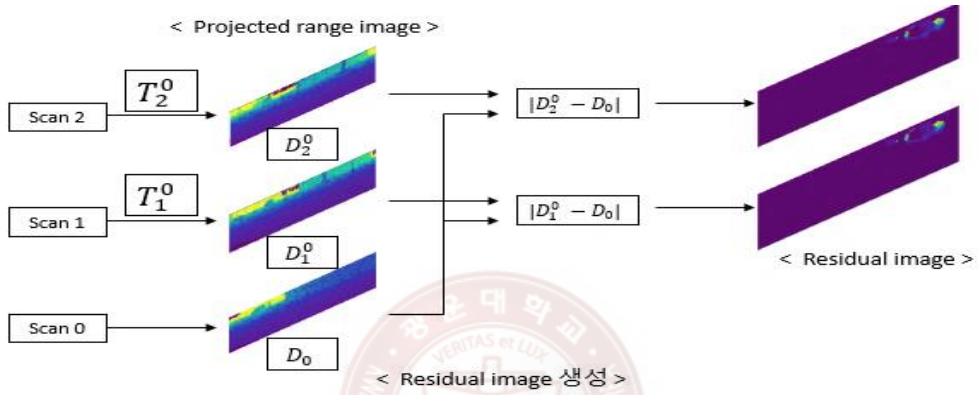


그림 3−1 레지듀얼 이미지 생성

Figure 3-1 Generate Residual Image

제 3.2 절 학습 모델 (Learning Model)

제안 방법은 기존에 사용된 네트워크[8]를 기반으로 움직이는 물체를 구분 및 제거하며 inpainting을 통해서 배경 정보를 복원한다. 새로 포인트를 복원하기 위해서 1 개의 Encoder와 2 개의 Decoder 모듈을 이용해 학습한다. 그림 3-2를 통해 제안 방법의 framework와 출력 정보를 확인할 수 있다. 먼저 Encoder 모듈을 통해서 input을 압축하게 되며 residual image 정보를 통해 움직이는 물체에 대한 특징 포함한 feature map을 생성한다. 다음으로 Encoder 모듈을 통해서 생성된 feature map을 이용해 움직이는 물체를 분류하는 이진 마스크를 생성한다. 이후, 이진 마스크와 feature map을 입력으로 inpainting 모듈을 통해 움직이는 물체로 인해 가려진 부분을 복원한다.

기존 네트워크의 경우, 단일 range image를 이용해 segmentation 모듈이 차량과 보행자의 움직임 상태와 관계없이 움직일 수 있는 물체를 분류하는 이진 마스크를 생성하며 이미지의 기울기 정보를 이용한 손실함수를 통해서 정적인 배경을 복원한다. 움직일 수 있는 물체를 모두 제거하는 경우, 주차된 차량과 같이 정적인 배경 정보 보다 위치 추정에 유효한 특징점을 추출할 수 있는 모든 부분을 제거하게 되는 문제가 있다. 또한 gradient 손실함수만을 이용했기 때문에 image의 특징점을 반영할

수 있으나 range image의 경우, 3차원의 포인트 클라우드 정보를 2차원으로 압축한 거리 정보임으로 기존 visual feature의 복원 방법으로는 일관성을 유지할 수 없다. 제안 방법의 경우, 유효한 특징점을 최대한 유지하며 필요한 부분만을 복원하기 위해서 Encoder 모듈에 residual image를 입력으로 사용해 segmentation 모듈의 결과로 움직이는 물체만을 분류하는 이진 마스크 $\tilde{m}$를 생성한다. 해당 이미지는 2개의 frame 간의 움직임 정보를 포함하고 있기 때문에 학습을 통해 움직이는 물체를 분류할 수 있게 된다. 이를 통해 주차된 차량, 서있는 보행자 등에서 유효한 특징을 유지해 위치 추정에 생기는 오차를 줄인다. 마지막으로 포인트 클라우드의 특징을 반영하기 위해서 여러 포인트 간의 거리의 연관성을 이용하는 특징[1], [2]을 반영한 손실함수를 이용해 기존의 배경의 부분과 새로 복원된 포인트를 자연스럽게 연결한 inpainted 이미지 $\tilde{I}$를 생성한다. 해당 방법에 관한 손실함수는 다음 절에서 서술한다.

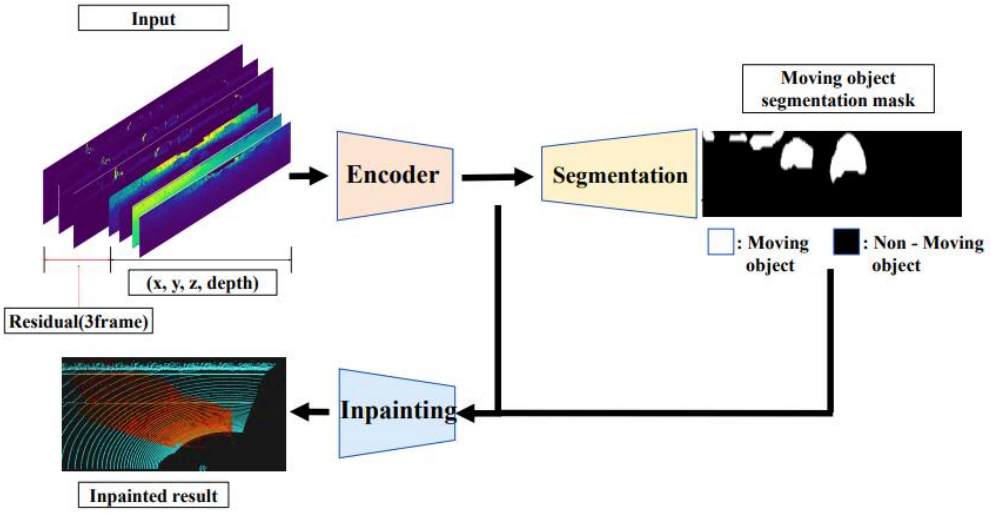


그림 3-2 제안 모델의 프레임워크

Figure 3-2 Framework of Proposed Model

제 3.3 절 스무스니스 손실함수(Smoothness Loss)

Smoothness feature points는 3D LiDAR SLAM에서 해당 포인트를 edge, planar를 판단해 위치 추정을 위해 사용되는 특징점이다. 이 특징점은 LOAM, LeGO-LOAM과 그 외에의 많은 연구에서 설명되며 효용성을 입증했다. 이번 절은 smoothness가 구해지는 방법과 해당 특징점을 새로운 손실함수로 적용하는 것을 설명할 것이다.

LiDAR 포인트를 range image로 변환 후, 행방향으로 연속적인 N 개의 포인트들의 묶음을 S, S에 포함된 smoothness c를 확인하려 하는 query 포인트를 p_i로 정의하면 다음과 같은 식을 통해 구할 수 있다.

$$c = \frac{1}{|S| \cdot |r_i|} \cdot \left\| \sum_{j \in S,\, j \neq i} (r_j - r_i) \right\| \qquad (3-3)$$

c의 값이 특정 임계값 보다 큰 경우, edge 포인트로 분류되고 임계값 보다 작은 경우는 planar 포인트로 분류한다. 두 개의 frame에서 획득한 edge, planar 포인트 간의 매칭을 통해 LiDAR의 odometry를 추정할 수 있다.

그림 3-3과 같이 식 3-3을 손실함수로 표현하기 위해서 max pooling layer와 convolutional layer를 이용한다. LiDAR 포인트의 경우, 센서 데이터의 밀도가 낮아서 range image의 모든 픽셀에 거리 정보를 가지고 있지 않다. 값을 가지고 있지 않은 픽셀이 유효 정보를 가지고 있는 포인트들의 사이에 존재하고 있으며 이는 학습에 방해요소이다. 해당 픽셀을 배제하기 위해서 max pooling을 통해 가장 큰 거리 값을 추출해 range image 정보를 압축한다. 이 후, [21]과 같이 새로운 kernel을 생성해 모든 포인트에 대해 행 방향으로 smoothness 정보를 획득한다. 이 후, max pooling을 통해 압축된 range image를 각각 픽셀 마다 나누어 정규화를 진행한다. 마지막으로 각각 inpainting 네트워크를 통해 생성된 $\tilde{I}$ 와 ground truth image I에서 각각 생성된 smoothness를 L1 loss $\mathcal{L}_1$ 을 이용해 비교한다. $\mathcal{F}$ 를 layers를 통한 변환 과정으로 정의하면 smoothness loss $\mathcal{L}_s$는 아래와 같다.

$$\mathcal{L}_s = \mathcal{L}_1\left(\mathcal{F}(\tilde{\mathrm{I}}), \mathcal{F}(\mathrm{I})\right) \qquad (3-4)$$

Max pooling을 이용할 경우, 입력 데이터가 압축이되 데이터의 손실이 일어날 수 있다. 이를 막기 위해서 max pooling의 filter의 크기는 2×2를 사용했으며 축소되는 정보를 최소화하며 거리 정보가 없는 픽셀을 제거했다. 다음으로 convolutional layer의 kernel을 S의

크기와 같게 생성하며 $1 \times |S|$로 생성한다. 본 논문에서 S의 크기는 query points를 포함해 9이며 query points의 양 옆으로 각각 4개가 배치된다. 그림 3-3에서 kernel의 회색 부분은 -1/9, 검은색 부분은 8/9이다. 학습에 사용된 가중치와 smoothness 손실함수를 제외한 다른 손실함수의 정보는 다음 절에서 자세하게 다룰 예정이다.

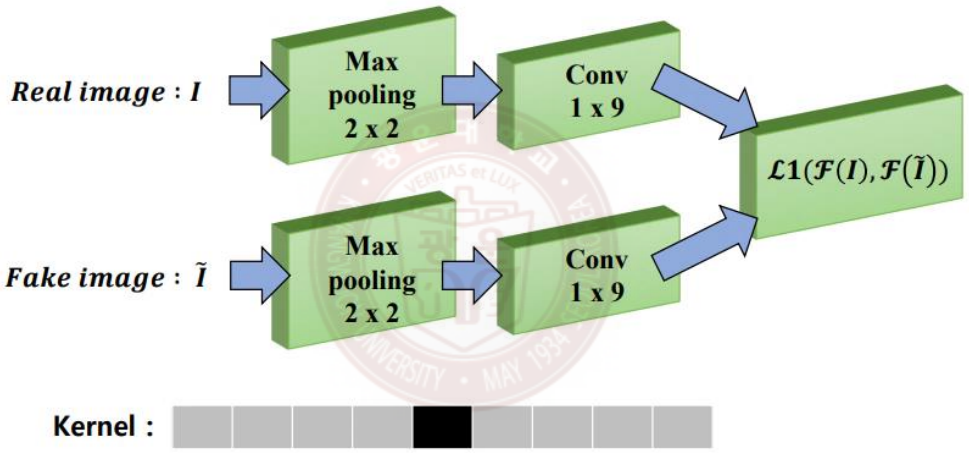


그림 3-3 스무스니스 손실함수

Figure 3-9 Smoothness Loss

제 3.4 절 모든 손실함수(Total Loss Function)

$\tilde{m}$ 를 이용해 지워진 움직이는 물체가 위치한 부분의 inpainting 을 진행하기 위해서 Generative Adversarial Networks(GAN)을 이용한다. Generator 와 Discriminator 의 경우, 기존의 SAM-Net 과 동일한 hinge loss 를 이용해 학습을 진행하며 각각 $\mathcal{L}_{\mathcal{G}}, \mathcal{L}_{\mathcal{D}}$로 표현한다. Generator 의 경우, segmentation 과 inpainting 을 동시에 진행하기 때문에 각각의 decoder 를 학습하기 위한 손실함수를 이용한다. Segmentation 에 사용되는 손실함수는 실제로 움직이는 물체를 포함한 ground truth 인 m 와 segmentation decoder 를 통해 생성된 $\tilde{m}$ 를 L1 loss 를 통해 차이를 구한다. 이를 Mask loss 하며 L_m으로 표현한다.

$$\mathcal{L}_m = \mathcal{L}_1(\tilde{\mathrm{m}}, \mathrm{m}) \tag{3-5}$$

이후 inpainting decoder 의 성능을 높이기 위해서 위에서 $\mathcal{L}_s$를 이용하며 추가적으로 거리 정보를 이용한다. 거리 관련 손실함수 $\mathcal{L}_d$ 는 생성된 거리 정보 $\tilde{d}$, ground truth I의 거리 정보를 d를 $\mathcal{L}_1$ 통해 계산한다.

$$\mathcal{L}_m = \mathcal{L}_1(\tilde{d}, d) \tag{3-6}$$

전체 손실함수의 합은 아래의 식과 같다.

$$\mathcal{L}_{total} = \mathcal{L}_{\mathcal{G}} + \lambda_{\mathrm{m}} \cdot \mathcal{L}_{m} + \lambda_{\mathrm{s}} \cdot \mathcal{L}_{s} + \mathcal{L}_{d} \qquad (3-7)$$

λ_m , λ_s 는 각각의 손실함수에 대한 가중치이다. 제안 모델은 초기 학습에 움직이는 물체에 대한 마스크 $\tilde{m}$ 의 학습이 중요하고 후반에는 $\tilde{m}$ 를 사용한 inpainting 이미지인 $\tilde{I}$ 에 집중이 필요하다. 이를 해결하기 위해서 1 epoch 마다 λ_m, λ_s에 0.998, 1.03 을 곱해 학습이 진행될수록 $\tilde{m}$ 에서 $\tilde{I}$ 로 중요도를 이동시켰다. 4 장의 실험 결과를 통해서 제안한 손실함수와 기존 방법의 결과를 비교할 것이다.

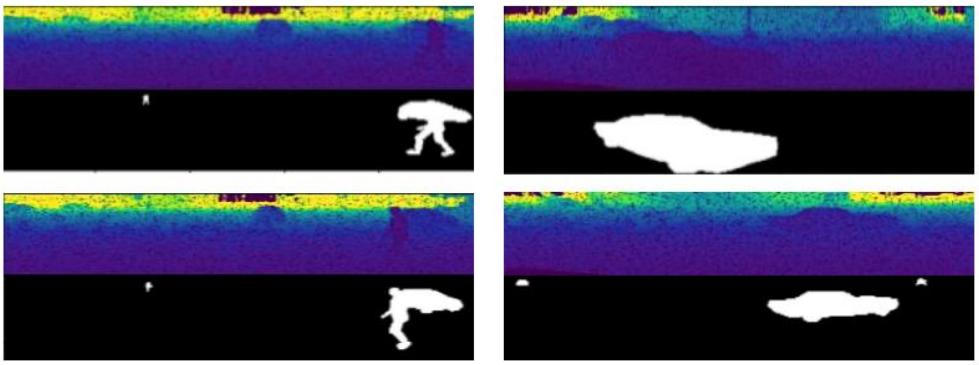
3.5 절 데이터 생성(Data Generation)

움직이는 물체를 특정하고 inpainting 을 진행하기 위해서 필요한 데이터는 움직이는 물체가 포함되지 않은 ground truth 와 동적 물체가 포함된 raw data 가 필요하다. 하지만 기존의 데이터 셋인 KITTI, Oxford Car[26]는 움직이는 물체의 의미론적 정보를 제공하지 않고 동적 물체로 가려진 부분의 ground truth 를 획득하기 어려워 inpainting 모델 학습에 사용하기 어렵다. Semantic KITTI[27]의 경우, 움직이는 물체에 대한 마스크 label 을 제공하지만 움직이는 물체로 가려진 부분의 ground truth 는 제공하지 않기 때문에 마찬가지로 적합하지 않다. 위 문제를 해결하기 위해서 기존 방법[8]은 KITTI 를 이용한 dataset 생성 방법을 제안했다. 하지만 빈 공간에 물체를 표현하는 포인트를 배치하는 인위적인 방법을 이용했기 때문에 실제 주행 환경과 유사한 데이터 셋은 획득하기 어렵다.

움직이는 물체의 label 과 가려진 ground truth 를 같이 획득하기 위해서 본 논문에서는 CARLA[9]를 이용해 데이터 셋을 제작했다. 먼저 동적 물체를 포함하는 환경에서 LiDAR 와 LiDAR semantic 을 통해 다양한 차량, 보행자와 같이 움직이는 물체를 포함하는 Raw dataset 을 생성한다. LiDAR 를 통해서 움직이는 차량에서 획득하는 포인트

클라우드를 획득할 수 있으며 LiDAR semantic 을 이용해 움직이는 물체를 분류하는 label 의 ground truth 를 획득할 수 있다. 이후, 동적 물체로 가려진 부분의 ground truth 를 획득하기 위해서 동일한 환경에서 동적 물체를 배제한 시뮬레이션을 통해 LiDAR 포인트 클라우드를 생성한다. 두 번의 시뮬레이션에서 LiDAR 를 부착한 차량의 위치를 frame 단위로 기록하며 위치를 비교해 동일한 위치의 포인트 클라우드를 비교하는 것으로 학습에 필요한 input 과 ground truth 를 생성할 수 있다.

총 2 개의 환경에서 주행을 진행하며 데이터셋을 생성했다. LiDAR 의 setting[28]은 KITTI 에서 사용한 HDL-64 와 동일하게 했으며 1 frame 당 약 15000 개의 포인트를 획득할 수 있었다. 기존의 방법과 달리, 그림 3-4 와 같이 움직이는 물체들이 겹치거나(그림 3-4(a)) 추월하는 상황(그림 3-4(b)) 등의 실제 상황에서 자연스러운 상황을 포함해 데이터를 생성할 수 있었으며 이를 통해 학습 모델이 동적 물체를 분류하는데 영향을 주었다.



(a) (b)

그림 3-4 다양한 상황을 포함한 데이터 셋

Figure 3-4 The Dataset with Various Situations

제 4 장 실험 및 결과

제 4.1 절 실험 준비 (Experiment Setting)

본 제안된 모델의 성능을 확인하기 위해서 segmentation, inpainting, 그리고 위치 추정의 성능을 확인한다. Segmentation 과 inpainting 성능은 SAM-Net과 비교하며 위치 추정의 성능은 LeGO-LOAM과 제안 모델을 더한 방법을 비교한다. 마지막으로 실제 환경에서 제안 모델의 성능을 평가하기 위해 KITTI의 Road data를 이용해 위치 추정의 성능을 확인한다.

데이터 셋은 CARLA를 이용해 생성했으며 Town_1, Town_4에서 100개의 차량이 움직이는 상황을 기준으로 생성했다. Range image를 64 x 512로 나누고 기본 4개의 channel(i.e., x, y, z, and depth)에서 residual image channel을 3개 추가하는 것으로 총 7의 channel를 input으로 사용한다. 출력으로는 움직이는 물체의 이진 마스크와 움직이는 부분이 가리는 부분이 inpainting된 배경 정보(i.e., x, y, z)를 생성한다. 학습은 NVIDIA RTX 3090을 이용해 Pytorch 기반으로 100회 진행했으며 Adam optimizer를 통해서 학습을 진행했다. hyperparameters λ_m, λ_s는 각각 5, 0.01으로 진행했다.

앞으로 제안된 방법은 "Proposed", 비교군은 "SAM-Net"으로 표기한다. Segmentation의 결과는 percision[%], recall[%], 그리고 F1-score[%]를 통해 비교하며, inpainting 결과는 root mean square error (RMSE) [mm], mean absolute error (MAE) [mm], iRMSE [1/km], iMAE [1/km]를 이용해 비교한다. 마지막으로 위치 추정의 오차는 RMSE[m, degree]와 Stand Deviation(S.D)[m, degree]을 기준으로 한다.

제 4.2 분할 성능 비교(Segmentation Performance)

그림 4-1 과 표 4-1는 dynamic 환경에서 Proposed와 SAM-Net의 성능을 보여준다. 그림 4-1의 (a),(b),(c)는 서로 다른 수의 움직이는 차량과 보행자를 포함하고 있으며 다른 속도로 움직이고 있다. 표 4-1은 분류성능평가지표를 통해 두 개의 모델의 성능을 비교한다. 두 모델은 마스크의 ground truth를 움직이는 물체로 학습을 진행했으며 residual image,와 단일 이미지 학습의 차이를 보여준다. 제안 방법의 경우, residual image를 통해 물체의 움직임을 추출해 이진 마스크를 생성하기 때문에 높은 성능을 보인다. 반면 단일 이미지를 이용한 SAM-Net의 경우, 물체의 움직임을 학습할 수 없기 때문에 움직이는 물체에 대한 특징을 추출하지 못해 제대로 된 결과를 생성하지 못한다.

표 4-1 기존 알고리즘과 Segmentation 성능 비교

Table 4-1 Comparison of the precision[%],recall[%], and F1-Score[%] of Proposed model against SAM-Net

	Precision	Recall	F1-Score
SAM-Net	0.8564	0.8676	0.8619
Proposed	**0.8626**	**0.9146**	**0.8878**

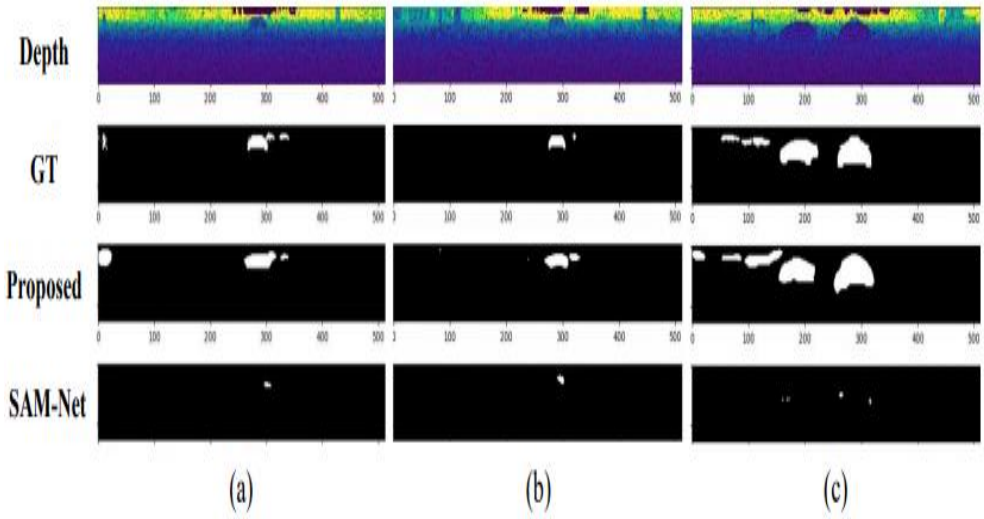


그림 4-1 Segmentation 성능 비교

Figure 4-1 Comparison of Segmentation Result

제 4.3 인페인팅 성능 비교(Inpainting Performance)

Inpainting network를 통해서 생성된 출력은 x, y, z로 생성된다. 결과를 비교하기 위해서 3개의 channel을 이용해 각각의 포인트의 거리 값으로 변환한다. 그림 4-2는 2차원과 3차원의 inpainting 결과이다. 붉은 박스는 움직이는 물체를 표현하고 있으며 해당 부분의 inpainting 결과를 비교한다. 그림 4-2에서 Raw는 움직이는 물체를 포함하는 이미지, GT는 움직이는 물체를 제거한 ground truth, Inpainted는 포인트 클라우드를 복원한 이미지이다. Diff는 GT와 Inpainted의 차이를 나타내며 Proposed에서 더 작은 차이를 보인다. 이는 표 4-2에서 정량적인 수치로 확인할 수 있다. 그림 4-2의 inpainting의 3차원 결과에서 SAM-Net은 지표나 모서리 특징을 잘 유지하지 못하는 것을 확인할 수 있다. 반면 Proposed에서는 smoothness loss를 통해 포인트 간의 상관 관계를 학습했기 때문에 포인트의 특징이 연속적인 것을 확인할 수 있다. Smoothness loss의 포인트의 연관성 유지는 그림 4-3을 통해 확인할 수 있다. 그림 4-3에서 푸른색 점은 배경에 포함된 포인트, 주황색 점은 ground truth를 뜻하며 붉은색 점은 inpainting 결과를 통해 생성된 점을 보여준다.

표 4-2 기존 알고리즘과 inpainting 성능 비교

Table 4-2 Comparison of RMSE[mm], MAE[mm], iRMSE [1/km], iMAE [1/km] of Proposed model against SAM-Net

	RMSE	MAE	iRMSE	iMAE
SAM-Net	3.3318	0.3612	5.8459	1.7117s
Proposed	**2.1930**	**0.3612**	**2.9843**	**0.6470**

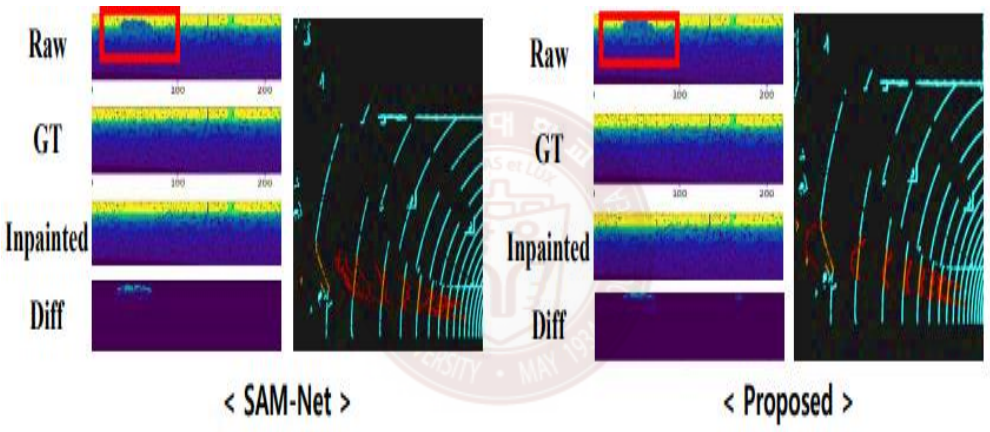
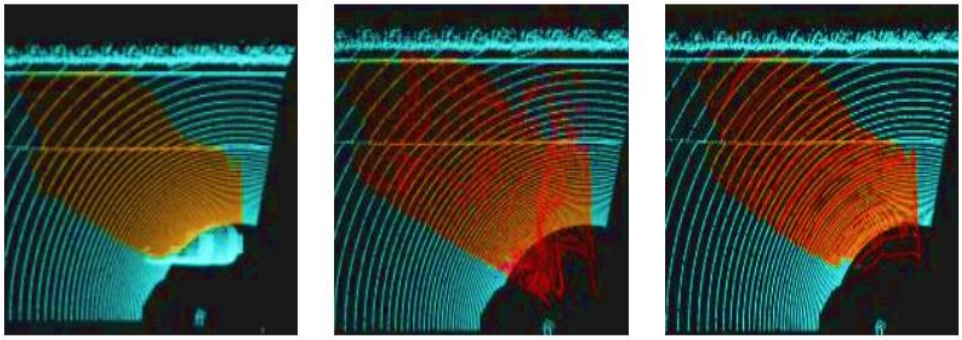


그림 4-2 Inpainting 결과 비교

Figure 4-2 Comparison of Inpainting **Result**



< Raw data > < SAM-Net > < Proposed >

그림 4-3 Smoothness Loss를 통한 연속성 비교

Figure 4-3 Comparison of Smoothness Loss

제 4.4 SLAM 성능 비교(SLAM Performance)

표 4-3 은 LeGO-LOAM 과 제안한 모델을 같이 이용한 방법의 비교 결과이다. 제안한 모델을 같이 사용하는 경우, 움직이는 물체가 많은 환경에서 더 좋은 성능을 보였다. 특히 여러 대의 움직이는 물체에 둘러싸여 있는 환경에서 움직이는 물체를 제거하고 새로운 포인트를 생성해 특징점 매칭을 진행하는 것으로 더 좋은 경로를 추정한다. 그림 4-4 는 Town_1, Town_4 에서의 추정 경로이다. Town_4 의 경우, 움직이는 물체의 속도가 높은 상태에서 회전을 진행하게 되어 전체적으로 위치 추정의 성능이 떨어진다. 하지만 기존의 방법은 속도와 주위의 움직이는 물체로 인해 회전하는 부분에서 직진으로 판단한 반면 제안 방법은 inpainting 을통해 유효한 포인트를 생성해 바르게 회전을 진행했다.

표 4-3 기존 알고리즘과 위치 추정 성능 비교

Table 4-3 Comparison of RMSE[m, degree], S.D[m, degree] of Proposed Model against LeGO-LOAM

Town.	LeGO-LOAM		Proposed	
	RMSE	S.D	RMSE	S.D
Town_1	0.569 / 0.861	0.303 /0.695	**0.365 / 0.857**	**0.214 / 0.625**
Town_4	394.0 / -	240.5 / -	**92.9 / 102.6**	**56.3 / 82.0**

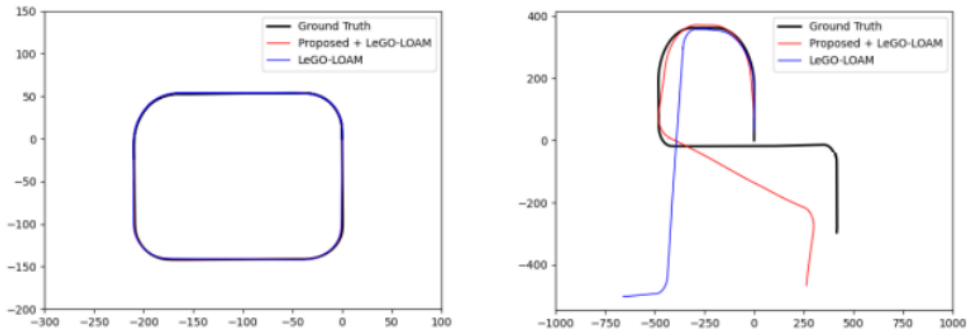


그림 4-4 Town1(a), Town4(b)의 이동 경로

Figure 4-4 Trajectory of Town1(a), Town2(b)

제 4.5 KITTI에서의 슬램 성능 비교(Performance on KITTI)

실제 환경에서의 모델의 성능을 확인하기 위해 2011_09_26_drive_0015'(seq_30),'2011_09_26_drive_0028'(seq_32), '2011_09_26_drive_0029'(seq_33)[17] 에서 실험을 진행했다. 그림 4-5 는 기존 방법과 제안 방법의 추정 경로를 보여주며 표 4-6 은 비교 결과이다. seq_30, seq_32 는 제안 방법을 통해 더 낮은 결과를 얻을 수 있다. 반면 seq_33 의 경우, 나무의 inpainting 결과가 frame 당 변하게 되어 초기 위치 추정에 문제가 생긴다. 이를 해결하기 위해서 실제 dataset 을 통한 학습을 진행해 일반화 성능을 높일 필요가 있다.

표 4-4 KITTI에서의 기존 알고리즘과 위치 추정 성능 비교

Table 4-4 Comparison of RMSE[m, degree], S.D[m, degree] of Proposed Model against LeGO-LOAM in KITTI

Seq.	LeGO-LOAM		Proposed	
	RMSE	S.D	RMSE	S.D
30	0.604 / **0.456**	0.307 / 0.526	**0.456** / 0.547	**0.207 / 0.522**
32	19.174 / 0.381	12.325 / 0.256	**14.411 / 0.314**	**10.462 / 0.182**
33	6.734 / 1.311	3.124 / 1.068	**3.504 / 1.002**	**1.283 / 1.060**

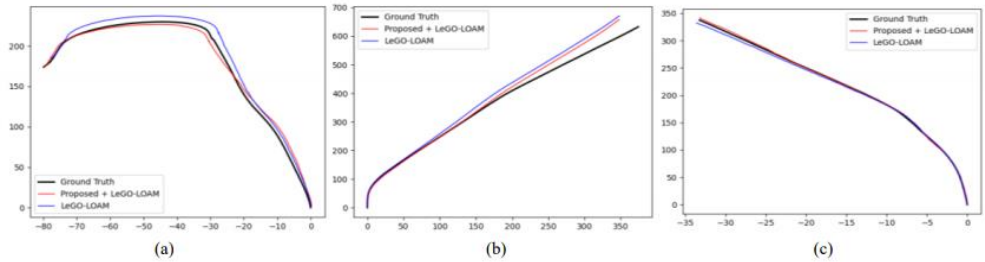


그림 4-4 seq_30(a), seq_32(b), seq_33(c)의 이동 경로

Figure 4-4 Trajectory of seq_30(a), seq_32(b), seq_33(c)

제 5 장 결론 및 추후 연구

본 논문은 동적 환경에서 움직이는 물체를 분류 및 제거하고 유효한 point 로 복원하는 새로운 LiDAR point cloud inpainting model 을 제안했다. 제안 모델은 residual images 를 이용해 움직이는 물체의 움직임을 학습하고 segmentation decoder 를 통해 이진화 된 마스크로 생성한다. 이후, 생성된 마스크를 이용해 움직이는 물체를 제거하고 빈 부분을 inpainting decoder 를 통해서 복원하며 smoothness loss 를 이용해 기존 방법보다 자연스러운 결과를 생성한다. 제안 모델은 CARLA 시뮬레이션을 이용해 생성한 2 개의 dataset 에서 보다 나은 동적 물체 segmentation 결과와 inpainting 성능을 보인다. 이후, 동일한 데이터셋에서 SLAM 알고리즘에 제안 모델을 전처리로 사용하는 것으로 움직이는 물체가 많은 환경에서 성능의 향상을 확인하였다. 또한 KITTI dataset 에서 SLAM 의 성능 통해 제안 방법의 유효성을 확인했다.

앞으로는 inpainting 성능을 높이기 위해서 특징점을 반영하는 것 뿐만 아니라 비슷한 특징을 공유하는 정적인 물체를 분류해 해당 특징을 반영하는 방법을 연구할 예정이다. 또한 ground truth 정보를 얻는 한계로 인해 시뮬레이션 데이터만을 이용했지만 새로운 데이터 생성 방법을 이용해 benchmark 데이터셋을 이용해 결과를 확인할 예정이다.

참고 문헌

[1] Zhang, Ji, and Sanjiv Singh. "LOAM: Lidar odometry and mapping in real-time." Robotics: Science and Systems. Vol. 2. No. 9. 2014.

[2] Shan, Tixiao, and Brendan Englot. "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain." *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018.

[3] Dewan, Ayush, Tim Caselitz, and Wolfram Burgard. "Learning a local feature descriptor for 3d lidar scans." *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018.

[4] Sun, Jiadai, et al. "Efficient Spatial-Temporal Information Fusion for LiDAR-Based 3D Moving Object Segmentation." *arXiv preprint arXiv:2207.02201* (2022).

[5] Rashed, Hazem, et al. "Fusemodnet: Real-time camera and

lidar based moving object detection for robust low-light autonomous driving." *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019.

[6] Dang, Xiangwei, et al. "Moving objects elimination towards enhanced dynamic SLAM fusing LiDAR and mmW-radar." *2020 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*. IEEE, 2020.

[7] Chen, Xieyuanli, et al. "Moving object segmentation in 3D LiDAR data: A learning-based approach exploiting sequential data." *IEEE Robotics and Automation Letters* 6.4 (2021): 6529-6536.

[8] Lee, Junhyeop, et al. "SAM-Net: LiDAR Depth Inpainting for 3D Static Map Generation." *IEEE Transactions on Intelligent Transportation Systems* (2021).

[9] Dosovitskiy, Alexey, et al. "CARLA: An open urban driving simulator." *Conference on robot learning*. PMLR, 2017.

[10] Geiger, Andreas, et al. "Vision meets robotics: The kitti

dataset." *The International Journal of Robotics Research* 32.11 (2013): 1231-1237.

[11] Steder, Bastian, et al. "NARF: 3D range image features for object recognition." *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Vol. 44. 2010.

[12] Rusu, Radu Bogdan, Nico Blodow, and Michael Beetz. "Fast point feature histograms (FPFH) for 3D registration." *2009 IEEE international conference on robotics and automation*. IEEE, 2009.

[13] Paz, Lina María, et al. "EKF SLAM updates in O (n) with Divide and Conquer SLAM." *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007.

[14] Chen, Qi-Ming, et al. "An improved particle filter SLAM algorithm for AGVs." *2020 IEEE 6th International Conference on Control Science and Systems Engineering*

(ICCSSE). IEEE, 2020.

[15] Milioto, Andres, et al. "Rangenet++: Fast and accurate lidar semantic segmentation." *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019.

[16] Ruchti, Philipp, and Wolfram Burgard. "Mapping with dynamic-object probabilities calculated from single 3d range scans." *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

[17] Chen, Xieyuanli, et al. "Suma++: Efficient lidar-based semantic slam." *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019.

[18] Liu, Hongyu, et al. "Pd-gan: Probabilistic diverse gan for image inpainting." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.

[19] Lahiri, Avisek, et al. "Prior guided gan based semantic inpainting." *Proceedings of the IEEE/CVF Conference on*

Computer Vision and Pattern Recognition. 2020.

[20] Bescos, Berta, et al. "Empty cities: Image inpainting for a dynamic-object-invariant space." *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[21] Bescos, Berta, Cesar Cadena, and Jose Neira. "Empty cities: A dynamic-object-invariant space for visual SLAM." *IEEE Transactions on Robotics* 37.2 (2020): 433-451.

[22] Qiu, Jiaxiong, et al. "Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.

[23] Xu, Yan, et al. "Depth completion from sparse lidar data with depth-normal constraints." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

[24] Ma, Fangchang, Guilherme Venturelli Cavalheiro, and Sertac Karaman. "Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera." *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[25] Li, Ying, et al. "Deep learning for lidar point clouds in autonomous driving: A review." *IEEE Transactions on Neural Networks and Learning Systems* 32.8 (2020): 3412-3432.

[26] Maddern, Will, et al. "1 year, 1000 km: The Oxford RobotCar dataset." *The International Journal of Robotics Research* 36.1 (2017): 3-15.

[27] Behley, Jens, et al. "Semantickitti: A dataset for semantic scene understanding of lidar sequences." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

[28] Deschaud, Jean-Emmanuel. "KITTI-CARLA: a KITTI-like dataset generated by CARLA Simulator." *arXiv*

preprint arXiv:2109.00892 (2021).