RGB and LiDAR Fusion-based 3D Semantic Mapping

Bingqi Sun
Stanford University
sbq@stanford.edu

Ruochen (Chloe) Liu
Stanford University
ruochenl@stanford.edu

Yan Wang
Stanford University
yan12@stanford.edu

Abstract

3D mapping remains a popular technology and has wide applications in areas of autonomous vehicles and robotics. Among all types of maps, including occupancy grid map, occupancy continuous map and semantic grid map, semantic map is arguably the most informative because it provides critical environmental information by classifying each grid into different classes. Mapping with LiDAR is common as LIDAR can provide high frequency range measurements where errors are relatively constant irrespective of the distances measured. However, map constructed from LiDAR is in lack of texture and occupies huge space. Also, 3D segmentation itself is one of the most challenging vision tasks. In this project, we propose a method to combine LiDAR information and RGB labels. We first obtain semantic labels for the RGB image with a classic U-Net network, and then create a projection between RGB and LiDAR data as well as generate a nice visualization. At the end of this report, we present some decent results and analysis on limitation and future work of this direction. We release our source code on Github: https://github.com/sunbingqi/CS231A_Final.

1. Introduction

The last few years have seen enormous progress in the 3D sensing capabilities of autonomous vehicles. Mature and robust LiDAR and INS technologies give self-driving vehicles an accurate and real-time sense of the geometric structure around them, immensely simplifying navigation related tasks. Mapping with LiDAR is common as LiDAR can provide high frequency range measurements where errors are relatively constant irrespective of the distances measured. [7]

However, due to the properties of LiDAR, it has several disadvantages comparing with RGB-D camera and stereo vision, such as point clouds sparsity, lack of meaning of a single point in the entire point cloud. Hence, using both LiDAR and RGB can potentially overcome the individual disadvantages of each sensor by mutual improvement and yield robust features which can improve the matching process. [10]

In this project, we investigate one potential way of improving LiDAR data by adding semantic information extracted from RGB images. We hypothesize that by taking the advantage of the relatively mature techniques in 2D image segmentation and creating a projection between LiDAR point clouds and RGB images, we are able to build a 3D semantic map, which can potentially have wide applications in areas of autonomous vehicles and robotics. Since 3D segmentation itself is one of the most challenging vision tasks

2. Related Work

2.1. Semantic Segmentation

Semantic Segmentation is the process of assigning a label to every pixel in the image. This is in contrast to classification, where a single label is assigned to the entire picture. Semantic segmentation treats multiple objects of the same class as a single entity.

Before the deep learning era, some image processing techniques were used to segment images into regions of interest. For example, the split and merge algorithm recursively splits an image into sub-regions until a label can be assigned, and then combines adjacent sub-regions with the same label by merging them. Later, Conditional Random Fields (CRF) were applied to image segmentation. Even today, CRF is still used as a post-processing tool to improve the performance of different algorithms.[12]

Recently with the development of deep learning in computer vision area, CNN makes great contribution to semantic segmentation. In 2015, Long et al. introduced how to use Fully Convolutional Networks (FCN) for semantic segmentation. They used FCN to first downsample the input image to a smaller size through a series of convolutions (encoder). The encoded output is then upsampled either through bilinear interpolation or a series of transpose-convolutions (decoder). This architecture makes the segmentation map generate images of any size. Despite effective, the architecture has some drawbacks. First, due to the uneven overlap of the output of the transpose-convolutions, there are checkerboard artifacts. The second drawback is poor resolution at the boundaries due to loss of information from the process of encoding.[6]

Researchers proposed two different architectures to solve the problems in FCN. The first one is the encoder-decoder architecture (U-Net). This architecture builds the connection between downsampling path and upsampling path so that the decoder can recover details better.[11] The Tiramisu Model (DenseNet) is similar to the U-Net but it uses Dense Blocks for convolution and transposed-convolutions.[5]

The other class of solution is dilated convolution. Dilated convolution method removes pooling layers so that some feature can be preserved. Dilated Convolutions present an efficient method to combine features from multiple scales without increasing the number of parameters by a large amount. By adjusting the dilation rate, the same filter has its weight values spread out farther in space. This enables it to learn more global context.[1]

2.2. Mapping

In real world, multiple types of Robotics application requires a 3D model of surrounding environments, which we would call as mapping, such as airborne, underwater, outdoor, or extra-terrestrial missions. However, there exist few readily available, reliable, and efficient implementations. Researchers have proposed several works to fuse semantic information into mapping process, most of which are probabilistic based.

2.3. 3D Segmentation

3D segmentation is a fundamental and challenging problem in computer vision with applications in autonomous driving, robotics, augmented reality and medical image analysis. 3D segmentation methods can be divided into five categories according to the data representation used: RGB-D image based, projected images based, voxel based, point based and other representations based.[4]

PointNet is a a point based 3D segmentation method that directly processes point clouds. It uses shared MLP to exploit points-wise features and adopts a symmetric function such as max-pooling to collect these features into a global feature representation. Because the max-pooling layer only captures the maximum activation across global points, PointNet cannot learn to exploit local features.[8]

Building on PointNet, PointNet++ defines a hierarchical learning architecture. It hierarchically samples points using farthest point sampling and groups local regions using k nearest neighbor search as well as ball search. PointNet++ framework expands the receptive field to exploit more local features collectively.[9]

3. Approach

In this paper, we propose an approach that uses RGB images and LiDAR point clouds together to build a mapping with semantic information. First, we train a U-Net model to do 2D image segmentation. Then we combine 2D segmentation results with LiDAR point clouds to build a full semantic mapping.

3.1. U-Net

U-Net is a convolutional neural network that was originally developed for biomedical image segmentation. The network is based on the fully convolutional network and its architecture was modified and extended to work with fewer training images and to yield more precise segmentations.[11]

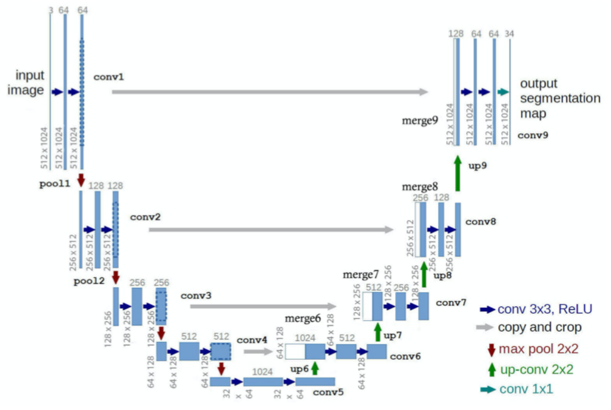

Figure 1. U-Net Architecture [11]

The encoder-decoder architecture of U-Net builds the connection between the downsampling section and upsampling section so that the decoder can better recover the details of the images. The special architecture generally produces a decent performance on various segmentation applications.

The U-Net architecture is shown in Figure 1. It consists of a contraction path and an expansion path. The contraction path consists of several contraction blocks. Each contraction block has two 3x3 convolutional layers followed by a 2x2 max pooling layer. The number of feature maps after each block doubles so that the architecture can learn the complex structures effectively. Similarly, the expansion section consists of several expansion blocks; each of them passes the input into two 3x3 convolutional layers followed by a 2x2 upsampling layer. Thus, after each block, the number of feature maps is reduced by half. Notice that the cropped feature maps from the contraction section get concatenated to ensure that the features that are learned while contracting the image will be used to reconstruct it. Finally, the resultant mapping passes through two 3x3 Convolutional layers. [11]

To train the model faster, we set the size of the input real-world RGB images as 512x1024x3. Since we have 34 different classes of objects, the shape of the outputs is 512x1024x34, which means for every pixel in the images we can get a label and we will use these labels for semantic mapping. However, because the Cityscapes real world images and labeled images are of size 1024x2048x3, we reshape their sizes to fit into our U-Net model. [2] During the training, we use cross entropy loss and the Adam optimizer.

3.2. LiDAR Mapping

In the LiDAR mapping section, we want to combine the results from the U-Net and LiDAR point clouds to build a full semantic mapping for a sequence of frames, which is illustrated in Figure 2. We will focus on the operation on a single frame of data first. Then we will introduce how to put all the frames in a sequence together to generate the final semantic mapping. A single frame of data consists of a raw image, LiDAR points for that frame, and its ground truth pose.

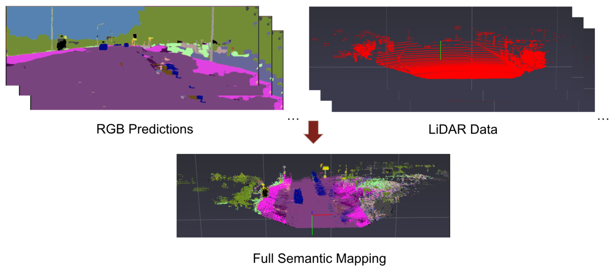

Figure 2. LiDAR Mapping

First, we use the trained U-Net model to generate semantic predictions for the raw image. Then, we use Equation 1 to transform the LiDAR points to the camera frame as in Figure 3. In Equation 1, K is the 3x4 camera matrix, d is the depth value, and T is the projection matrix from the LiDAR coordinates to the camera coordinates. We can extract the depth value d after projecting LiDAR points into camera frame from the third dimension of the projected point. By this transformation, we can get the corresponding pixel of the LiDAR point.

$$\text{point}_{\text{pixel}} = \frac{1}{d}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{d}KT\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

After transforming a LiDAR point to the camera frame, we can extract the semantic label for that pixel and associate the semantic label with this LiDAR point. Figure 4 displays the result after coloring the LiDAR points in one frame. For all the LiDAR points in the single frame, we extract their semantic labels.

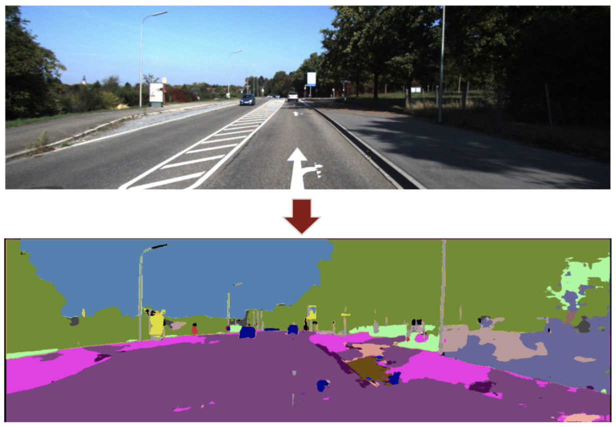

Figure 3. U-Net results on KITTI Odometry Dataset

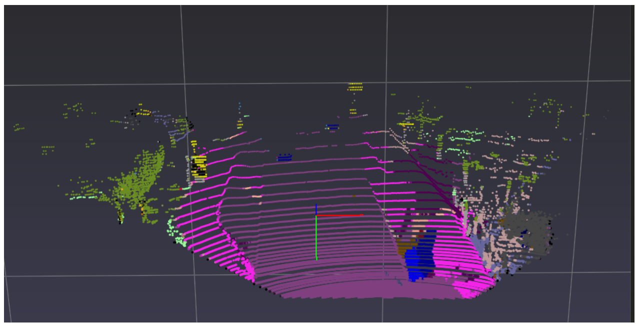

Figure 4. Mapping for one frame

$$\text{point}_{\text{pose}} = \text{pose} \cdot \text{point}_{\text{LiDAR}} \tag{2}$$

The next step is to connect all the frames of labeled LiDAR points together. The ground truth pose is the camera's pose in the world reference system, so we use this pose transformation matrix to get the coordinates for every LiDAR point in every frame of data, which is illustrated by Equation 2. $\text{point}_{\text{LiDAR}}$ is a 4x1 vector, the result $\text{point}_{\text{pose}}$ is a 3x1 vector, and the pose matrix is 3x4. Since we have found the semantic label for every LiDAR point, we can associate the corresponding RGB values for that class with the coordinates information.

3.3. Visualization

Now that we have collected the coordinates and color information for all the LiDAR points in a sequence, visualization is our final step. Due to the huge amount of LiDAR cloud points in a sequence, an efficient visualization tool is necessary. We choose to use Point Processing Toolkit

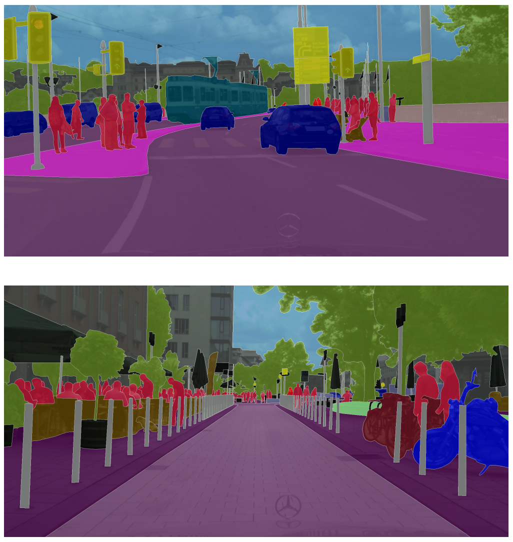

Figure 5. Example labeled image in Cityscapes. [2]

(PPTK) implemented in Python to do the final visualization for our full semantic mapping. The viewer supports interactive visualization of tens of millions of points via an octree-based renderer, which provides efficient and interactive visualization. Figure 4 is an example of LiDAR points visulization based on PPTK.

4. Experiments

4.1. Datasets

We work with two datasets in this project. To train and test the U-Net that we implemented from scratch, we use the Cityscapes dataset, which contains 5000 images (1024x2048x3) in total, including 2975 images for training, 1525 images for validation and 500 images for testing. Each image in the dataset is annotated with a ground-truth segmentation map (1024x2048x3) for classified objects (34 classes) and pixel-wise category IDs. [2] We then apply our model on the RGB odometry benchmark of the KITTI Odometry dataset. The odometry benchmark consists of 22 stereo sequences taken per frame in scenes of cities, saved in png format: it provides ground truth trajectories, Velodyne laser data, RGB images, ground truth poses, and calibration files. The shape of images in the KITTI data set is 375x1242x3. [3]

For the second part of our project we only work on KITTI dataset since CityScapes dataset does not provide LiDAR data. We will generate the final semantic map based on the prediction of the U-Net. The reason we used two datasets is because KITTI Odometry dataset only provides 200 semantically annotated images, which is not sufficient for our U-Net training.

4.2. Evaluation Metrics

For the evaluation of the U-Net, we used the Intersection over Union (IoU), which is a commonly-used metric to measure the accuracy of object detection. It is defined by

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} \tag{3}$$

where area of intersection is the area where our predicted image overlaps the ground truth image, while the area of union is the area combining the predicted image and the ground truth image. Generally, accurate detection will have a value toward 1. More concretely, we can also interpret the metric as

$$\text{IoU} = \frac{\text{TP}}{(\text{TP+FP+FN})} \tag{4}$$

where TP, FP, and FN are the numbers of true positive, false positive, and false negative pixels, respectively. For each image, we compute IoU of different categories, and we average the IoU for each category across all images. Note that we have 34 classes in total, but some of the 34 classes are not present in many images in the dataset, which generates a lot of noises in the prediction, so we will not use the common mean IoU as our main metric.

The evaluation of our final LiDAR mapping is qualitatively done by showing the 3D semantic map using the visualization tool described in 3.3. We consider various cases in the dataset and analyze how well our mapping performs in different situations.

4.3. Results

4.3.1 U-Net

We trained U-Net for 110 epochs and evaluated the result on the Cityscapes validation set. We observe that for classes that are more common in the scene the model achieves better performance, Table 1 shows the top-10 classes and the corresponding IoU scores. For less common classes, such as ”rail track” and ”tunnel”, the IoU scores are much lower. Overall the performance of our model is satisfactory, with a maximum category IoU of 0.8023. Figure 9 visualized the model prediction. Although there are still some artifacts visible on the boundary of the objects, especially when the lighting condition is complex, our model is able to correctly segment the major parts of each image.

4.3.2 LiDAR Mapping

To evaluate the quality of our 3D semantic map, we choose two different sequences from the KITTI Odometry dataset. Here we include screenshots from the scene and our 3D semantic map. Since the scenes in the KITTI Odometry are by nature part of a sequence, we stitched the images together to recreate the sequence video and generated an interactive

Class Name	IoU
vegetation	0.8023
road	0.7911
building	0.7468
sky	0.7048
car	0.6390
sidewalk	0.4823
traffic sign	0.4218
pole	0.3744
person	0.3374
license plate	0.2496

Table 1. Classes with the top-10 highest IoU

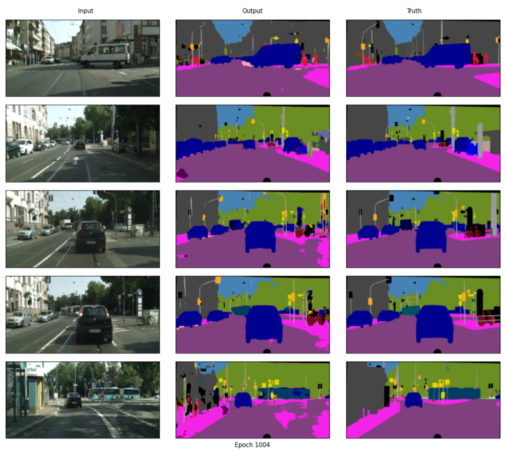

Figure 6. Prediction result after epoch 110, compared with the raw image and ground truth labels

3D map from this whole sequence. We submit these demos as supplementary materials. Figure 7 shows the raw image and 3D semantic map generated with our method on KITTI sequence 00. We can see that the map provides important information about the environment with explicit segments that represent different objects in the scene. Note that in this map, we see this colorful strip in the center of the lane. This can be attributed to the moving motorcycle in the scene. It's velocity and distance from the camera in the scene are constantly changing, which prevents the model from providing a good static map. This discovery might uncover a potential shortcoming of our method: it is not good at handling dynamic objects in the scene. Figure 8 shows the mapping result on KITTI sequence 02. This result is worth mentioning because the camera turns right in the middle of this sequence. After making the right turn, the camera is in the face of a completely different scene, and the lighting condition also changes abruptly. Thus, we suspect that this sequence is harder for our model to generate the map compared to the previous sequence, in which the surrounding area of the camera is relatively static. The visualized map proves that our method is robust enough to deal with this harder situation, as it successfully captures the semantic information of both the original road and the new road appeared only after the right turn.

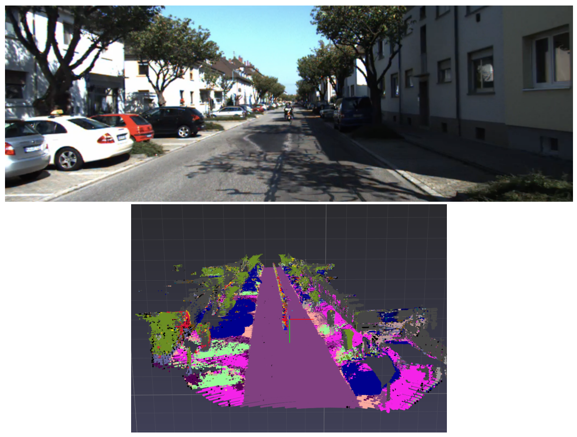

Figure 7. 3D Semantic LiDAR map for KITTI sequence 00

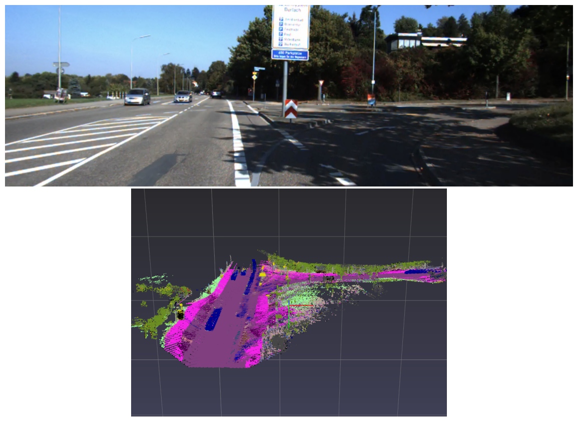

Figure 8. 3D Semantic LiDAR map for KITTI sequence 02

We noticed that in Figure 7 and Figure 8, the labels look a little noisy. We suspect that this is because we include whole 34 classes in the scene. To further improve the map, we tried merging similar classes. For example, we merged ”road”, ”sidewalk”, ”parking”, ”rail track” since they all have similar appearance and function. We also merged all kinds of vehicles including ”car”, ”truck”, ”bus”, etc. Besides, we merged ”person” and ”rider”, ”vegetation” and ”terrain”. After merging, the 34 classes are reduced to 8. Figure 9 shows the mapping result after the classes were merged. We can see that the map is much less noisy with fewer number of different colors, so that it provides more useful information regarding the major categories of objects in the scene.

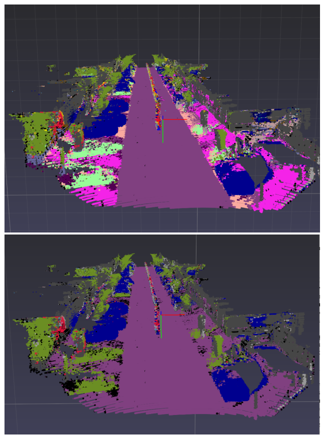

Figure 9. Before and after merging labels

5. Discussion

While we were working on this project, we did encounter some difficulties that we want to discuss in this report. Firstly, the training time of U-Net is much longer than we expected while we do not have sufficient hardware resources. Each epoch took around 1 hour to train on Google Colab using GPU, and we had to train more than 100 epochs to achieve a decent result. Another issue we encountered came from switching between dataset. Although we decided to use KITTI dataset at the very beginning, we soon realized that it only contains 200 semantically annotated images, so that we had to choose another dataset for the pretraining purpose. We decided to go with Cityscapes because it has sufficient number of images with pixel-level semantic labels and also shares the data format with KITTI. However, categories of scene objects in these two datasets are slightly different and the light regimes are not the same neither, which reduces the model performance when it was transferred from Cityscapes to KITTI.

Despite these difficulties, we still made several contributions in this project. We were able to implement and train a U-Net model from scratch and achieve acceptable evaluation results. With the prediction of the U-Net, we successfully generated a decent result of fusing semantic information into LiDAR point cloud. We demonstrated the potential of a simple U-Net model on downstream tasks. Also, we confirmed our hypothesis that the relatively mature techniques in 2D image segmentation could be helpful in generating 3D segmentation. We only conducted a straightforward projection between RGB pixels and point clouds and the generated mapping is already good enough to clearly classify major categories of objects in the environment. We hope that more complicated approach could lead to even better result. We believe this approach could be useful in many real world applications, while more auxiliary techniques would be needed to solve problems such as dealing with moving objects and estimating a trajectory for traveling vehicles while mapping.

References

[1] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[3] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[4] Y. He, H. Yu, X. Liu, Z. Yang, W. Sun, Y. Wang, Q. Fu, Y. Zou, and A. Mian. Deep learning based 3d segmentation: A survey. *arXiv preprint arXiv:2103.05423*, 2021.

[5] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[6] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[7] D. Maturana, P.-W. Chou, M. Uenoyama, and S. Scherer. Real-time semantic mapping for autonomous off-road navigation. In *Field and Service Robotics*, pages 335–350. Springer, 2018.

[8] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[9] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[10] R. Rishav, R. Battrawy, R. Schuster, O. Wasenmüller, and D. Stricker. Deeplidarflow: A deep learning architecture for scene flow estimation using monocular camera and sparse lidar. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10460–10467. IEEE, 2020.

[11] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[12] H. M. Wallach. Conditional random fields: An introduction. *Technical Reports (CIS)*, page 22, 2004.