Project1

Mapping and perception for an autonomous robot/ 0510-7591

By: Tomer Shimshi

Tel Aviv University

August 2023

Abstract:

The aim of this project is to gain hands-on experience with the kitti dataset, in my case the **City 2011_09_26_drive_0104,** and to implement the algorithm for robot mapping. The project is divided into three parts: Geodetic coordinate system and KITTI dataset, Probabilistic Occupancy Grid, and Sensor fusion and semantic segmentation. In the first part, we analyze the technical details of our scenario, display the trajectory in ENU and NED representation, and show one example each of image and point cloud from our record. In the second part, we implement a grid on a single scan and update and shift the Occupancy Grid Map according to the next pose of the car. We analyze our results by showing examples of good and bad mapping and changing the parameters of the model. In the third part, we use a pretrained NN for road segmentation from an input image and project LiDAR Point clouds onto image coordinates to apply this segmentation onto the LiDAR points. We attach examples of the projection onto image coordinates. Through this project, we aim to gain a deeper understanding of the practical applications of the concepts learned in class and improve our skills in analyzing and implementing algorithms for robot mapping.

Contents

List of figures

Part A: Geodetic coordinate system and get familiar with the KITTI dataset

A:

The dataset I received is **City 2011_09_26_drive_0104**, the scenario entails driving along a linear route within an urban environment. The dataset encompasses parked cars situated along the curbside, moving vehicles on the roadway, and a sparse population of pedestrians navigating the sidewalks. Notably, there are few occluded regions captured in the dataset

B:

We can see in the image the trajectory of the car that collected the data for this test.

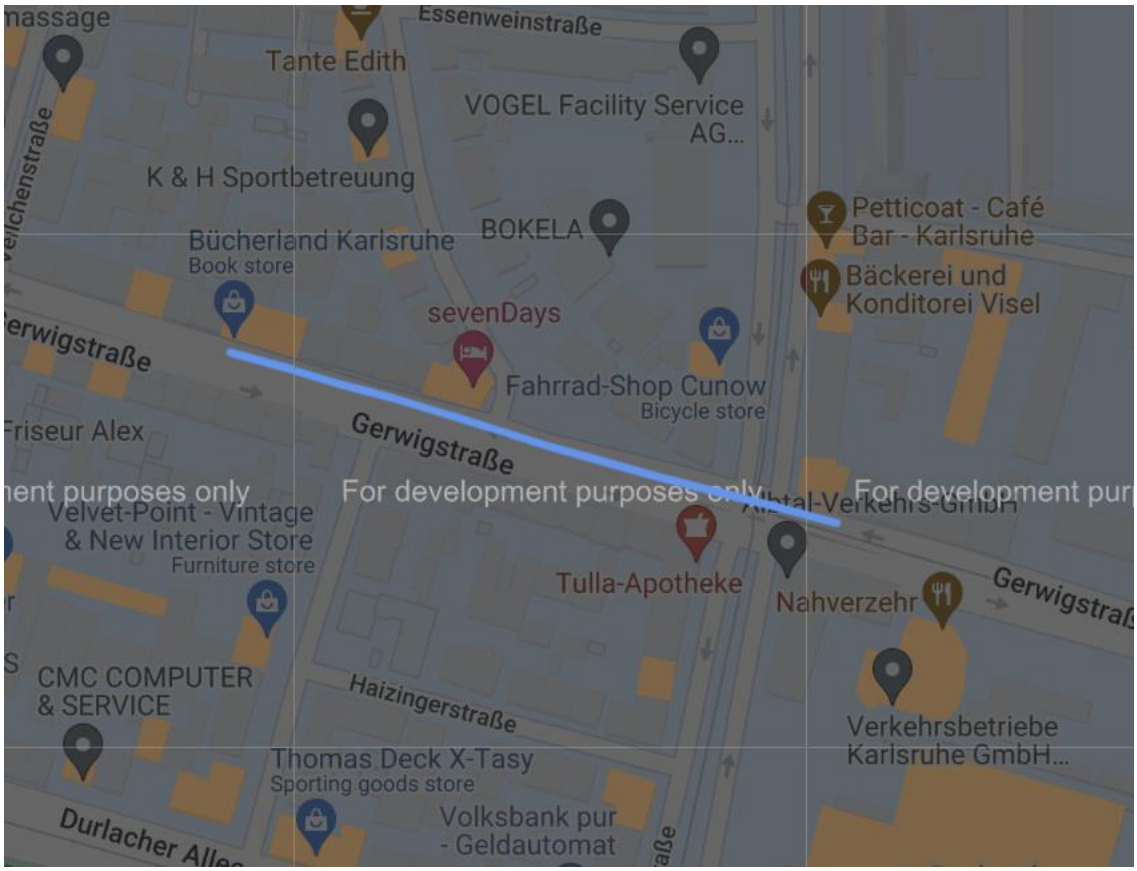


Figure 1 car trajectory

C:

Now we shall display the trajectory in ENU and NED representation of local coordinates, right

side angular velocity (to the right), pitch, roll and yaw

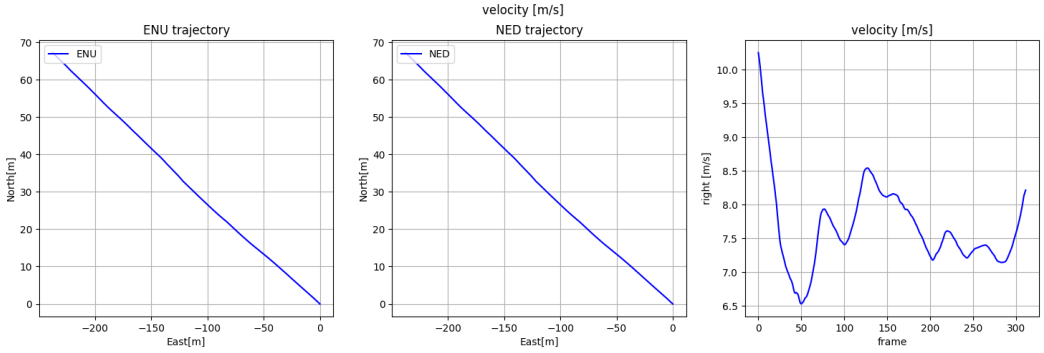


Figure 2 Positions

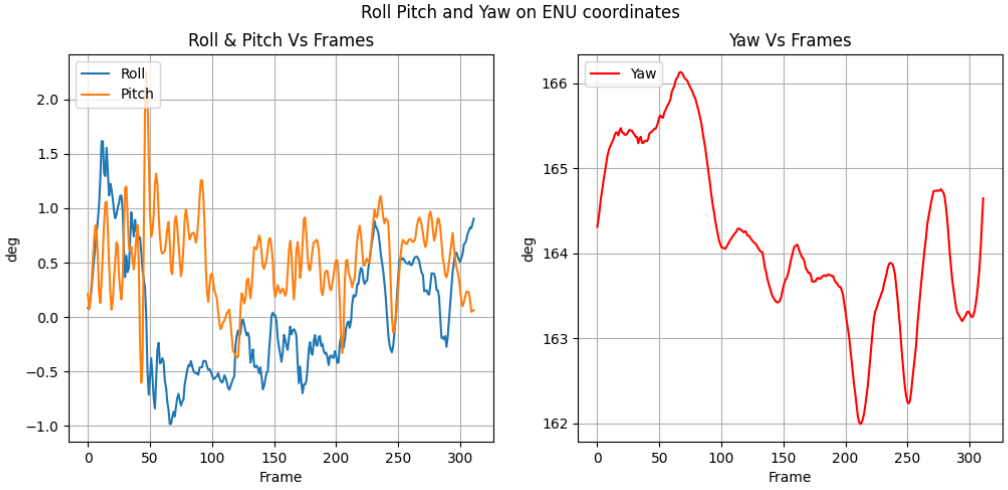


Figure 3 Roll Pitch and Yaw on ENU coordinates

Upon reviewing the attached images, it is evident that the vehicle captured in the dataset followed a linear trajectory, as demonstrated by the Roll & Pitch versus Frames graph. Notably, minor perturbations observed in the pitch and roll can be attributed to minor imperfections in the road surface or sensor noise. Analysis of the yaw data suggests that occasional sensor noise may have impacted individual frame readings. However, the average yaw reading indicates that the vehicle moved in a direction of approximately 164 degrees, which corresponds to the trajectory displayed in the NED and ENU graphs.

D:

Attached below is the quality of the GPS signal reception.

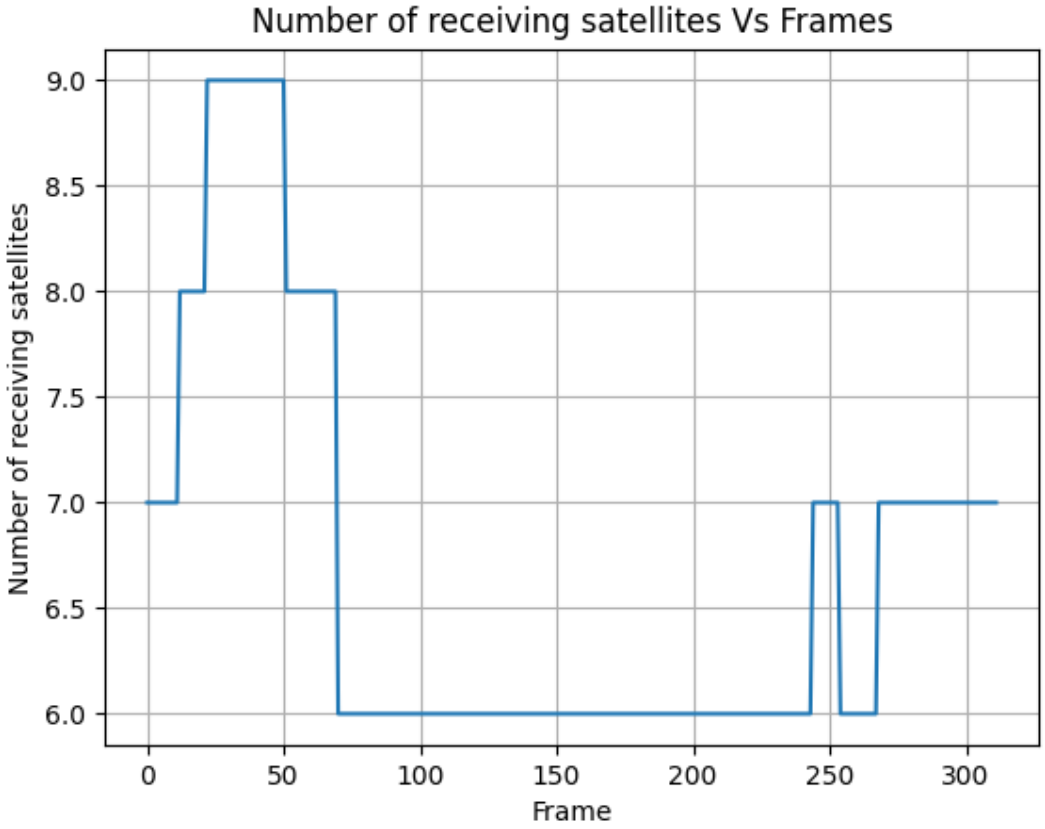


Figure 4 Number of receiving satellites Vs Frames

Upon inspection of the provided image, it is evident that a minimum of six satellites were in view during each phase of the data recording process. Given that a minimum of four satellites is required to establish a reliable GPS signal, the received readings can be deemed as satisfactory.

E:

We will now show the first image and point clod received

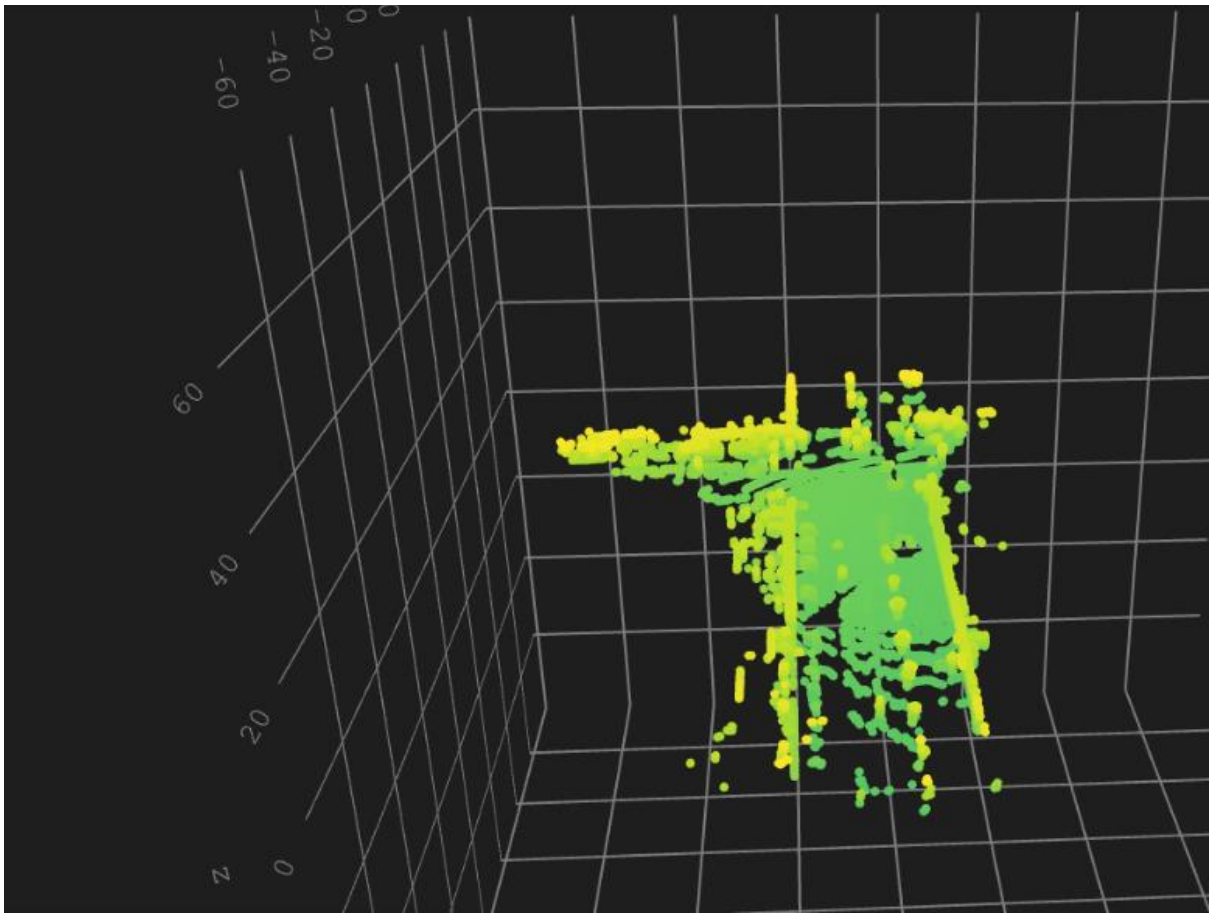


Figure 5 visualized point cloud

Figure 6 Cam2 image index 0

Part B: Probabilistic Occupancy Grid

1

C.

For the purpose of implementing a grid on a single scan, we employed a Probabilistic Occupancy Grid algorithm, as learned in class. Firstly, we excluded all the points within a radius of 2.5m of the car itself. To determine the drivable path, a naïve road segmentation approach was employed whereby all Z point values below 0.3m, after adding the Velodyne LiDAR sensor height of 1.73m (as per figure 7 from the kitti cite), were deemed as road points. Next, we utilized the defined parameters from the assignment definition table to create a grid map following the formulas: R = (x 2 + y 2)/ALPHA and $\emptyset$ = ($arctag2\ y\ x$ + π)/BETA. Each entry in the grid map corresponded to a specific R and $\emptyset$ in the LiDAR surrounding. We initialized all the cells to have 0.5 value, indicating an unknown status with respect to road or non-road. Where there was an occupied cell, i.e., a LiDAR point, we changed its value to 0.8 to indicate that it was occupied. For each angle, we tested if there was an occupied cell within its radius, and if found, marked all the cells that were prior to the first known occupied location as free. Conversely, when no occupied cell was detected for an angle, all the cells (radiuses) for that angle were marked as free. We then utilized the cv2.warpPolar to convert from Polar coordinate system to cartesian coordinate system and rotated and flipped the map to align the axis with the image for a better comparison between the two. The resulting output comprised of the filtered point cloud, scan grid in probability, and logit

representations, as depicted figure 8.

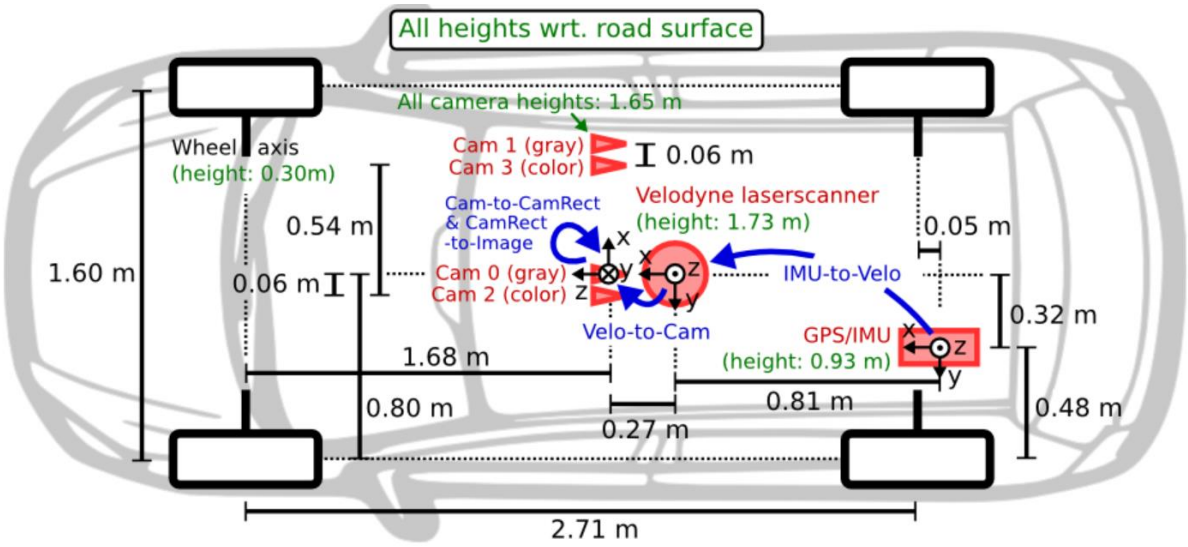


Figure 7 Sensor Setup

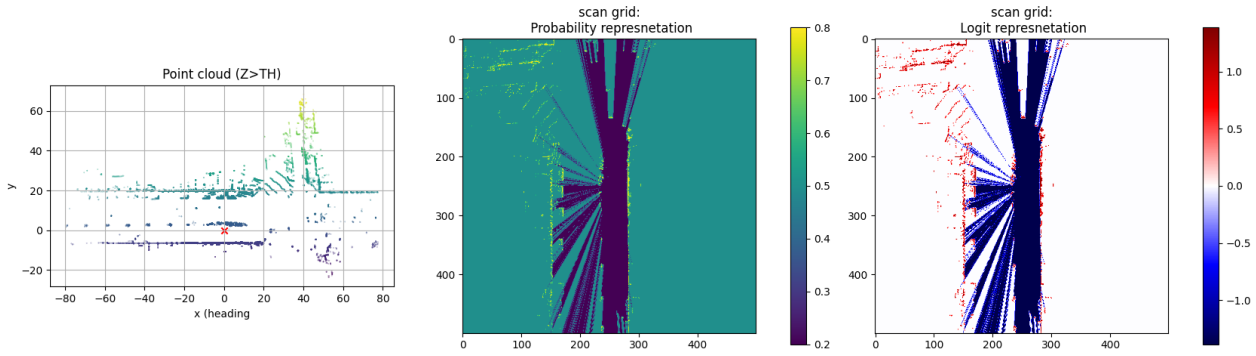


Figure 8 Scan Grids

D+E.

The current task requires updating and shifting the Occupancy Grid Map (OGM) to align with the next pose of the car. The objective is to incorporate the previous data collected while maintaining the map's ability to update itself with current readings. Figures 9 and 10 illustrate the OGM shift and update process. Initially, the previous OGM is shifted to match the coordinate system of the current OGM using the car shift between the two frames. Next, we applied a logarithmic transformation to both the shifted and the current OGMs, followed by summing their values. The clipped values, ranging from 0.02 to 0.98, were

incorporated to prevent an excessive influence of cells when combining multiple observations. Finally, the summed log values were converted back to the probability representation. This process enables a better understanding of the surrounding environment by fusing the data collected from previous and current readings.

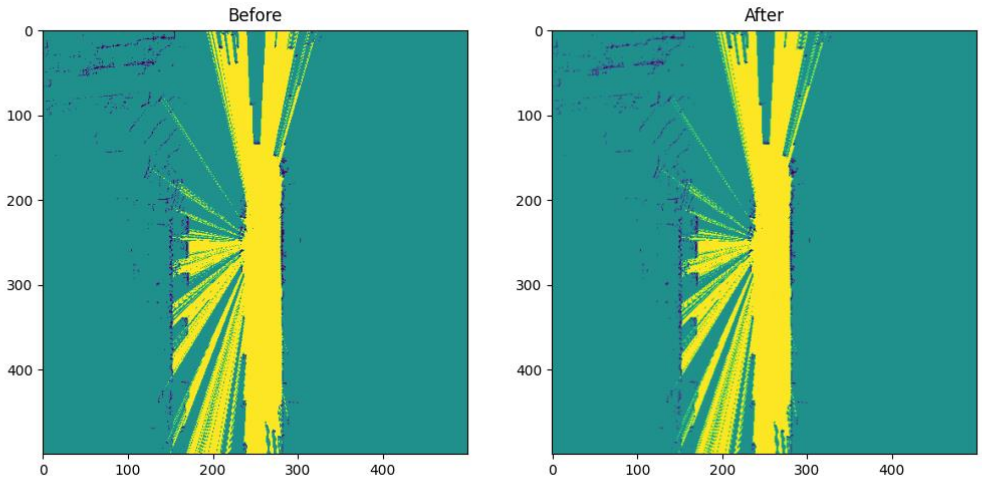


Figure 9 Shifted OGM

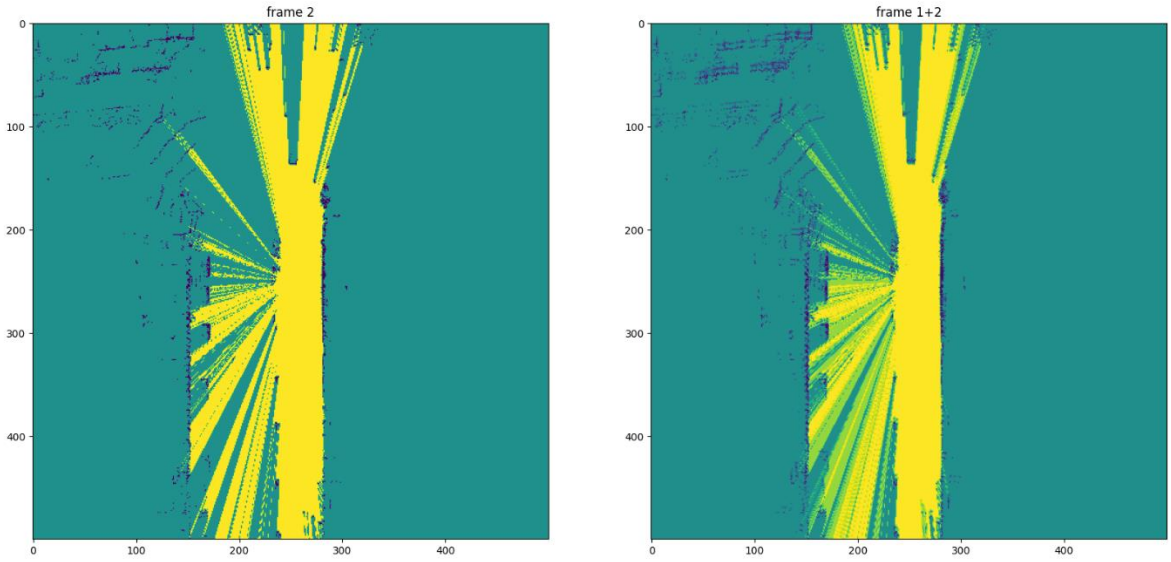


Figure 10 Updated OGM

2.

A-E

In the present study, we implemented a Probability Occupancy Grid over all the recorded LiDAR data. The pre-processing step involved transforming the raw LiDAR point cloud to INS location and aligning it based on the IMU inputs. Subsequently, we built an OGM per LiDAR scan using the function described in section 1. This process was repeated for all the scans/frames in the dataset. An animation was created with subplots displaying the OGM for each frame. The code and animation have been included in the project submission. The work process can be summarized as follows:

I. Loading the raw image and LiDAR readings for each frame in the dataset.
II. Filtering the raw point cloud to isolate the road and obstacles.
III. Transforming the point cloud to be on the same coordinate system as the IMU.
IV. Applying inverse of the roll and pitch on the point cloud.
V. Obtaining the pose of the current frame, shifting the previous OGM to be aligned with the OGM of the current frame, creating the OGM of the current readings, and updating the OGM accordingly.

F.

Attached below is the finale frame of the last frame of the occupancy map.

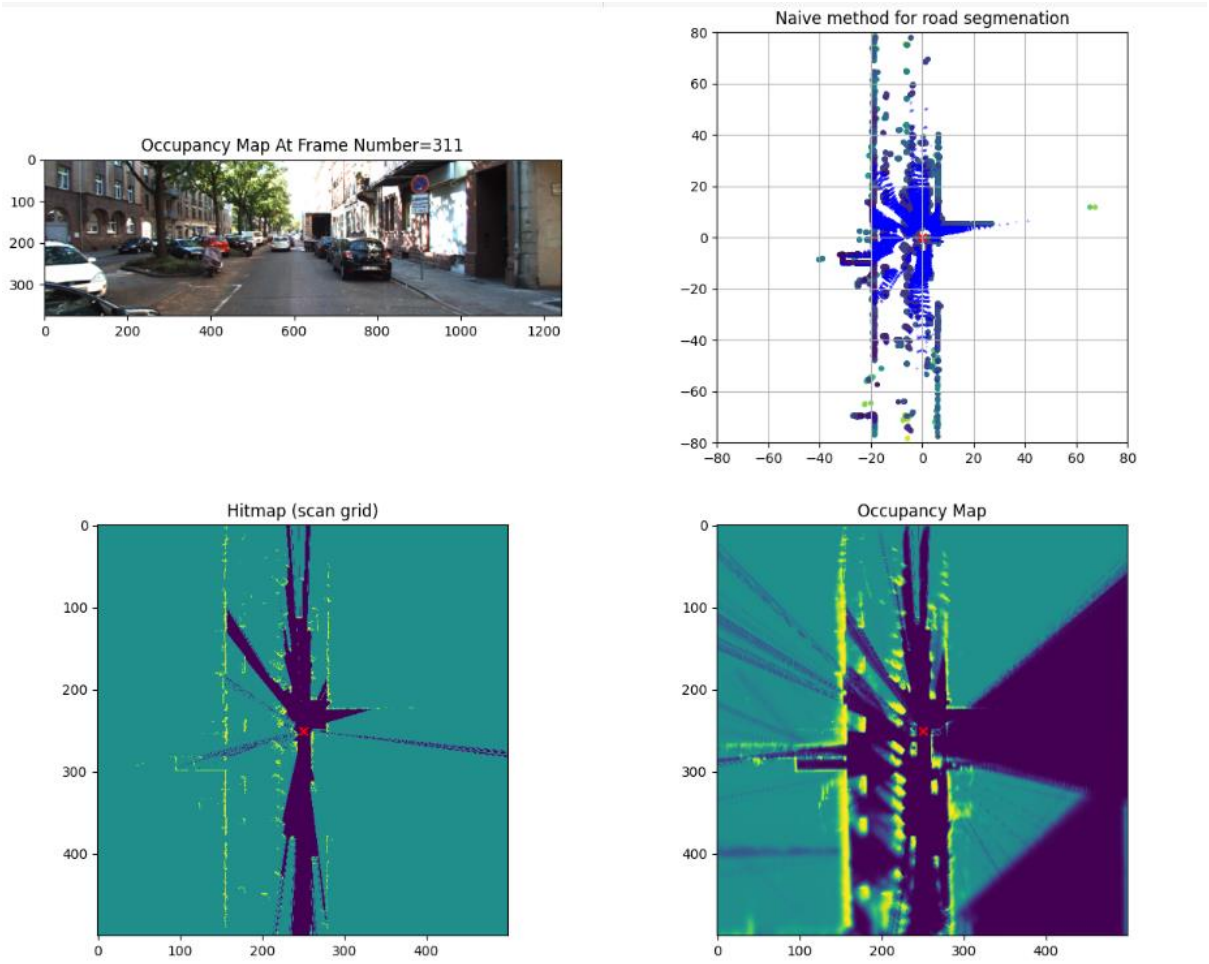


Figure 11 finale map frame

G.

The following image is the OGM with threshold.

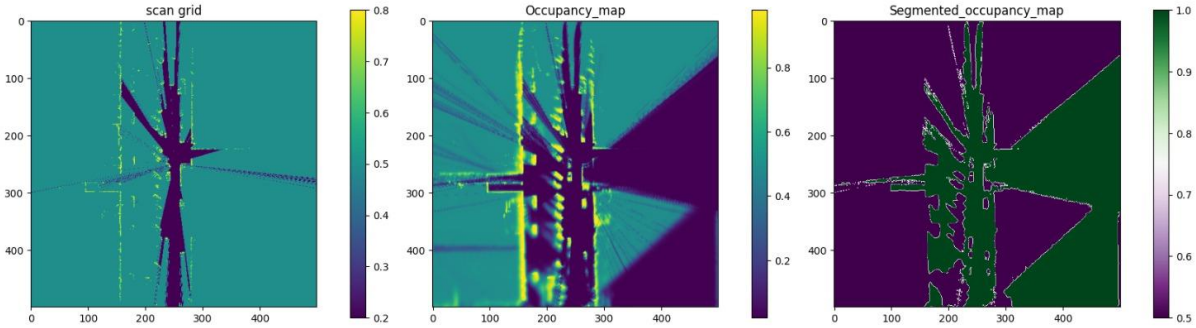


Figure 12 segmented OGM

3.

A.

In order to assess the performance of the occupancy grid map algorithm, examples of both good and bad mapping results were analyzed.

The following 2 frames are a representation of 2 good frames:

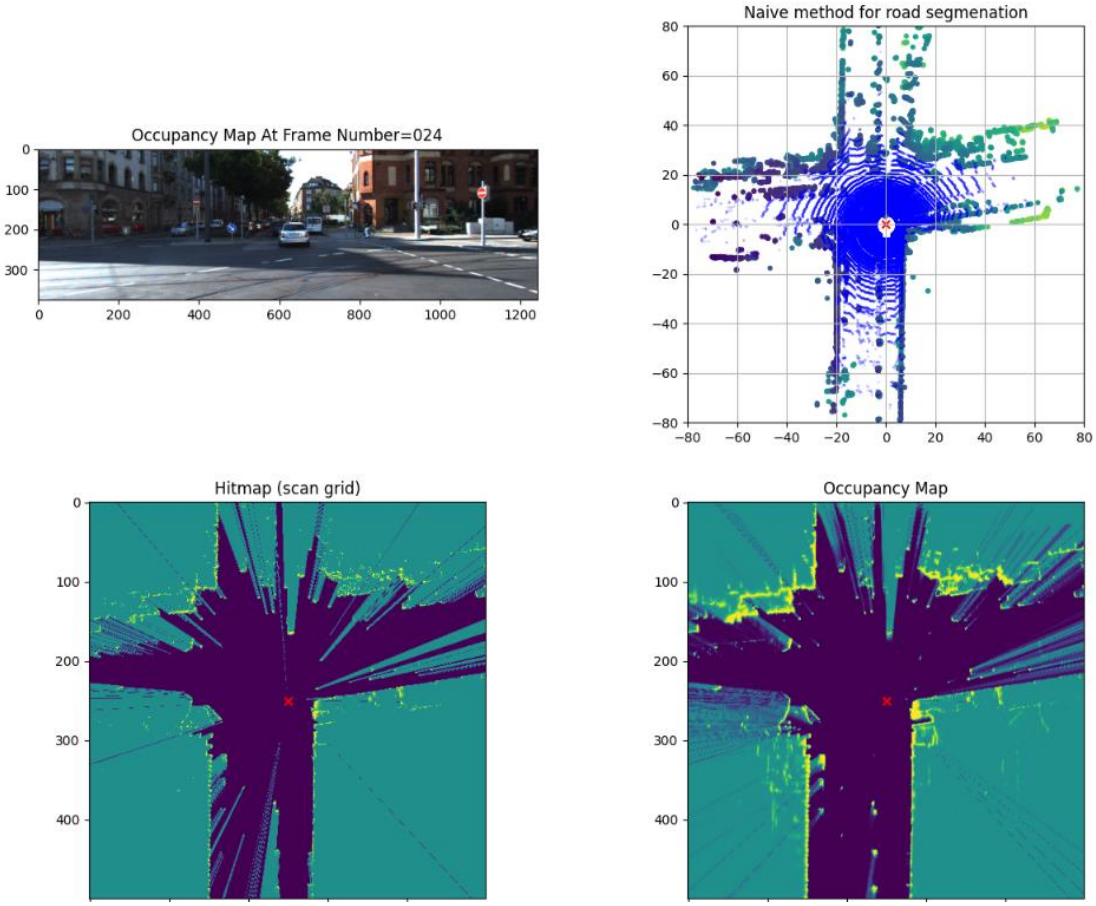


Figure 13 1st good frame

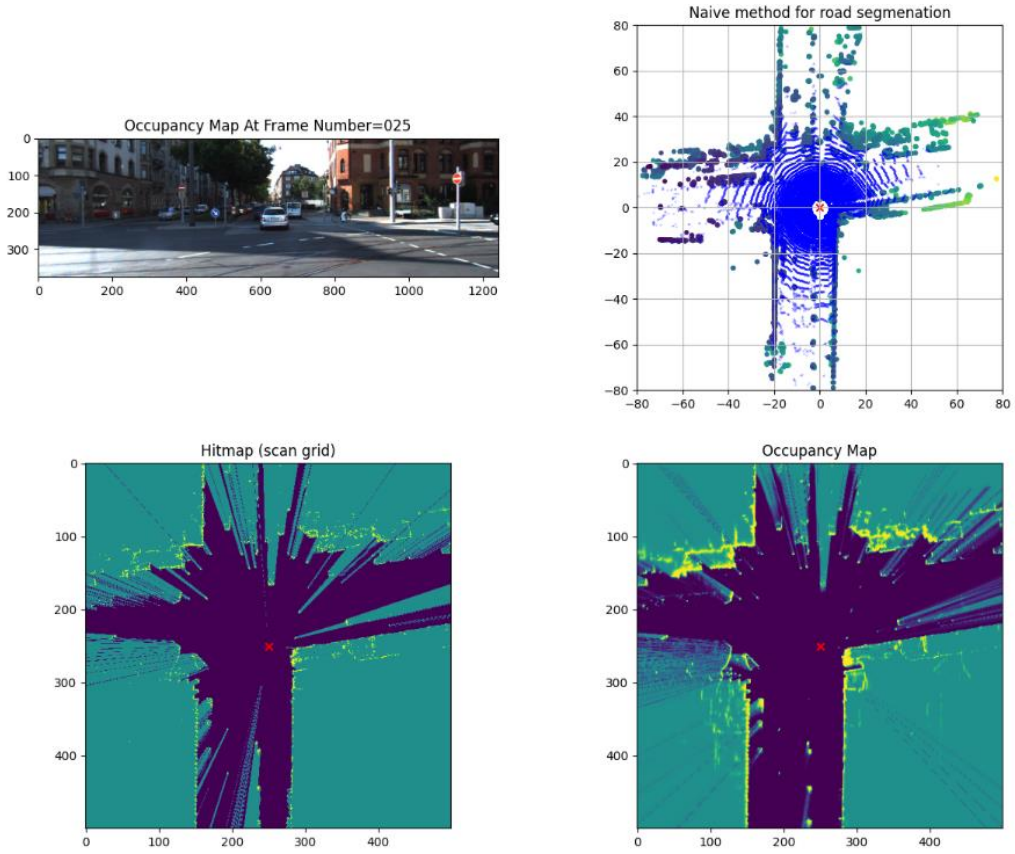
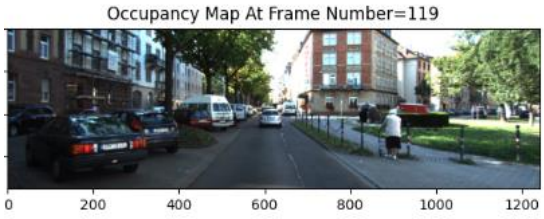
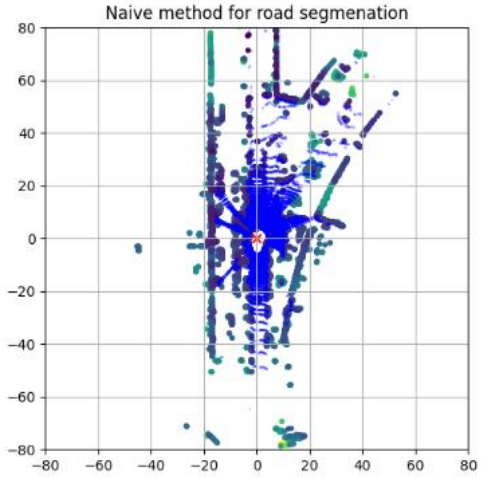


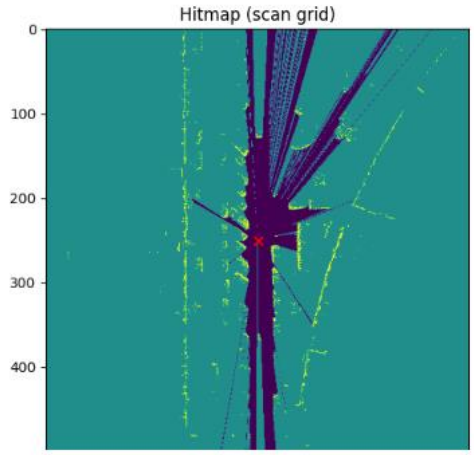
Figure 14 2nd good frame

The good examples were chosen since the surrounding area was predominantly open, which allowed the LiDAR sensor to accurately capture and represent the road surface.

The following 2 frames are a representation of 2 bad frames:







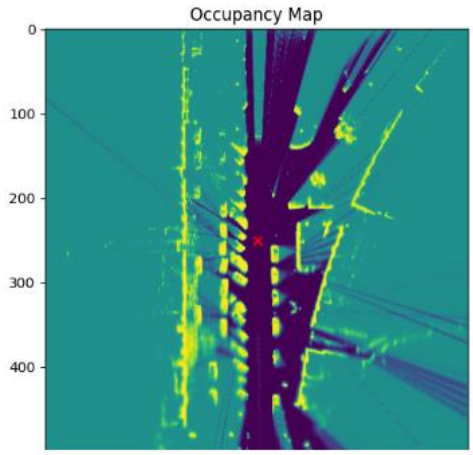


Figure 15 1st bad frame

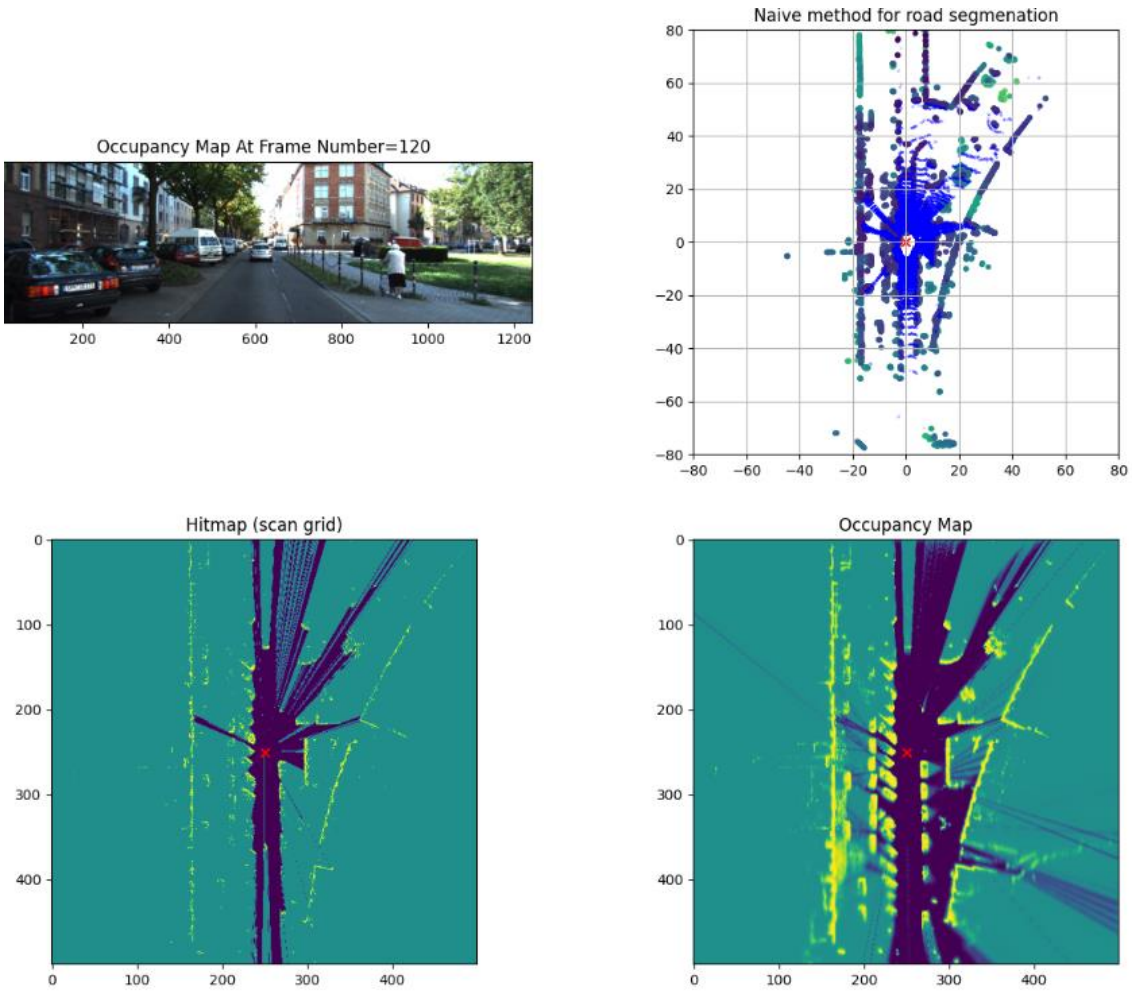


Figure 16 2nd bad frame

The two bad examples, on the other hand, were characterized by the presence of a nearby sidewalk, which the naïve segmentation algorithm mislabeled as part of the road. This error has significant implications, as the use of such inaccurate data could lead to erroneous car decision making, such as driving on the sidewalk.

B.

I. In order to investigate the effect of parameter adjustment on the model, the probability saturation values were altered from the default settings of 0.02 and 0.98 to 0.1 and 0.9, respectively. The anticipated outcome was an increased adaptability to changes in the environment and a more accurate representation of the surroundings within the map.

II. Upon analysis of the attached animation, it is evident that the modified parameter values resulted in a more dynamic and flexible mapping system. The updated map provides a more comprehensive and timely

depiction of the surrounding area, which is critical for applications such as autonomous vehicles or robotic navigation.

III. The new animation is added to this report.

C.

We will describe how objects of different types affect the mapping

I. Dynamic objects: Dynamic objects such as moving vehicles, pedestrians, or animals can affect the occupancy map by introducing sudden changes in the environment. In an OGM, these changes can lead to inconsistencies between consecutive frames, resulting in incorrect occupancy estimates. For example, if a vehicle suddenly appears in the field of view of the Lidar sensor, it can cause a sudden increase in the occupancy value, which may not accurately represent the true occupancy of the area. To mitigate this effect we used a probabilistic approach to update the OGM. In our dataset we don't have a lot of dynamic objects, we do have a car that is driving in front of us for most of the video but since we are driving in the same lane that change in the map is mostly neglectable. In case that there were a car crossing the road we are on than it would have a grater affect in confusing our map generation algorithm

II. Parking cars/trees: Objects such as parked cars or trees can affect the occupancy map by blocking the view of the Lidar sensor. This can cause areas behind the objects to be incorrectly classified as unoccupied. In an OGM, this effect can accumulate over time, leading to significant errors in the occupancy map. For example, if a parked car blocks the view of the Lidar sensor, the algorithm may classify the area behind the car as unoccupied, even if there are obstacles or other objects in that area. To mitigate this effect, we can use multiple sensors or incorporate contextual information to infer the occupancy of obstructed areas.
I our video we can see many parked cars that are considered to be obstacles on the road

III. Sparse/dense objects: Objects that are sparsely distributed or densely packed can also affect the occupancy map. Sparse objects may be missed by the Lidar sensor, leading to gaps in the occupancy map. On the other hand, dense objects may cause multiple reflections and scattering of the

Lidar beam, leading to errors in the depth estimation and occupancy classification. In an OGM, these effects can lead to incomplete or inaccurate occupancy estimates. For example, a sparse object such as a traffic cone may not be detected by the Lidar sensor, leading to an incorrect classification of the area as unoccupied.

IV. Transparent materials (such as glass): Transparent materials can affect the occupancy map by reflecting and refracting the Lidar beam. This can cause areas behind the material to be incorrectly classified as occupied or unoccupied, depending on the reflection properties of the material. In an OGM, this effect can lead to inconsistencies between consecutive frames and inaccurate occupancy estimates. For example, a glass window may reflect the Lidar beam back towards the sensor, causing the algorithm to classify the area behind the window as occupied. To mitigate this effect, some OGM algorithms use sensor fusion or incorporate contextual information to infer the occupancy of transparent materials.

V. Different color objects of the same type: Objects of the same type but different colors can affect the occupancy map by reflecting the Lidar beam differently. This can lead to variations in the intensity and depth of the data, which can affect the accuracy of the occupancy estimates. In an OGM, this effect can lead to inconsistencies between consecutive frames and incorrect occupancy estimates. For example, a black car may absorb more Lidar energy than a white car, leading to different depth readings and occupancy values. In our case we do not use the intensity readings at all so as for the lidar readings it does not affect us

Part C: Sensor fusion and semantic segmentation

1.

A.

To project LiDAR point clouds onto the image plane, the following workflow was adopted:

I. The calibration data was loaded from the KITTI dataset to obtain the lidar2cam_extrinsic matrix, as described in this github page.

II. Points with height more than that of the sensor (z=0) were filtered out.
III. The filtered point cloud was then multiplied with the extrinsic matrix to achieve a coordinate transformation.
IV. Points located behind the camera's field of view were removed by filtering out points with negative z values, where z after the coordinate transformation corresponds to depth, ensuring that only points with z>0 were retained.
V. Finally, two masks were created to filter out points outside the image frame, i.e., points with pixel location greater than zero in both axes but smaller than the actual image height and width. The resulting output for the first two readings of the dataset can be seen in the accompanying images.

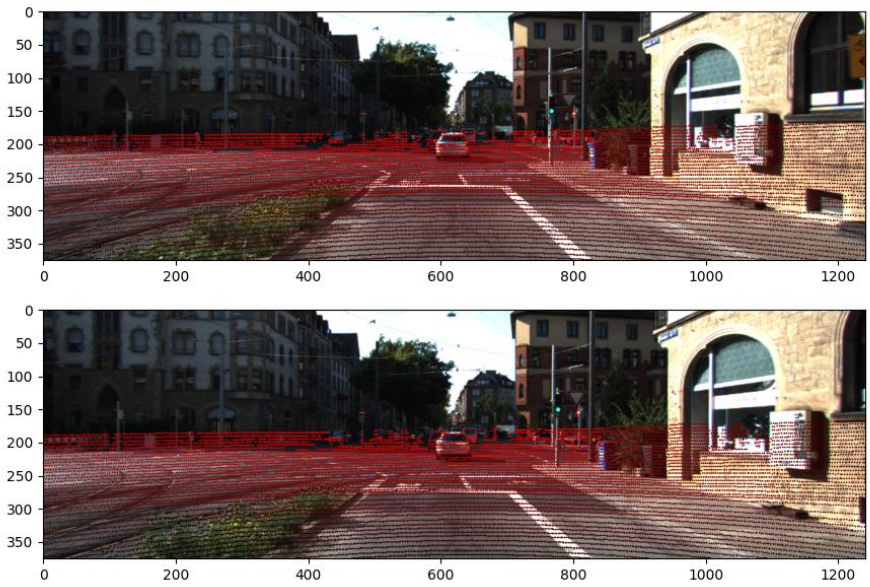


Figure 17 projected point cloud.

2. Road segmentation

A+B+C.

The process of using the DeepLabV3+ on my images is as follows:

I. First we crop the images to be in the desired ratio that model was trained on. For this we first convert the image ratio to be 4:3 (W:H) as can be seen in Figure 18.
II. After this we resize the image to be in the desired size that the model was trained on which is 513x513

III. We load the pretrained model and run it on the rescaled images and get as an output the probability of each pixel to belong to the road and we resize this probability map to the input image size

IV. From the probabilities we create a binary mask so that each pixel with probability larger than the threshold (that was set to 0.5) probability is considered as road and the rest are non-road

V. We use the cv2.connectedComponents function to identify regions in the predicted binary mask that are separated from the main road. It assigns a unique label to each connected component and returns the labeled image and the number of connected components.

VI. We calculate the mode of the labeled image, which corresponds to the label of the main road. It sets all pixels in the predicted binary mask that do not belong to the main road label to 0.

VII. We return the probability map and the predicted binary mask and use them to segment the lidar.

VIII. Finally, we use the cast lidar 2d point to assign each lidar point as road or not and visualize it on the image (as seen in the Figure 19) and apply the same binary on the cropped image (as seen in the Figure 20)

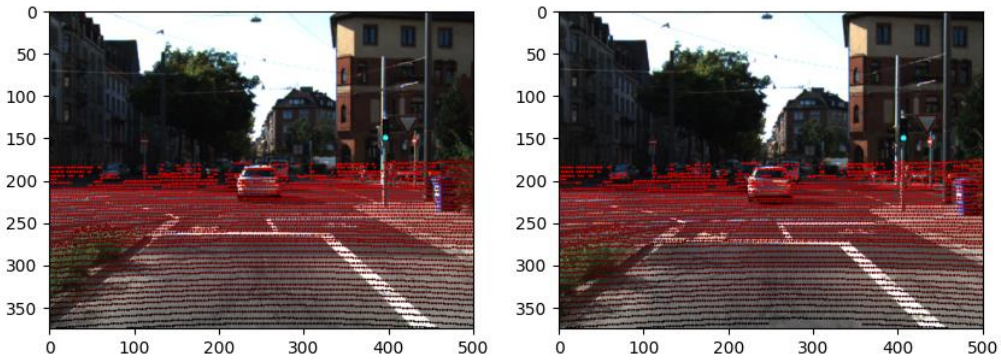


Figure 18 cropped and projected road

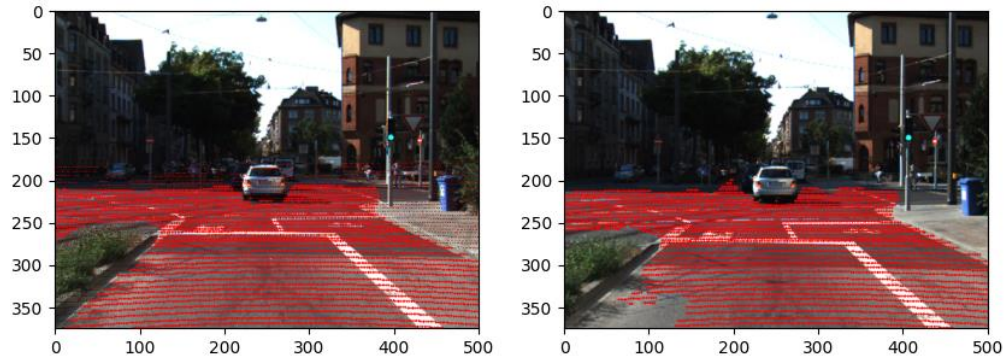


Figure 19 road segmentation

Figure 20 road segmentation just image

3. LiDAR Road Filter

A+B

In this section we used the Random Sample Consensus (RANSAC) algorithm which is a robust statistical method used for fitting models to data that may contain outliers.in order to remove outliers from our road prediction the. RANASAC algorithm works as follows:

I. Randomly select a minimum number of data points required to fit the model. These data points are called the inliers.
II. Fit a model to the inliers using a model estimation algorithm.
III. Test the fitted model against the remaining data points. Calculate the residuals, which are the differences between the actual data points and the model predictions.
IV. Count the number of data points whose residuals fall within a predefined threshold (inlier threshold). These data points are considered as inliers, while the rest are outliers.
V. Repeat steps 1-4 for a fixed number of iterations or until a desired level of confidence is achieved.
VI. Select the model that has the largest number of inliers.

After creating the model using the RANSAC algorithm we used this model to filter the road points and got a better model of the road as can be seen in figure 21 for the first 2 readings

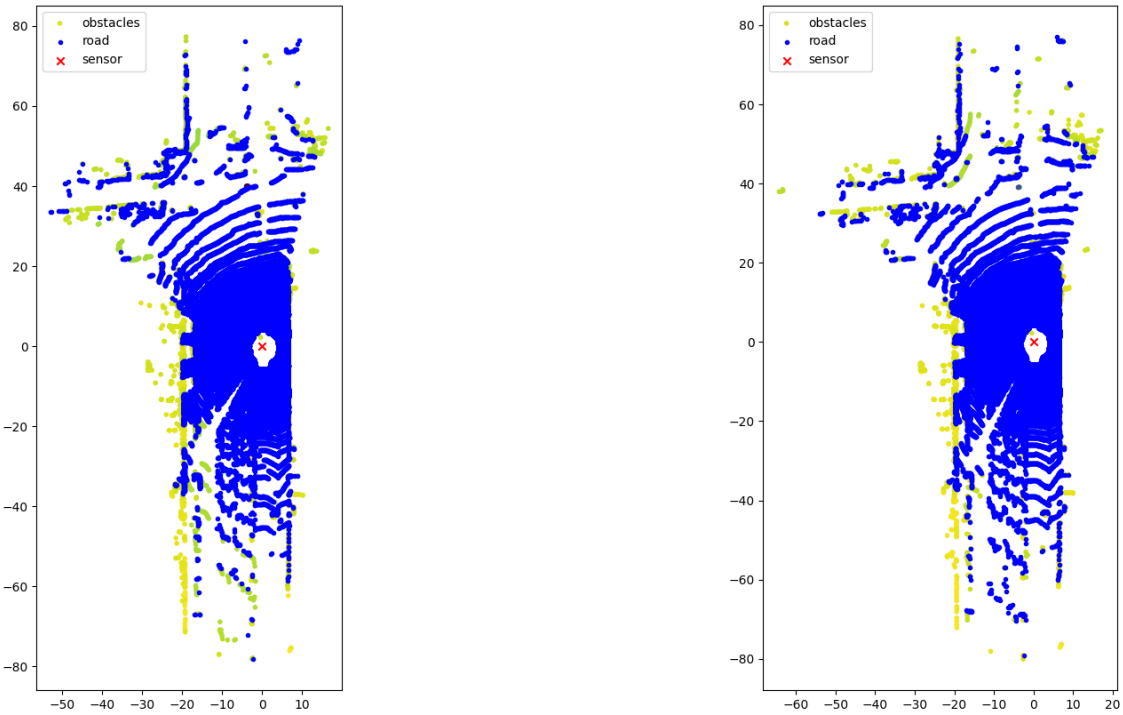


Figure 21 point cloud filtered.

4.Implementation of a probability occupancy grid based on deep learning!

A-D.

Now we will Repeat the occupancy map process based on deep learning over all your

scans/frames and create the animation that is added to this report.
the flow of the code is as follows:

I. We load the pre trained Deep_Labv3 model.
II. For each frame we load the raw image and LiDAR readings
III. We filter the LiDAR points below the sensor.
IV. Transform the LiDAR to the camera coordinates.
V. project the point cloud from camera coordinates to image coordinates.
VI. Crop the image so it would fit the desired input size for the Deep_Labv3 model and run it through the model.
VII. Use the get_road_model_ransac def to create a road model and filter the LiDAR point using this model.
VIII. Transform the point cloud to be on the same coordinate system as the IMU.
IX. Finally, advance the former OGM to be on the same map as the current readings, generate OGM for the current reading and update the map accordingly.

4.Analyze your results.

A.

When comparing the two road mapping results (the naïve way and the deep learning model) we can see that the results we receive are very similar even though the deep learning results provide a mor accurate way to perform the road segmentation. In my opinion the reason for this similarity is that the given dataset we received is quite simple in terms of mapping, we only drove straight throughout the entire test drive and the road we drove on was horizontal all the way and without any holes in it.

B.

Example of 2 good mappings:

As can be seen in figures 22,23 we have an example of 2 goop mappings, the algorithm mange to segment the road from the sidewalk and tell apart between the road and the cars that parks on it.

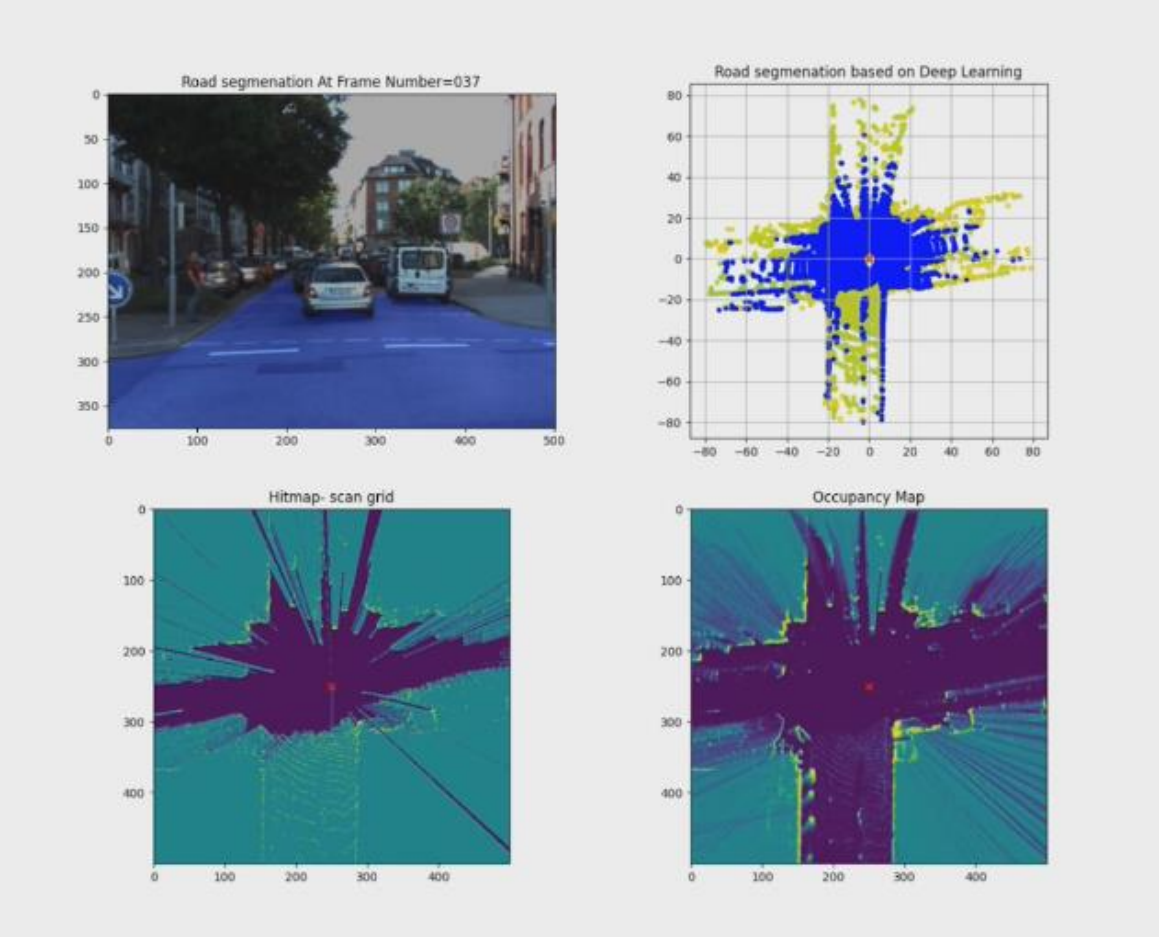


Figure 22 1st good mapping example

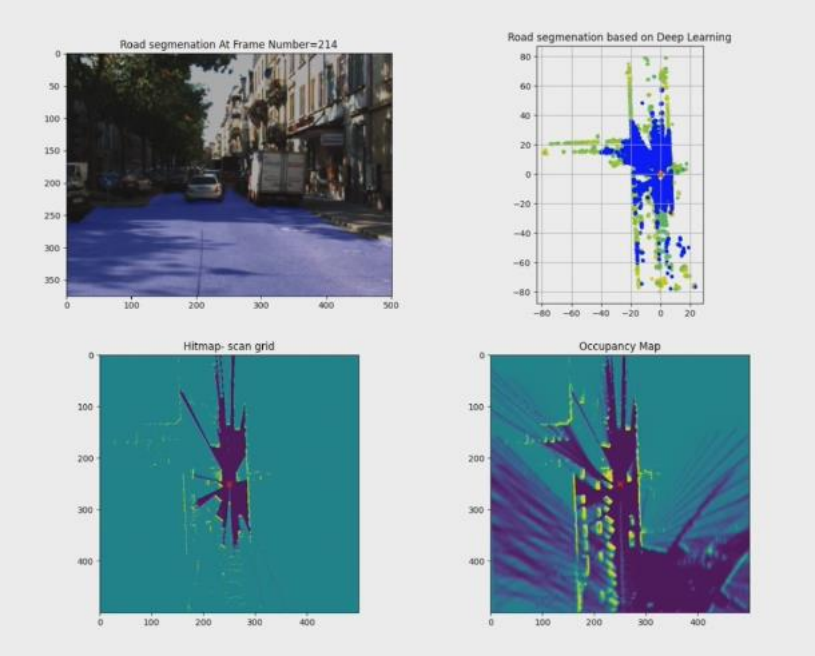


Figure 23 2nd good mapping

Example of 2 bad mappings:

In figures 24,25 we can see an example of bad mapping. For some reason whenever the mapping algorithm passes through a track it appears as an opening in the road, meaning that the algorithm confuses an obstacle for a road. This can be seen from the LiDAR reading that does not show an opening to right while the OGM considers it to be a road opening.

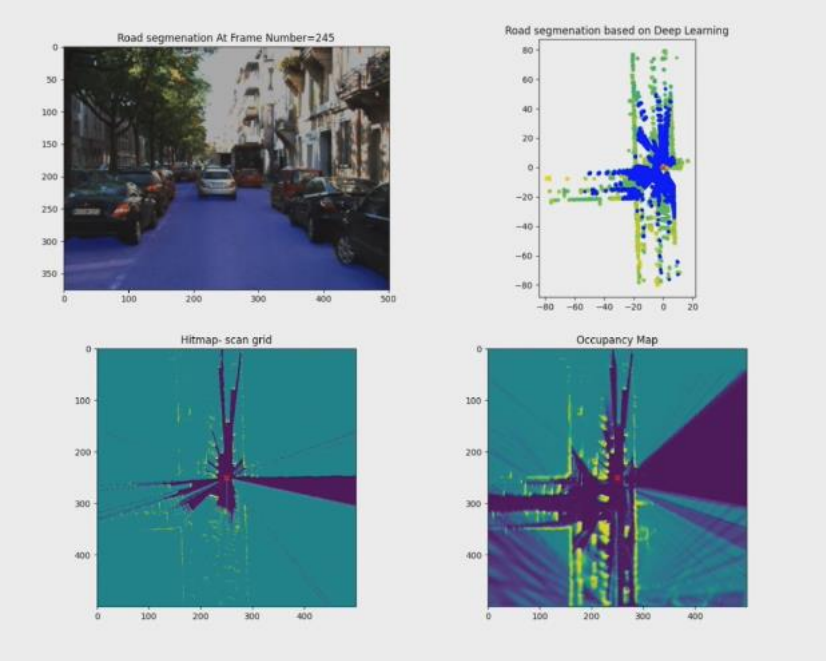


Figure 24 1st bad frame

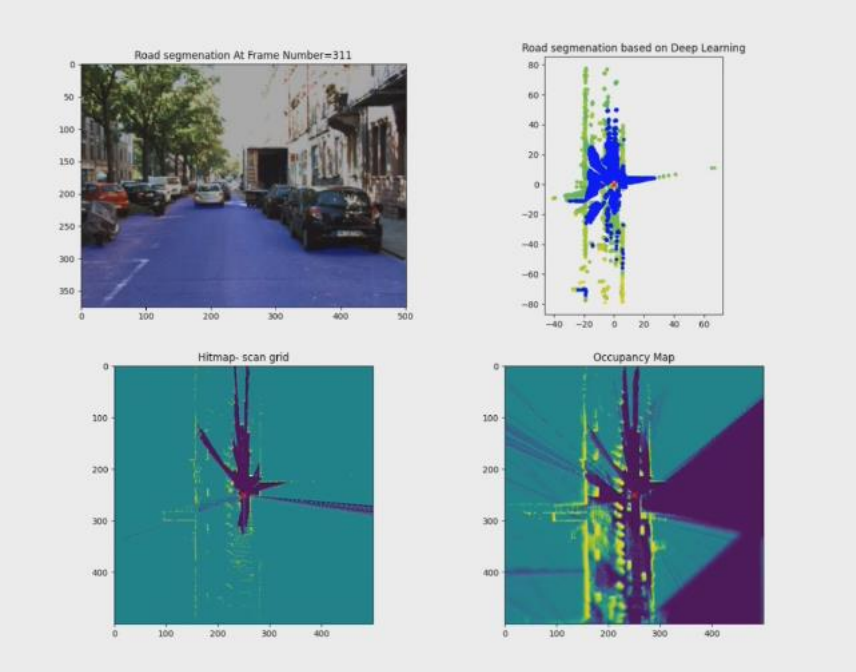


Figure 25 2nd bad frame

C.

Mapping based on deep learning has several advantages and disadvantages compared to the naïve method. Three pros of using deep learning for mapping are:

I. Superior performance in distinguishing between road and sidewalk due to the ability of deep learning models to learn complex features and patterns from large amounts of data.
II. Deep learning models are less prone to sensor noise and can utilize advanced sensor denoising techniques such as RANSAC, leading to improved accuracy in mapping.
III. Deep learning models are highly robust to variations in the input data, enabling them to identify features such as holes in the road that may be overlooked by the naïve method.

However, there are also several cons to using deep learning for mapping, including:

I. Deep learning models may be computationally expensive and may not be suitable for real-time systems that require fast processing times such autonomous vehicles.
II. Deep learning models can be more complex and difficult to track compared to the naïve method, as it can be challenging to understand how the model arrived at a given output.and thus harder to debug
III. There is a need for a large labeled dataset in order to get a good NN model while the naïve doesn't need any prior dataset.

5.Semantic segmentation/DeepLabV3+:

A.

Atrous convolution, also known as dilated convolution, is a powerful technique used in semantic segmentation and image analysis tasks. Here are some of the main advantages of atrous convolution:

Increased receptive field: Atrous convolution allows us to increase the size of the receptive field without increasing the number of parameters. By adjusting

the dilation rate, we can control the size of the effective kernel size, which enables us to capture larger context information in the input image.

Fine-grained feature extraction: Atrous convolution enables us to extract features at different scales. By applying atrous convolution with different dilation rates, we can extract fine-grained features that capture local details as well as coarse features that capture global context.

Reduced computation cost: Atrous convolution reduces the computational cost of feature extraction by avoiding the need for upsampling or pooling operations, which can lead to loss of information. By using atrous convolution, we can capture multi-scale features efficiently without sacrificing the resolution of the feature maps.

Overall, atrous convolution is a powerful technique that enables us to extract multi-scale features efficiently and effectively, making it a valuable tool for tasks such as semantic segmentation and image analysis.

B.

Jaccard/Dice coefficient is a commonly used metric for evaluating the performance of semantic segmentation models. It measures the similarity between the predicted segmentation mask and the ground truth mask by computing the ratio of the intersection over the union of the two masks. While the Jaccard/Dice coefficient is a useful metric for evaluation, it is not an ideal choice for use as a loss function during training. Here are some reasons why:

Non-differentiable: The Jaccard/Dice coefficient is a non-differentiable metric, which means that we cannot compute gradients directly with respect to it. This makes it difficult to use as a loss function during training with gradient descent-based optimization algorithms.

Unbounded: The Jaccard/Dice coefficient is an unbounded metric, which means that it can take values from 0 to infinity. This can make it challenging to compare the performance of different models, as a model that achieves a higher Jaccard/Dice coefficient may not necessarily be the best performing model in practice.

Unstable gradients: During training, the Jaccard/Dice coefficient can lead to unstable gradients, especially when the predicted mask and ground truth mask have small or no overlap. This can make it challenging to train the model effectively, leading to slow convergence or even divergence.

For these reasons, it is common to use other loss functions such as cross-entropy or focal loss, which are differentiable and have more stable gradients. These loss functions can be optimized efficiently using gradient descent-based algorithms and can lead to more stable and effective training of semantic segmentation models.

C.

Road segmentation for offroad is a challenging task due to the following reasons:

I. Heterogeneous and complex terrain: Offroad areas are often characterized by complex terrain and a heterogeneous mix of surfaces such as dirt, rocks, vegetation, and water. This makes it difficult to accurately detect and segment roads, as the appearance and texture of the road surface can vary significantly across different regions.
II. Limited availability of training data: Obtaining labeled data for offroad areas can be challenging, as it requires extensive manual annotation of images or lidar data. This is due to the high variability of offroad terrain, which can make it difficult to capture representative samples of the different road types and conditions.
III. Adverse weather and lighting conditions: Offroad areas are often subject to adverse weather and lighting conditions such as rain, fog, and shadows. This can affect the visibility of the road surface and make it challenging to accurately detect and segment the road. Additionally, offroad areas may not have consistent lighting conditions, which can make it difficult to train models that are robust to changes in lighting conditions.
IV. Presence of obstacles and occlusions: Offroad areas may contain various obstacles such as trees, boulders, and other vehicles that can occlude the road surface. This can make it challenging to accurately segment the road, as the model needs to be able to distinguish between the road and other objects in the scene.
V. Limited computational resources: Offroad environments often have limited computational resources due to the remote nature of the terrain. This can limit the use of advanced segmentation models that require significant computational resources, making it challenging to develop

accurate and efficient road segmentation models for offroad environments.

SUMMARY

Upon completing this project, I gained practical experience with implementing the robot mapping algorithm, which I learned in class, using recorded data from the KITTI dataset. The project covered three main topics: Geodetic coordinate systems and the KITTI dataset, Probabilistic Occupancy Grids, and Sensor fusion and semantic segmentation.

In the Geodetic coordinate system section, I downloaded my recording data from the KITTI dataset and described technical details about my scenario, such as the driving route, presence of dynamic objects, occlusions, and turns. I plotted the trajectory of the recorded data using Google Maps and displayed it in ENU and NED representations of local coordinates, along with the right side angular velocity, pitch, roll, and yaw. I also displayed the quality of GPS signal reception and provided an example each of an image (camera) and point cloud (LiDAR) from my record.

In the Probabilistic Occupancy Grid section, I implemented a grid on a single scan using a naïve rule to filter all the point clouds above ~30 cm off the ground and as non-road and created an Occupancy Grid Map (OGM). I added figures of the first scan, including segmented point cloud, scan grid in probability and logit representations, and described my implementations in detail. I also updated and shifted the Occupancy Grid Map (OGM) according to the next pose of the car and showed an example of the first two frames. Lastly, I created an animation that includes subplots showing the OGM per LiDAR scan and described my work process.

For semantic segmentation, I used the DeepLab v3+ model. This model is capable of accurately labelling each pixel in an image with a corresponding

object class road/ non-road by using this pretrained model, I was able to save time and resources compared to training a segmentation model from scratch.

For sensor fusion, I integrated the semantic segmentation results with the lidar and camera sensor data to create a more comprehensive understanding of the surrounding environment. By fusing these different sensor modalities along with the NN's output and applying the RANSAC algorithm, I was able to improve the accuracy and robustness of the perception system. And create a more accurate OGM.

Overall, I think I got good results and learned a lot about robot mapping and perception, including showing examples of good and bad mapping results and explaining how objects of different types affected the mapping. I also changed parameters of the model and described the changes, final results, and the effect of these changes on the mapping results, including providing examples and animations.