

**Project 1**

Mapping and perception for an autonomous robot/ 0510-7591

By

Nimrod Curtis

Tel Aviv University  
April 2023

**Contents**

[1. Introduction 2](#_Toc132584893)

[2. Solution 3](#_Toc132584894)

[A. Geodetic coordinate system and get familiar with the KITTI dataset 3](#_Toc132584895)

[1. Knowing the dataset 3](#_Toc132584896)

[B. Probabilistic Occupancy Grid – Naïve Role 6](#_Toc132584897)

[1. Grid on a single scan 6](#_Toc132584898)

[2. Implementation of a Probability Occupancy Grid over all the record 8](#_Toc132584899)

[3. Analyzation 9](#_Toc132584900)

[C. Sensor fusion and semantic segmentation 12](#_Toc132584901)

[1. Sensor fusion: Projection LiDAR2Image 12](#_Toc132584902)

[2. Road Segmentation 13](#_Toc132584903)

[3. LiDAR Road Filter 14](#_Toc132584904)

[4. Probability OGM based on DL: 14](#_Toc132584905)

[5. Analyzation 15](#_Toc132584906)

[6. Semantic segmentation/DeepLabV3+ 17](#_Toc132584907)

[3. Summary 18](#_Toc132584908)

# Introduction

This is the initial project for a mapping and sensing course on autonomous systems, which covers three primary subjects:

* Geodetic coordinate system and the KITTI dataset
* Probabilistic Occupancy Grid
* Sensor fusion and semantic segmentation

During the project we used the recorded data from KITTI dataset which is a collection of real-world data for computer vision and robotics research, captured from a car equipped with various sensors such as cameras and LIDAR.

Using this data, we applied algorithms that generate and continuously update probabilistic occupancy maps. We utilized both a naïve approach and a camera-LiDAR sensor fusion method based on deep learning techniques.

The record number: 2011\_09\_26\_drive\_0005

The recording features driving in an urban setting that involves navigating through a roundabout with dynamically moving vehicles ahead of the car.

# Solution

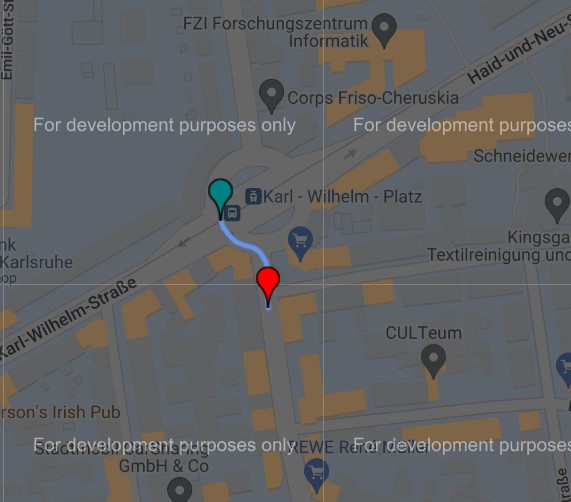
## Geodetic coordinate system and get familiar with the KITTI dataset

### Knowing the dataset :

1. Dataset technical details:

The information pertains to driving a vehicle in an urban area. Initially, the vehicle enters a roundabout and takes the first right exit after turning counterclockwise. While on the road, the vehicle follows a bicycle rider with a van ahead of them. Immediately after turning right, a camp van, which belongs to road workers, is seen on the travel lane. The vehicle passes the van and continues its route. The database reveals the presence of 9 vehicles, including 3 vans, 2 pedestrians, and a cyclist (stats obtained from the official website of the database). A video in the 'results' folder provides a visualization of the data from the lidar and left camera on the rviz platform (using 'kitti2bag' pkg).

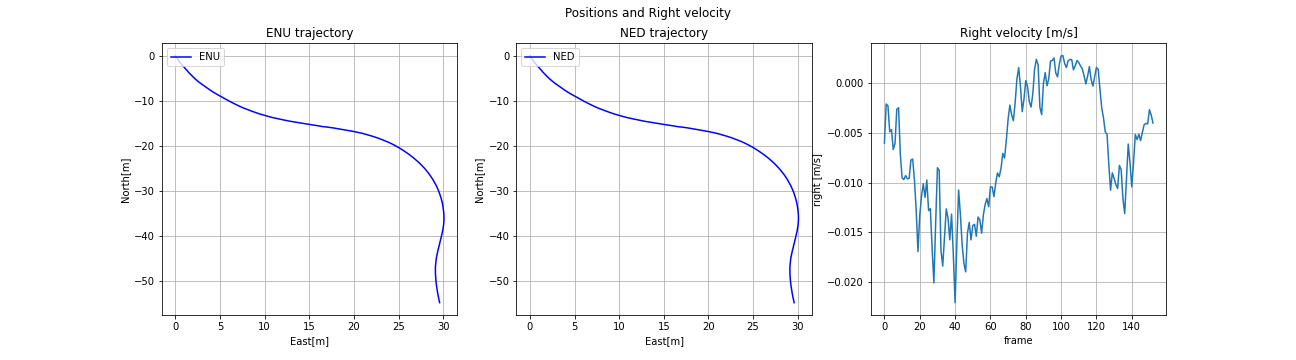
1. Trajectory on GoogleMaps:



The image above describes the driving route on a map from within Google, the green marker marks the beginning and the red marker the end of the route. It is important to note that the driving route takes place in the city of Karlsruhe in Germany.

1. Vehicle pose:

**Positions and right velocity:**



The plots above describe the location of the vehicle locally in relation to the ENU axis system. When turning around the square, the vehicle drives in a southeast direction and then turns right to the south. On the route in relation to NED, a transformation was carried out to display in relation to ENU in order to verify the correctness of the information and the conversion.

From the information of the right velocity we learn that indeed the vehicle is initially facing left because the speed to the right becomes more negative and this corresponds to the fact that the vehicle rotates counter-clockwise. And during the turn to the right, the speed increases and becomes positive until the end of the vehicle's overtaking of the van.

**Orienatation:**

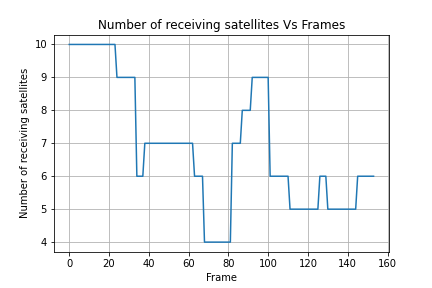
**תמונה שמכילה טבלה

התיאור נוצר באופן אוטומטי**

The graphs above describe the orientation of the vehicle while driving. From the roll and pitch information, it can be said quite clearly that the vehicle is driving on a plane where the average of the tilt angles is approximately 0-0.5 degrees and the standard deviation is also in the 0.5 degree area.

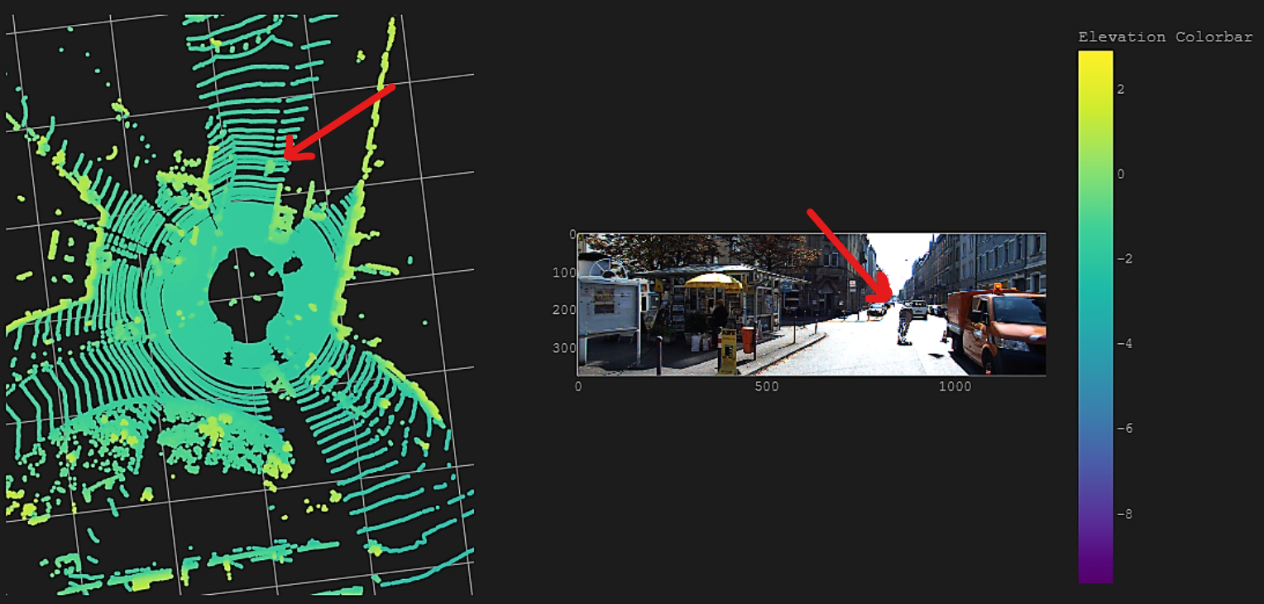
From the yaw angle it can be seen that the information describes the vehicle's driving style well. First the vehicle travels in a southeasterly direction, therefore the yaw is negative but changes towards 0 degrees and then the vehicle turns to the right and therefore descends in degrees until an absolute southern direction (-90 degrees).

1. Quality of satellites:

****

The graph describes the quality of the information from the GPS sensor by showing the number of satellites that took part in updating the information. In total, the number of satellites ranges from 4-10. According to the given graph, it can be assumed that the information at the beginning (first 30 frames) will be more accurate because it was taken from a larger number of satellites.

1. PCL and Image pair of frame #100:



The scene is displayed in both Lidar points cloud and camera image. The uniform turquoise color represents the lidar's rays reflecting from the road, which confirms the road's appearance in front of the vehicle, as seen in the image. Additionally, a red arrow marking the cyclist in front and to the right of the vehicle confirms that the scenes are the same.

## Probabilistic Occupancy Grid – Naïve Role

### Grid on a single scan

This section involves generating a scan grid through a basic filtering process applied to the LiDAR data. Using probabilities and the Bayesian approach, the occupancy map is updated as the system moves to the next frame.

* 1. Point Clouds filtering:

Points that were more than approximately **30cm above the ground** were segmented as non-road. The height of the LiDAR sensor mounted on the car was obtained from the official KITTI dataset website. With a **sensor height of 173cm**, the filtering **threshold used was -143cm** (since the LiDAR z-coordinate is positive upward) in relation to the LiDAR coordinate system.

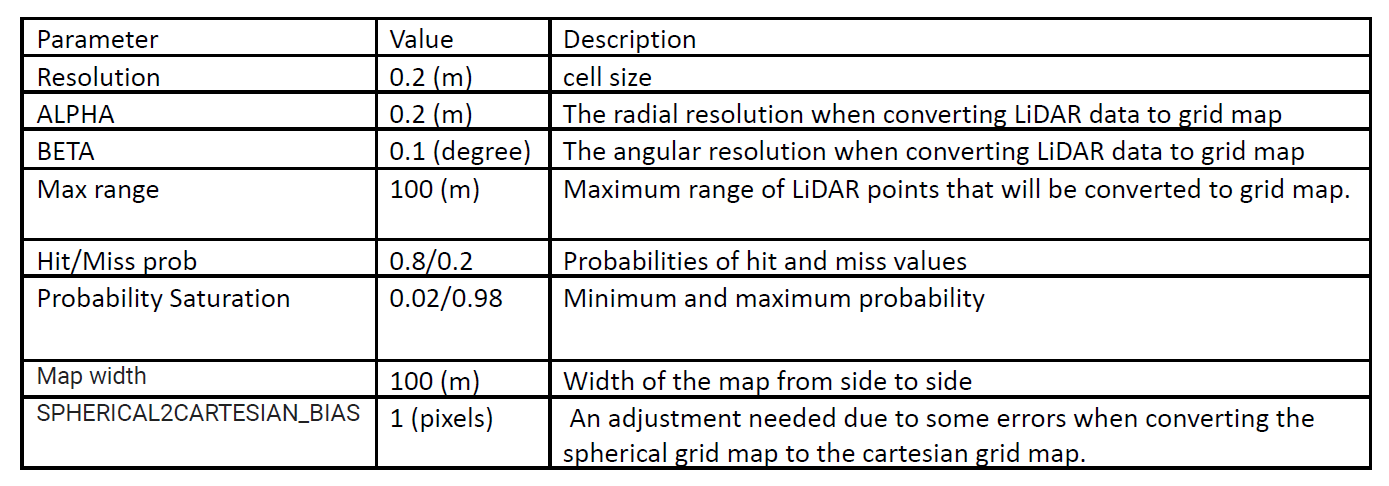
תמונה שמכילה טבלה

התיאור נוצר באופן אוטומטי

The figure displays the filtered raw LiDAR data. Points above 30cm off the ground are marked as non-road and displayed in various colors based on their elevation. Road points below the threshold are colored blue.

* 1. Recommended parameters

Recommended parameters for the scan-grid algorithm were taken from the project instructions.



* 1. Scan grid in probability and logit representations
     + To update the OGM (occupancy grid map) with the most recent measurement, the LiDAR points need to be converted to the similar grid format. This grid is called the scan grid (SG).
     + In this project, the SG created in spherical coordinate first.
     + Once the spherical scan grid is generated, it can be populated based on the LiDAR data and the parameters table provided, before being converted to Cartesian coordinates.
     + The OGM is initialized with probability of all its cells = 0.5.
     + Then a single scan grid can be generated.
     + The OGM is updated by the recent scan grid using Bayesian update, according to the formula:

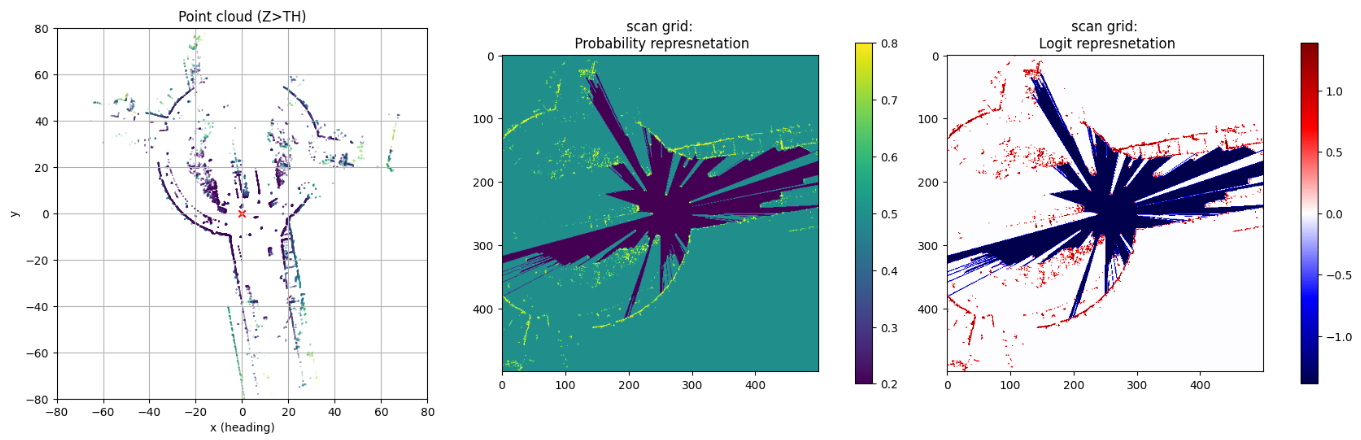
Where:

is the logit of the i -th cell of the updated OGM.

is the logit of the i -th cell of the previous OGM.

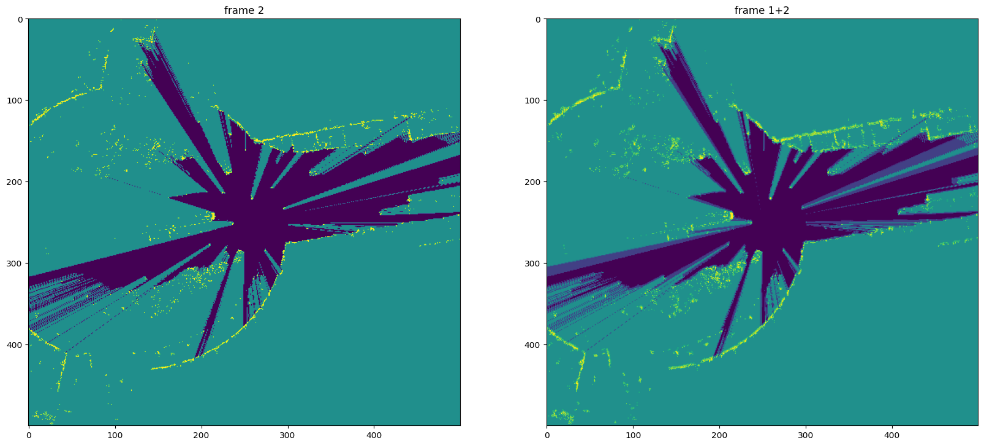
is the logit of the i -th cell of the scan grid that are generated from the latest LiDAR points.

* + - Then, the usable OGM can be found by calculating the inverse-logit of



* 1. Updating and shifting the OGM

The Occupancy Grid Map (OGM) is shifted using the car's odometry readings from the inertial system. This enables the OGM Bayesian updating process to function accurately. See the first two frames for an example:



* 1. Analyzation:

Analyzing the algorithm's effect is challenging with just two frames as an example. However, it is evident from the updated image (frame 1+2) that a range of colors shows the probability deterioration/strengthening in the map cells, as indicated by the red arrow. For example new areas are revealed, and updating the map results in cells previously marked as unknown having a decreased probability of being occupied, thus appearing blue.

### Implementation of a Probability Occupancy Grid over all the record

1. Pre-processing:

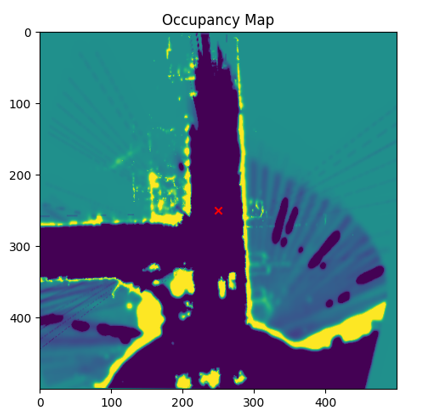
Transforming to the point clouds data was executed by the LiDAR-to-IMU calibration matrix which is the inverse homogeneous matrix of the IMU-to-LiDAR that already included in the dataset.

The ego-sphere aligining of the point clouds executed by multiplying the rotations matrices of the roll and pitch angles with the LiDAR points.

1. The OGM updating process per-scan was built.
2. The OGM updating process was implemented over all LiDAR scans within a loop cycle.
3. Animation of the mapping process with the recommended parameters was saved inside **"Results/figs\_b"** folder and under the name **"Animation\_default"**
4. Process flow:

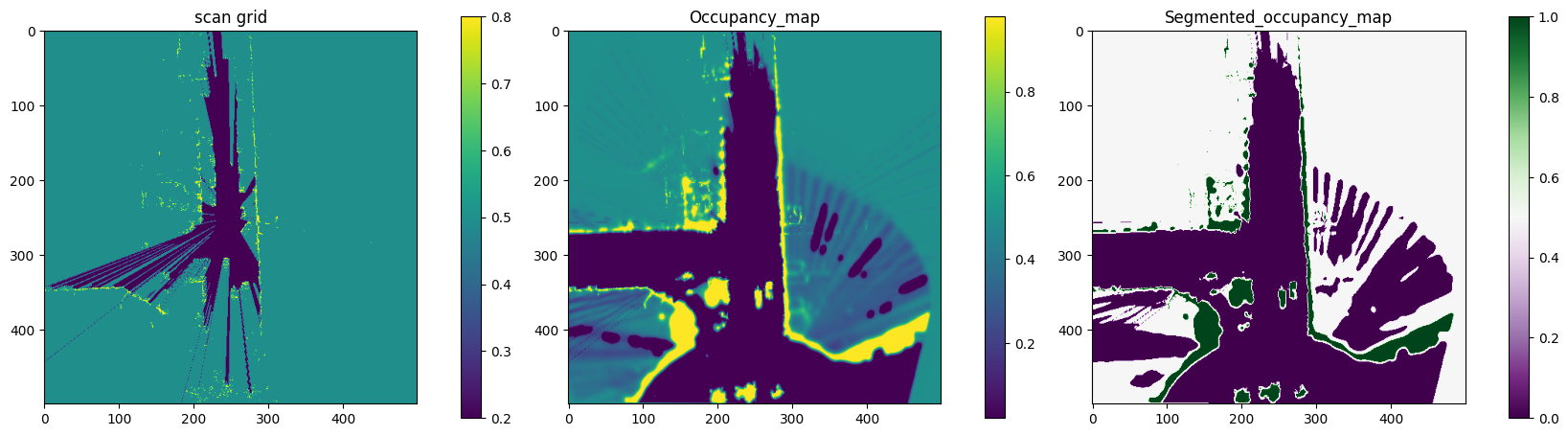
Initialization of occupancy map with unknown cells (0.5 prob) -> Open a for loop iterating over all the data files -> Load the LiDAR raw data -> Filter the LiDAR as described -> Project the points onto the INS frame as described. -> Align with ego-sphere -> Load car POSE from the INS -> Generate OGM as described -> Shift last OGM as described -> Update the OGM based on the shifted and current OGM.

1. Last frame OGM:

****

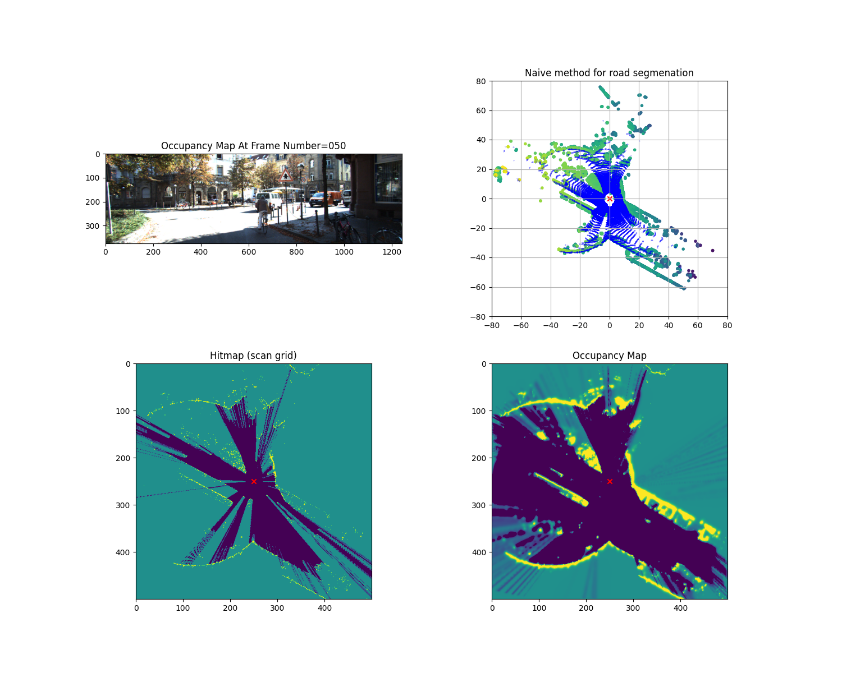
1. **Segmented map:**

Segmented map was generated using the thresholds 0.35 for free<->unknown and 0.8 for unknown<->occupied.



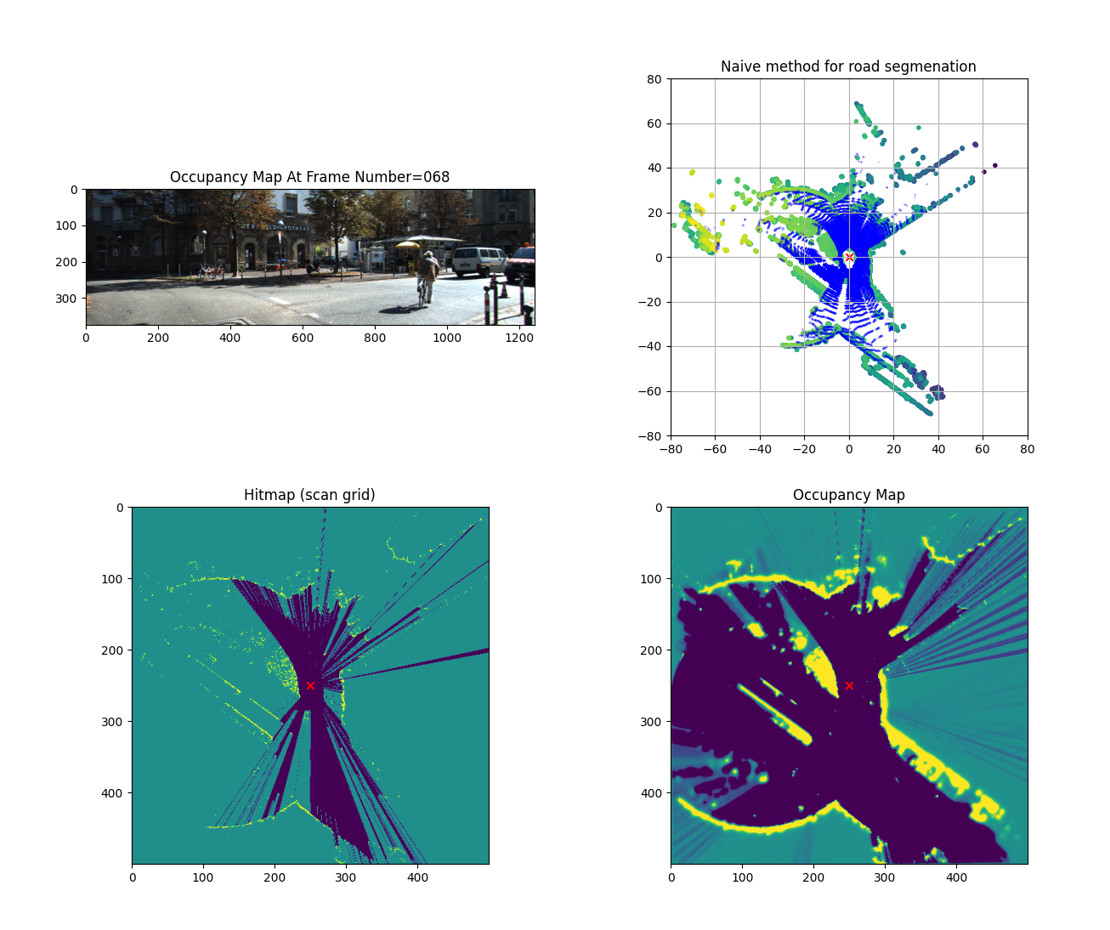
### Analyzation:

1. **Example 1 – frame 50**



The image shows both a good and bad example. The bad example is that in the heatmap SG, the cyclist ("dynamic obstacle") can be distinguished, but in an updated map, he cannot be detected because he was in a previously empty space. The good example is that the updated map accurately shows a free space behind the cyclist, therefore, if the car wea an autonomous vehicle it could navigate towards this space and if it was based on single heatmap the car couldn’t navigate towards this space because it is "unknown".

**Example 2 – frame 68**

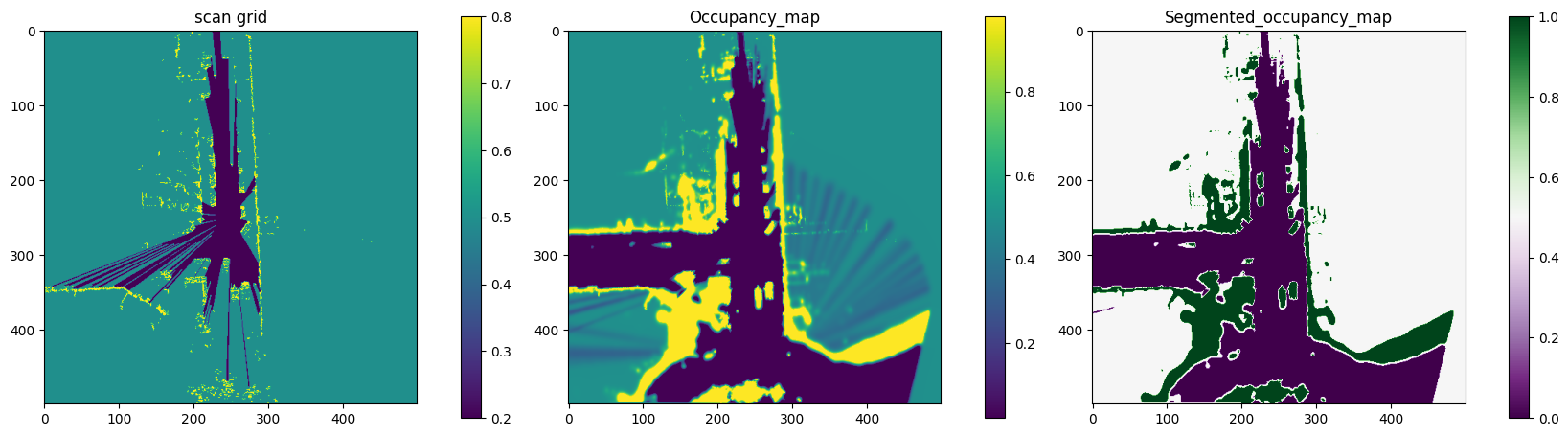


The image depicts both good and bad examples. The bad example is that in the SG heatmap, a false ray of empty space is shown due to a glass window in a building corner. As the car moves, the ray shifts clockwise in the updating OGM, making the data on the map unreliable. However, a good point is that the rays on the updating map are weaker than on the heatmap because there were no rays in previous frames. Additionally (and more trivial), the map accurately shows the general scene of the roundabout and road.

1. **Parameter changed – Beta from recommended 0.1 -> to 0.4 [degrees]**

The angular resolution of Beta in converting LiDAR data to a grid map can directly impact how the sensor's information is displayed on the map layout. By increasing the Beta value to 0.4, more cells surrounding a LiDAR returned point, were occupied. This resulted in two significant improvements for example:

* + 1. **The effect of the unreliable rays was decreased.** We can see this on the final OGM and segmented OGM (with same 0.35&0.8 thresholds) that the rays are weaker significantly and the segmented map is not including them (red arrow)



(See the figure on section 2.g for comparison)

* + 1. **The effect of real objects was increased.** Parking cars now can be seen better on the OGM (see the final frame above) because there were more cells occupied around the LiDAR filtered scans points.
    - The new animation of the results is in the "**Results/figs\_b**" folder under the name **"Animation\_beta\_0.4".**

1. **Objects of different types can affect the occupancy grid map generated using naive Bayesian method in various ways, for example:**

* Dynamic objects: such as pedestrians or moving vehicles can cause abrupt changes in occupancy status, requiring additional measures for motion modeling.
* Parking cars/trees: can obstruct LiDAR's line of sight causing unoccupied cells, which can be mitigated by adjusting the sensor's height.
* Sparse/dense objects: such as poles or buildings can make it difficult to distinguish between occupied and unoccupied cells, but increasing spatial resolution or using additional sensors can help.

## Sensor fusion and semantic segmentation

### Sensor fusion: Projection LiDAR2Image

* 1. Work Process:

The main goal of this section was to project a filtered LiDAR data to the image 2d plane.

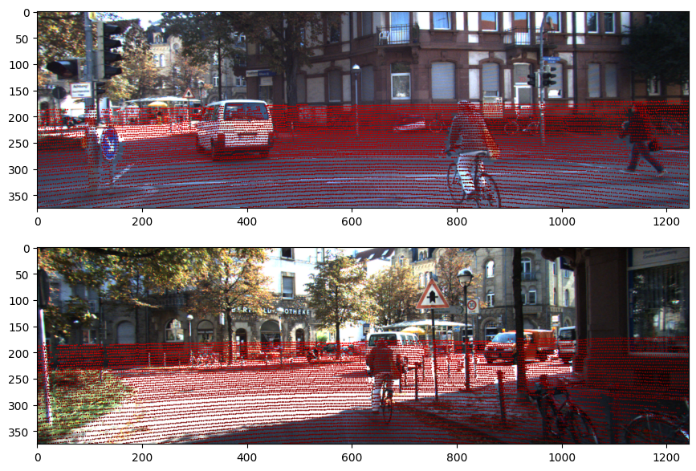
* Filtering the LiDAR raw data below the sensor and at least 2.5 meters away (points[where z<0 & norm(x,y) > 2.5 in the lidar coords system])
* Then the data was projected onto the camera coords system using the extrinsic matrix that was taken from the calibration files and finaly from the cam system to the image plane using the intrinsic matrix that was taken from the calibration files as well. Total flow can be described in the equation below:

Diagram

Description automatically generated

Where K=Intrinsic, [R|t] = Extrinsic

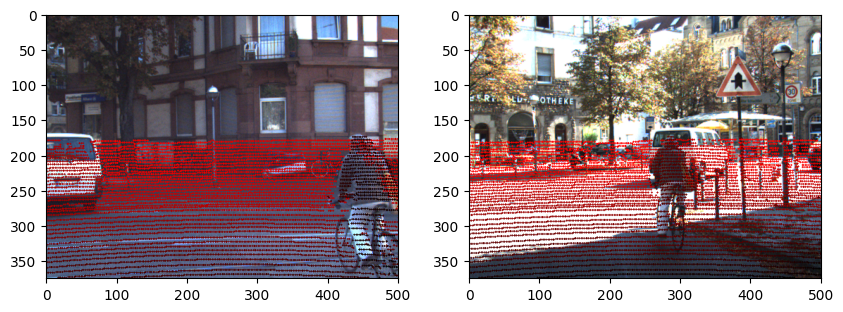
* The points outside the image resolution (=constant WxH) were masked and filtered.
  1. Examples: (up=frame0, down=frame50)

****

### Road Segmentation

* + - * 1. Image Cropping:

Images were cropped with ratio of 4:3 (W:H) as required and then were resized to 513X513 for the DL model. You can see an example below: (left=frame0, right=frame50)



* + - * 1. DeepLabV3+ implementation:

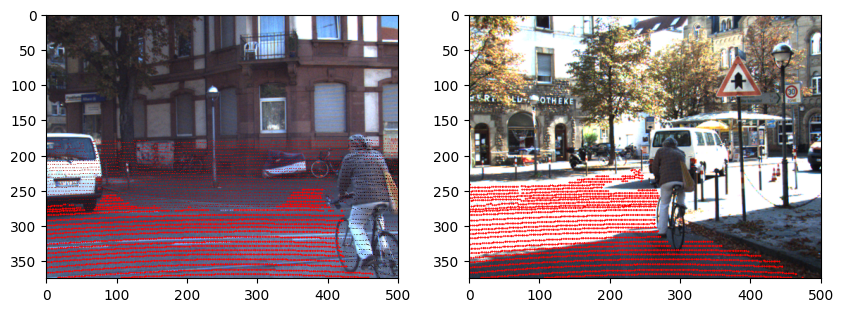
Pre-trained DeepLabV3 was first loaded. The model is already trained on the KITTI dataset. The model takes an image and return the segmentation map of pixels that segmented as "road" , then using a threshold and connected components method the segmented map was tuned and post-processed.

Final segmented image – frame0:

A picture containing calendar

Description automatically generated

* + - * 1. Points corresponding to the road. (left=frame0,right=frame50)



### LiDAR Road Filter:

1. Plane model was calculated from the DL output (=road segmented point)
2. Drivable path was predicted using the model. Example: (left=frame0, right=frame50)

Chart, scatter chart

Description automatically generated

### Probability OGM based on DL:

1. OGM update was repeated over all scans.
2. Animation was created and saved as "**Animation\_default.mp4**" on "**results/figs\_c**" folder.
3. General flow of code:

Initialize OGM and load DeepLab model -> iterate over the frames-> load the img and lidar data -> filter below lidar sensor points as described -> project the lidar points onto the image plane as described -> pre-process the images fused with the corresponding lidar points -> apply DeepLab model on the images and get the segmented road lidar points -> fit a road plane using Ransac regression based on DeepLab outputs-> find the points that located on this plane (approximately) from all the lidar points-> project those points onto the INS (baselink) frame -> get the current car pose from INS system -> shift and update occupancy map!

1. See section b above.

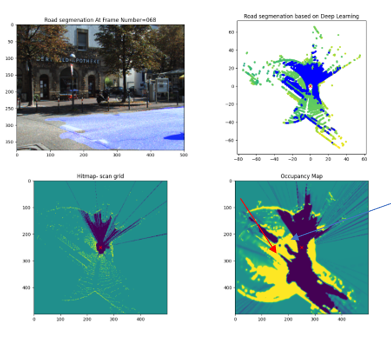
### Analyzation

1. The new mapping based on DeepLabV3 model can be compared and analyzed with the naïve method as follows:

Comparison and analysis:

* The deep learning method generally produces more accurate occupancy grid maps than the naïve method. It seems from first impression that there much less false road segmentation in comparison to the naïve method. It means that the DL model can handle complex and diverse environments better and can identify the road with greater accuracy.
* However, the deep learning method requires a large amount of annotated training data to achieve optimal performance. In addition it requires much more computation capabilities, it reflected in that the occupancy grid map process is run slower than the naïve method.

1. **Example 1 – frame 68**

****

This figure can be seen as a good example and bad as well. On the good side we can see that (on the blue arrow) the model can segment sparse objects such as bushes more accurately as non-road( we can't see the bushes on the cropped image just on the original one). But on the other hand we can observe that road points far away from the camera are less likely to be segmented as road (more False negative), on possible way we can improve it is by increasing the Ransac height threshold (red arrow).

**Example 2 – frame 133**

Graphical user interface, application

Description automatically generated

On this example we can see as well on the good side that the model can predict really accurately the close range road such as between parking cars, but on the bad side it doesn't segmenting a true road points on the far range.

1. Pros and cons of the DL approach:

Three pros and cons of mapping based on deep learning vs. the naïve method are:

Pros:

* Higher accuracy in identifying and classifying objects in complex environments.
* Ability to handle variations and changes in the environment better.
* Can produce more detailed and accurate maps in real-time.

Cons:

* Requires a large amount of annotated data and training time to achieve optimal performance.
* Requires more computation power and can be slower than the naïve method.
* Naïve method follows a more straightforward probabilistic approach that can be easier to interpret, validate and debug.

### Semantic segmentation/DeepLabV3+:

1. Main advantages of Atrous convolution:

* The sparsity of the atrous convolution kernel allows us to expand the receptive field significantly. This means that the convolutional layer can analyze larger sections of the input without reducing the image's clarity or increasing the size of the kernel.
* Atrous convolution reduces the number of parameters in a network compared to regular convolution, which can help reduce overfitting and improve training efficiency by making the network deeper.

1. The Jaccard equation for example is :

**Text, schematic

Description automatically generated**

t = ground truth pix, y = output pix

The main problem of using a geometrical loss function which based on **Intersection over Unit approach** such as Jaccard or Dice criteria to train a segmentation deep learning model is that these loss functions only measure the overlap between the predicted and ground truth masks, but they do not consider other aspects of the objects/images such as the spatial information of structure and shape. In case there is no overlap at all, for example the ground truth has no "class 1" at all and the predictions include even a minimal probability of "1" it still will return approximately a loss value of 1 which is the maximum loss value according to the equation above. This result will happen because the intersection (the numerator) will be equal to zero.

1. Challenges in off-road environments:

* Off-road environments **may not have clear boundaries between the road and surrounding terrain**, making it difficult to identify the road surface.
* Off-road terrain can vary greatly **in terms of elevation, roughness, and texture,** making it difficult to develop a single algorithm that works in all conditions.
* **Limited sensor information in off-road environments** can make it difficult to accurately segment the road surface, due to **factors such as vegetation, uneven terrain, and dust and debris.**

# Summary

This project focused on understanding the KITTI DATSET and using it to create an occupancy grid map that can aid autonomous navigation. We explored two approaches to constructing the map - a naive approach and a fusion of sensors and deep learning tools. The project also highlighted the effects of different parameters on map construction and discussed the trade-offs between short-term accuracy, processing power, and calculation time. We concluded by highlighting the advantages of the DEEPLABV3+ network, used to segmenting road and assist to build the map, while also acknowledging its drawbacks.

**THANKS!**