Mapping and Perception for an Autonomous Robot
Project 1

Nadav Marciano 305165698

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

1. In accordance with the table, I downloaded the data of driver 61 from the KITTI dataset.

    a. The vehicle's path throughout the scenario was almost a straight line with no turns within a residential neighborhood characterized by narrow streets lined with parked cars and tree vegetation on both sides of the road. The shape of the road prevented the simultaneous passage of two vehicles, requiring coordination with oncoming traffic. In the scenario, it appeared that most of the objects were static, such as parked cars, vegetation, and houses, while most of the dynamic objects were cars moving toward the driver. The driving conditions seemed slow due to the narrow road environment filled with obstacles.

Figure 1 - Frame number 110

Figure 2 - Frame number 376 (A vehicle clears the lane while driving)

b. According to Google Maps, this is the route:

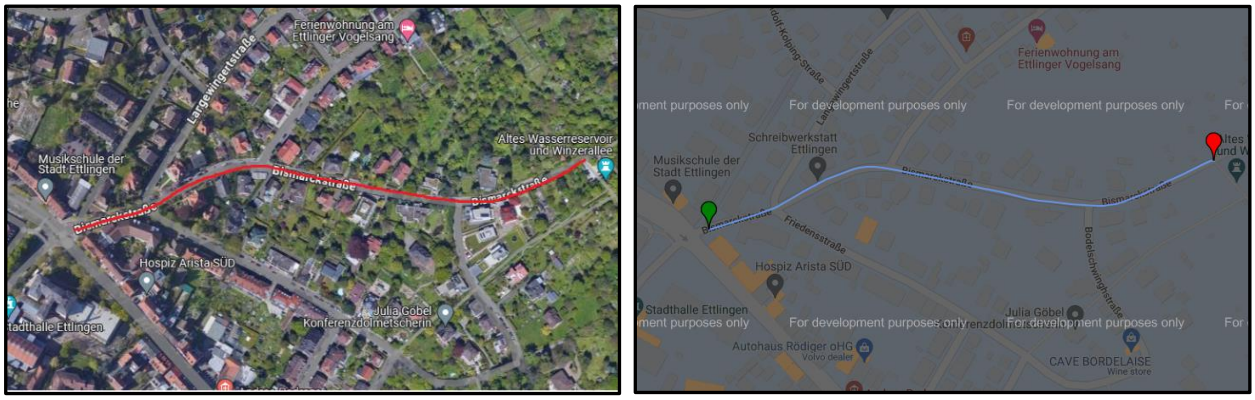

Figure 3 - Vehicle Route on Google Maps

c. Trajectory (ENU and NED): The ENU and NED plots display the vehicle's spatial path relative to its starting point. ENU coordinates represent East-North-Up, while NED coordinates represent North-East-Down. Both methods show an identical trajectory, indicating correct calculations in each system.

Right-side Angular Velocity: This plot presents the vehicle's angular velocity to the right side, showing the rotation rate around its vertical axis (yaw) per frame. The vehicle's right-side velocity values are small, ranging approximately from -0.1 to 1.1 meters per frame.

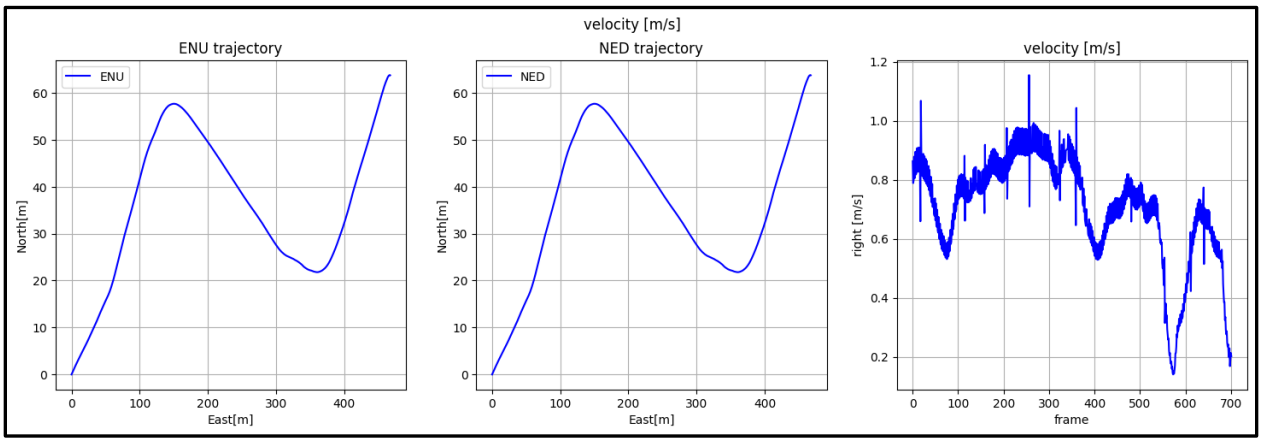

Figure 4 - Vehicle Trajectory in ENU System, Vehicle Trajectory in NED System, Vehicle Right-side Velocity

Orientation (Roll, Pitch, Yaw): These plots depict the orientation of the vehicle over time. Roll and Pitch represent the tilting angles of the vehicle along its lateral and longitudinal axes, respectively. Yaw represents the vehicle's rotational angle around the vertical axis.

Roll and Pitch: During the drive, there are fluctuations of approximately 2 to 7 degrees around the zero axis. These variations represent small oscillations in the vehicle's height and angle, influencing its lateral and longitudinal lean. Yaw: There are noticeable increases of about 20 degrees around axis 10 between frames, indicating rapid yaw movements of the vehicle around its central axis throughout the drive.

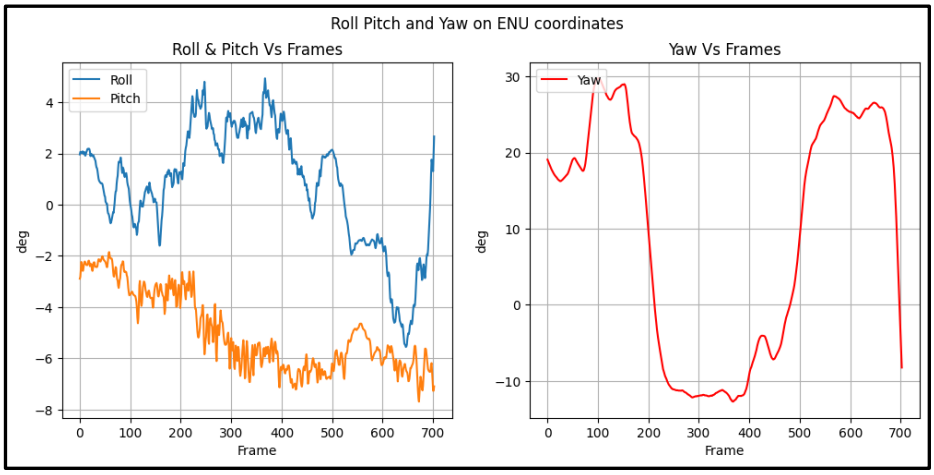

Figure 5 - Vehicle body angles for Roll and Pitch, Vehicle body angle for Yaw

d. The vehicle's journey, which involved traveling between houses and trees, resulted in variability in GPS signal quality. The number of satellites ranged from 5 to 13, with both houses and vegetation contributing to signal obstruction. Despite these fluctuations, the satellite count generally remained above the minimum required for accurate positioning, indicating that the GPS system could still function effectively throughout the road.

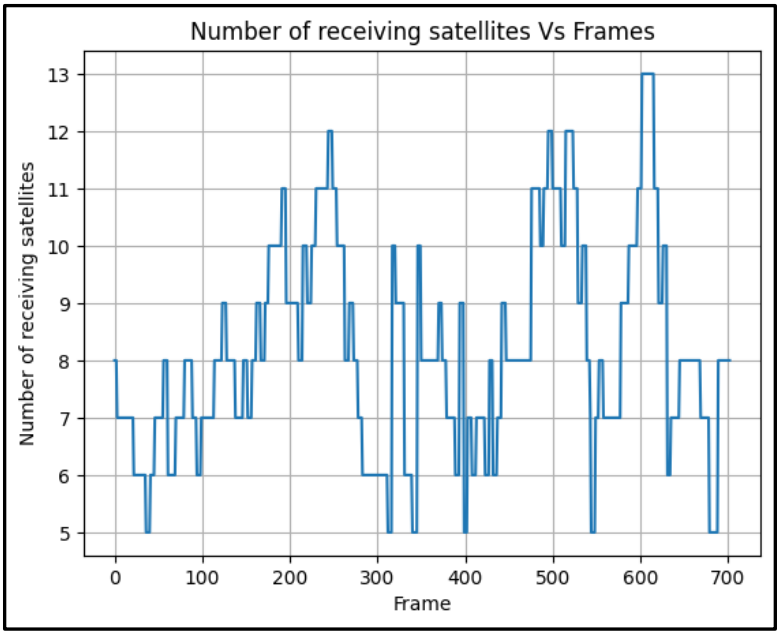

Figure 6 - Number of satellites recorded during the drive

Five GPS satellites are captured in figure 7. The vehicle is located under tall vegetation and between buildings that hide the sky, and affect the reception of the satellite.

Figure 7 - Frame number 400

There are 13 GPS satellites captured in Figure 8, indicating maximum reception along the route. The open sky suggests unobstructed and well-illuminated driving conditions, with no blockage from buildings or tall trees.

Figure 8 - Frame number 600

e. In autonomous driving, the integration of different sensors, such as cameras and LiDAR, plays a crucial role in the perception task. Each sensor has unique advantages that complement each other, enabling the vehicle to make informed decisions and navigate safely.

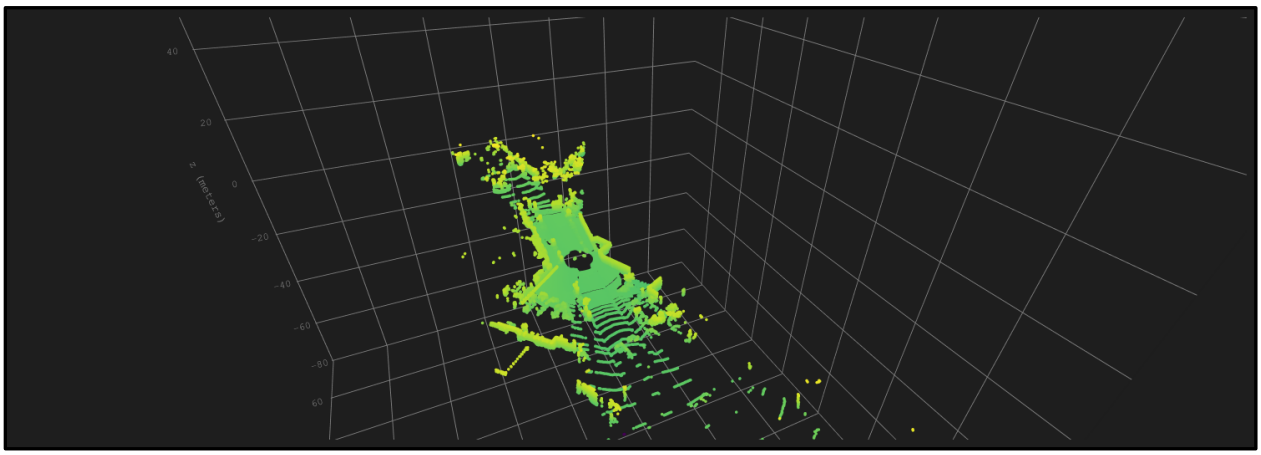

Figure 9 - The point cloud that corresponds to frame number 33

Figure 10 - Frame number 33

You can see the point cloud corresponding to figure 10. Our vehicle is located in the empty black area in the center, while the fully green area displays the vehicle's driving route. Comparing the images reveals that the point cloud obtained matches expectations approximately 20-30 meters ahead can be seen. Towards the upper right, there is a black area resembling a gap further down (identifying the truck ahead).

In the described scene where a truck is entering the road from the right and blocking the lane, while the driver continues to drive straight, the combination of both sensors provides a comprehensive understanding of the situation. Benefits of Each Sensor in the Scene:

Camera:

- High Resolution and Detail:

Captures detailed images with high resolution, allowing the system to identify and classify objects like trucks, cars, and road signs with high accuracy.

Scene Example: The camera can clearly see the truck entering the road from the right and can distinguish it from other objects on the road.

- Contextual Understanding:

Offers a comprehensive view of the scene, allowing for better interpretation of the context and predicting the behavior of other road users.

Scene Example: The camera can capture the entire scene where the truck is entering and the driver is continuing straight, providing context for decision-making.

LiDAR:

- Depth Perception:

Provides accurate 3D distance measurements, enabling the vehicle to understand the spatial structure of its surroundings.

Scene Example: LiDAR can accurately measure the distance to the truck and other nearby objects, helping the vehicle to maintain a safe distance.

- Works in Low Light:

Operates independently of lighting conditions, making it effective in low light or nighttime scenarios.

Scene Example: Even if it is dark or the lighting is poor, LiDAR can still detect the truck entering the road.

- Obstacle Detection:
Detects obstacles with high precision, which is essential for collision avoidance and path planning.
Scene Example: LiDAR can detect the truck as an obstacle blocking part of the lane and help the vehicle plan a safe path to avoid a collision.

Part B: Probabilistic Occupancy Grid

1.

a. Drivable path in the session that filters out all point clouds above 30 cm and points that are at least 2.5 m away from the sensor:

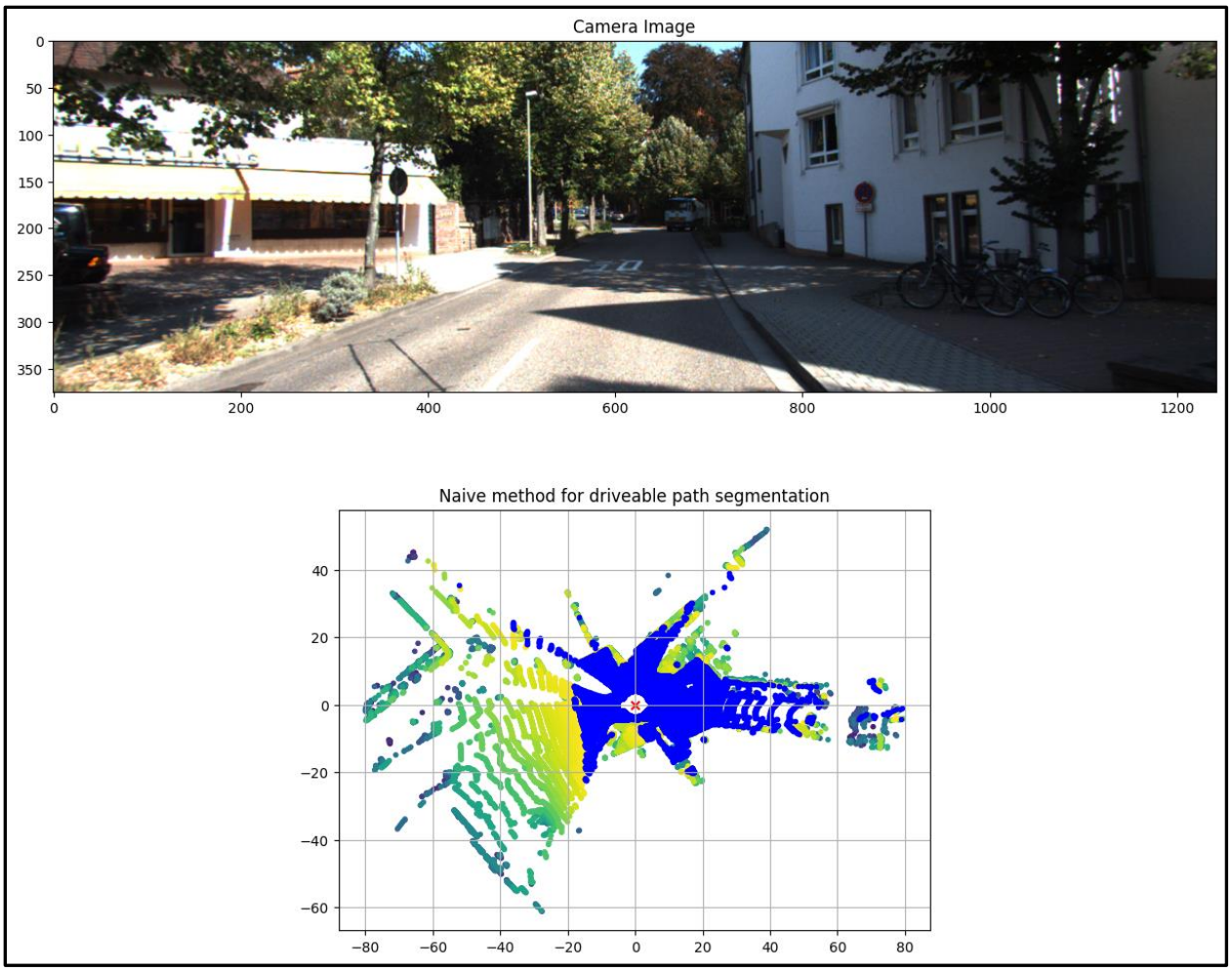

Figure 11 - Frame number 1, Naive method for drivable path segmentation (from up to down)

b. The functions for constructing the Occupancy Grid were implemented according to the following parameters:

Parameter	Value	Description
Resolution	0.2 (m)	cell size
ALPHA	0.2 (m)	The radial resolution when converting LiDAR data to grid map
BETA	0.1 (degree)	The angular resolution when converting LiDAR data to grid map
Max range	100 (m)	Maximum range of LiDAR points that will be converted to grid map.
Hit/Miss prob	0.8/0.2	Probabilities of hit and miss values
Probability Saturation	0.02/0.98	Minimum and maximum probability
Map width	100 (m)	Width of the map from side to side
SPHERICAL2CARTESIAN_BIAS	1 (pixels)	An adjustment needed due to some errors when converting the spherical grid map to the cartesian grid map.

Table 1 - Values of the parameters

Based on these parameters, the following maps were generated for frames 0 and 1:

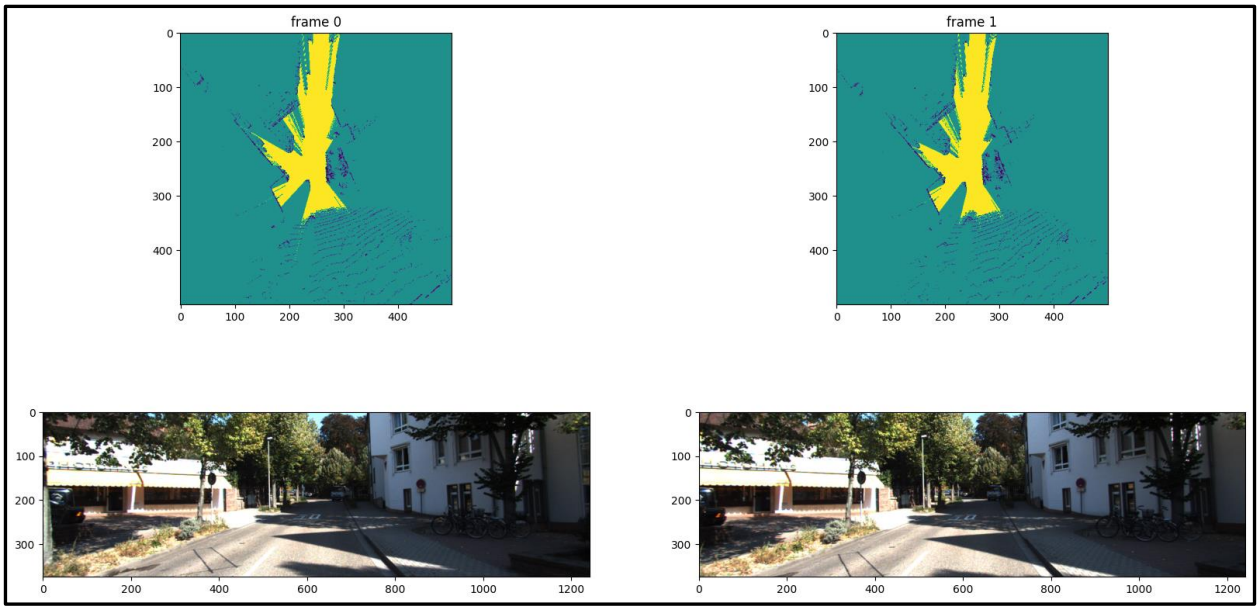

Figure 12 - An occupied grid image is based on the point cloud and the camera image for frame 0 and frame 1

It can be observed that there is no significant difference between the occupancy grid images of the two frames. This is reasonable considering that the vehicle moved only a few meters and no substantial changes occurred in the environment.

c. Implementation and Results:
- The first step was filtering the point cloud. In this step, we filtered the points using the `Filter_PC` function to include only the relevant points within our area of interest.
- The next step was to create an occupancy grid map based on LiDAR scan measurements. The `generate_measurement_ogm` function converts the polar coordinates of the LiDAR points

into a Cartesian occupancy grid map.
- The occupancy grid map was then updated using the `update_ogm` function, which calculates the logit function and its inverse to update the map with new measurements.

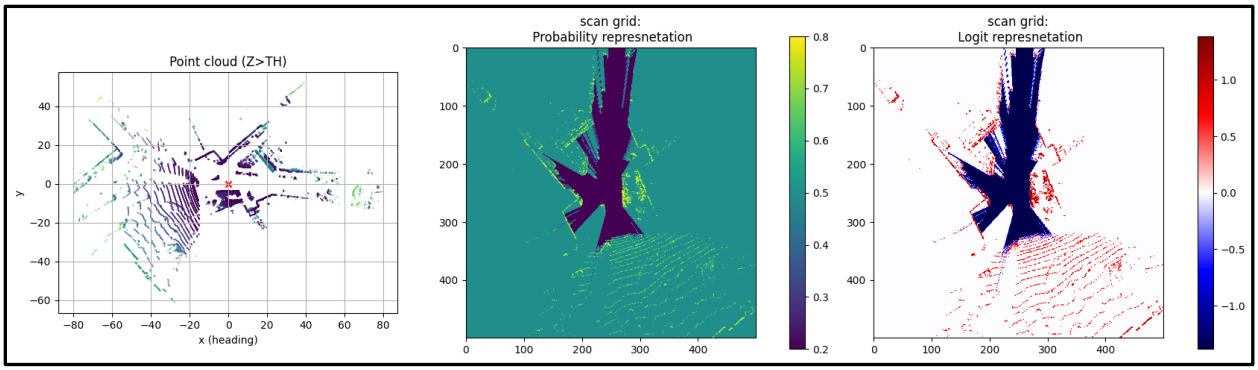

Figure 13 - Point Cloud, Probability Representation, Logit Representation (from left to right)

Following these steps, we filtered the data from the point cloud, created an occupancy grid map, and updated it with new measurements. The results were then visualized in various representations:

Point Cloud Visualization:
- The first subplot shows the point cloud data with the points color-coded based on height (Z-axis).
- The red 'x' marks the location of the LiDAR sensor.

Scan Grid - Probability Representation:
- The second subplot displays the shifted occupancy grid map in the probability space.
- This representation indicates the occupancy probability of each cell, where higher values indicate higher occupancy probability.

Scan Grid - Logit Representation:
- The third subplot shows the logit-transformed occupancy grid map.
- This representation is useful for understanding the changes in the log-odds space, which helps in updating the map with new measurements.

d. OGM Update and Shift According to Vehicle Pose, First Two Frame Demonstration:

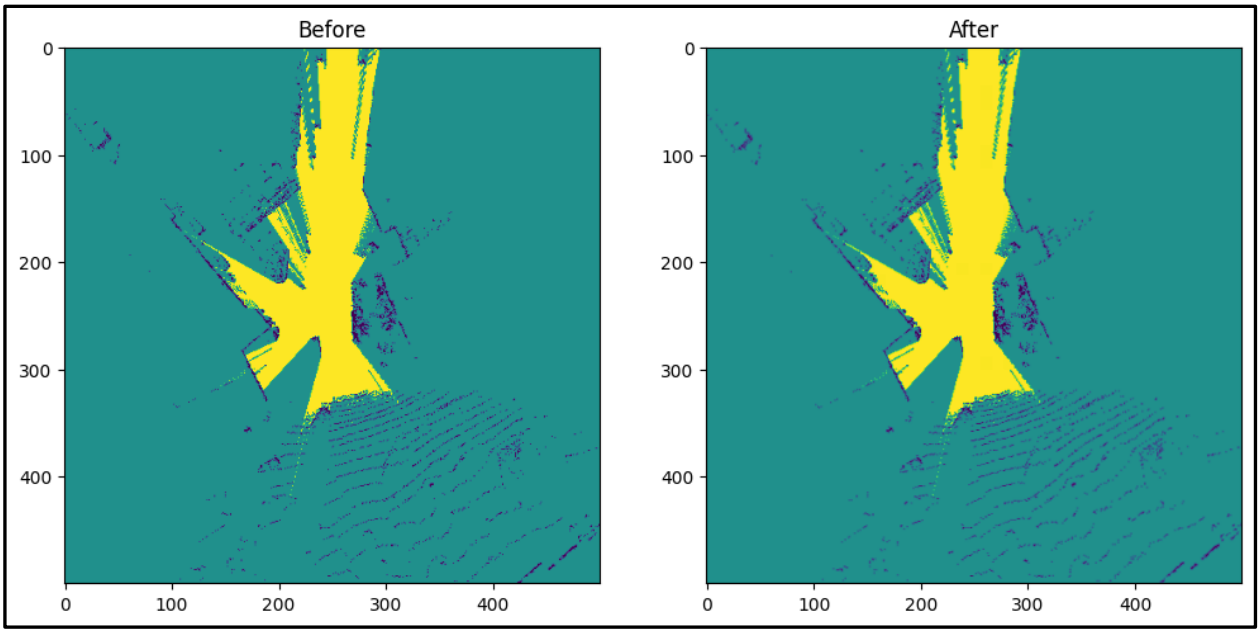

Figure 14 - OGM of frame 1 before and after the shift

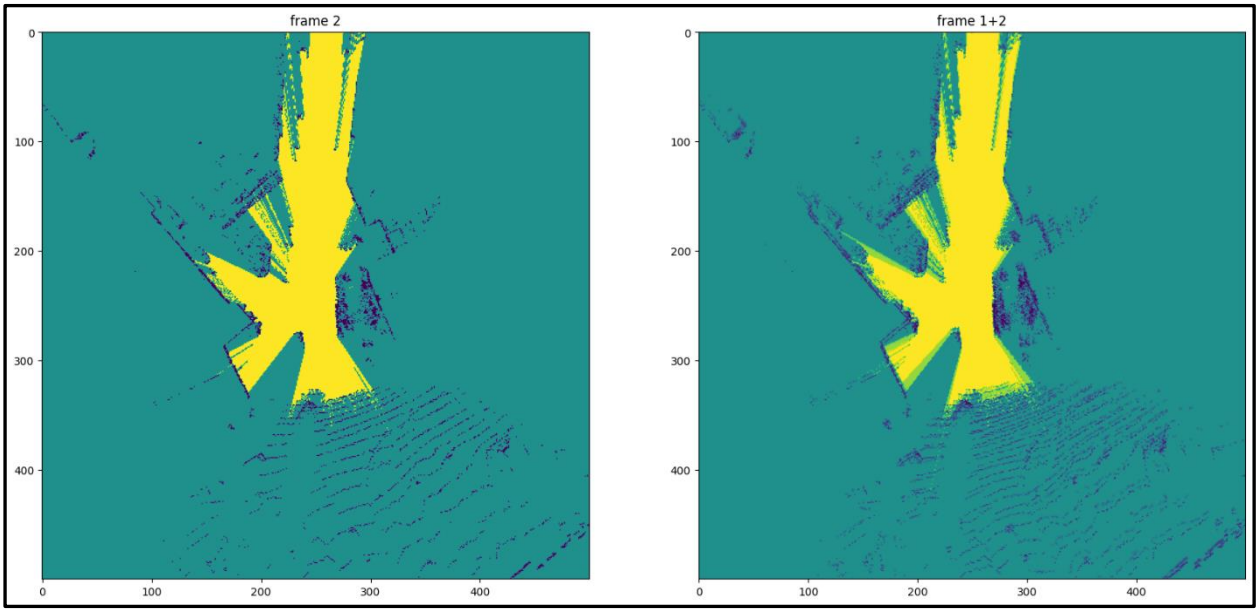

Figure 15 - OGM of frame 2, OGM of frame 2 after adding the shifted frame 1 (from left to right)

e. The OGM evolves frame by frame as the vehicle moves, reflecting changes in detected obstacles and free space based on lidar measurements.
Each frame shifts the OGM according to the changes in vehicle pose (pose_prev to pose_curr). This ensures the map stays aligned with the vehicle's movement. The overall representation of detected obstacles and free space. When connecting two consecutive frames in the Occupancy Grid Map, we observe that areas classified as free or unknown tend to show a reinforcement of probabilities. This is noticeable in the color map representation where these areas exhibit increased probabilities. It reflects the cumulative effect of multiple measurements, enhancing confidence in identifying free space while highlighting persistent uncertainty in ambiguous regions.

Figure 14 shows no significant change between the images since the vehicle moved only a few meters forward, corresponding to approximately 30 pixels. Given the resolution and size of the diagram (500x500), these 30 pixels represent just 6% of the image. Therefore, when displayed at a smaller size, it becomes challenging to discern the difference, making the change visually indistinguishable.

In Figure 15, the Occupancy Grid Map (OGM) of the current frame is shown before and after incorporating the historical OGM. The accumulated history is aligned to the current vehicle position. It's noticeable that after integration, the image appears slightly blurred, likely due to variations in the vehicle's viewing angle of obstacles across frames. Storing history as an image rather than a collection of point clouds introduces potential inaccuracies when aligning historical and current positions compared to recalculating the OGM based on the previous point cloud position and then re-evaluating it. However, this blurring isn't significant since areas with sharp obstacles sharpen over time, similar to how open areas remain consistent.

2.

e. The construction process of OGM all the way in the car is done in the following way:

1. Initial initialization of the OGM.
2. Loop on the frames indices, within which the following is performed:
   a. Loading the image and corresponding point cloud for the current index.
   b. Filtering points from the point cloud based on the processes described in previous sections.
   c. Transition from the lidar coordinate system to the IMU coordinate system for rotation and translation
   d. Calculation of the forward and rightward velocities relative to the previous frame and the current yaw angle accumulated so far based on the current speed and angle.
   e. Updating the OGM based on the calculated movement. f. Accumulating the OGM based on the current frame's point cloud and adding it to the accumulated history so far.
   f. Presentation of results per index frame: Camera image, point cloud of the current frame distinguishing between obstacle points and free-space points, and the OGM figures derived from the accumulated history including the current frame.

f. Last frame of the occupancy map:

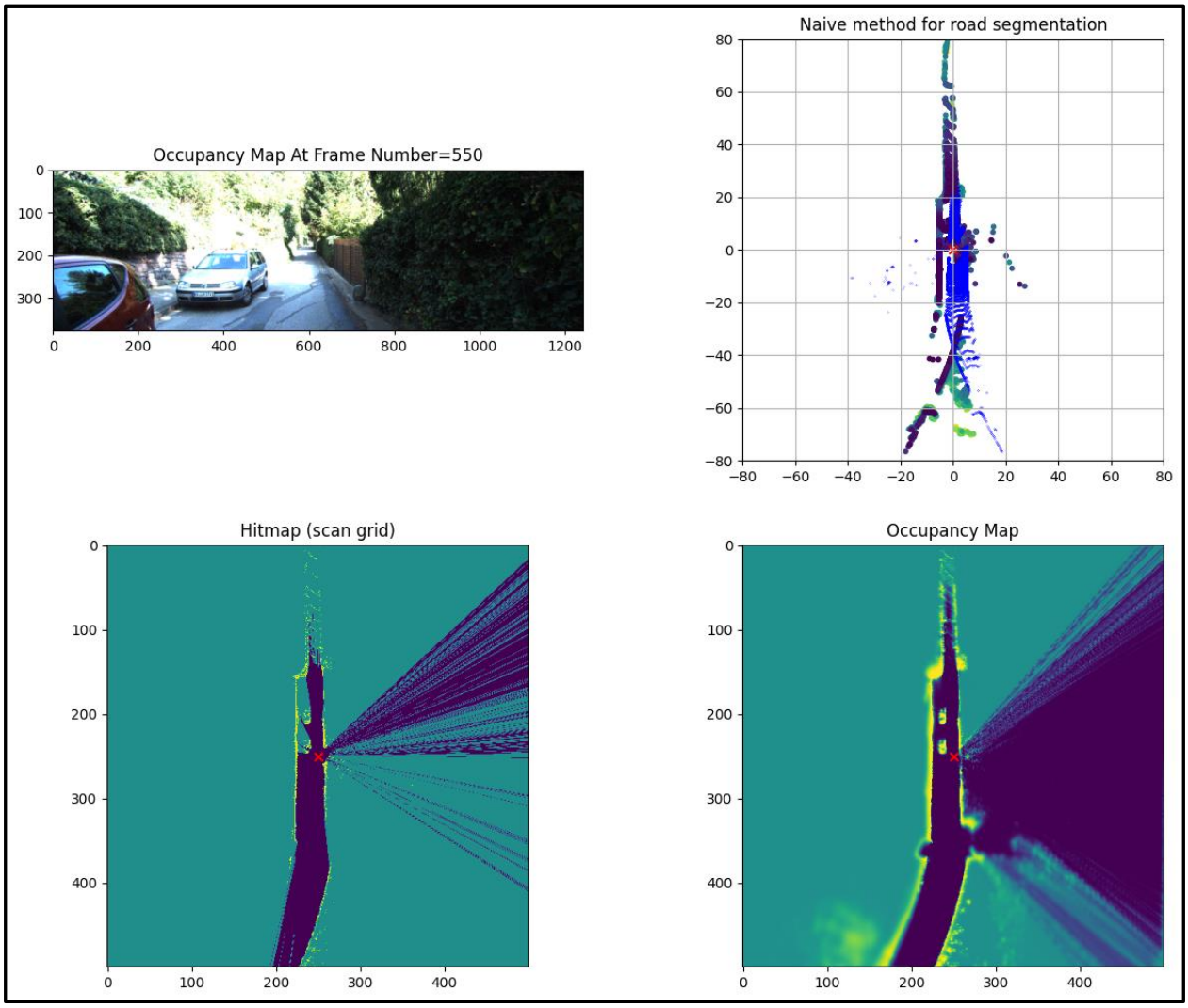

Figure 16 - Last frame 550 of the occupancy map

g. Segmentation of OGM Using Thresholds:

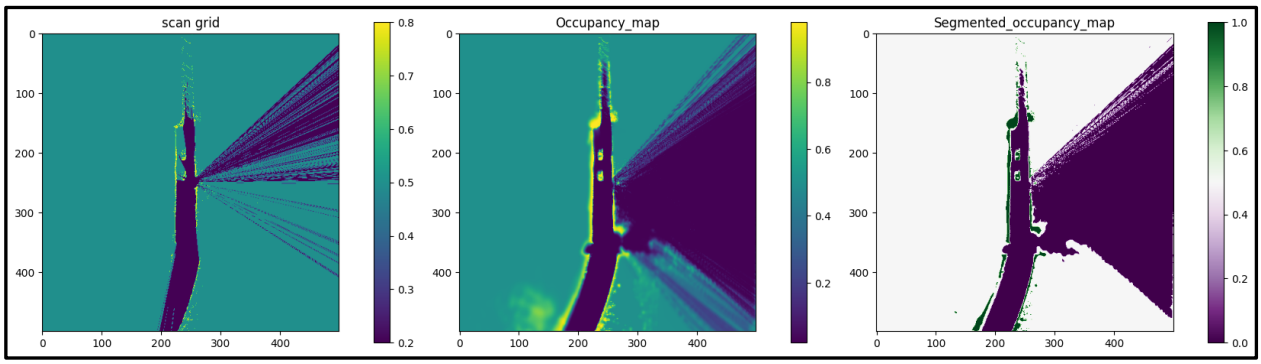

Figure 17 - Scan grid of frame 702, OGM of frame 702, Segmented OGM of frame 702 (from left to right)

Right-hand side shows the post-filtering OGM, where obstacle-free areas are marked in purple, areas containing obstacles are marked in green, and uncertain areas are marked in white.

3.

a. Here are some good examples:

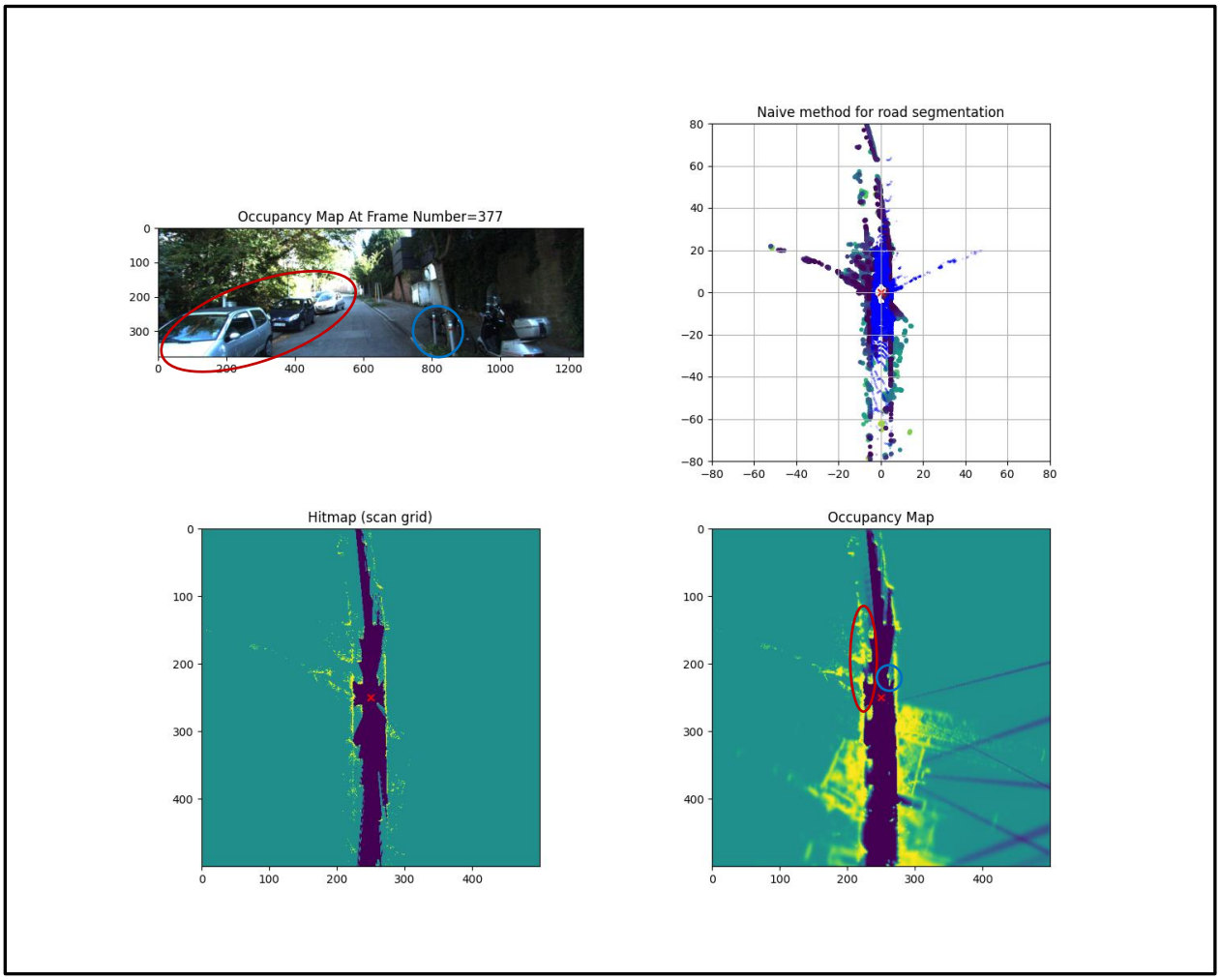

Figure 18 - Frame 377 of the occupancy map

In Figure 18, the LiDAR detects parked vehicles on the left (marked in red) and pillars on the sidewalk on the right (marked in blue). This example demonstrates that the chosen OGM parameters are sufficiently sensitive to clearly display these features.

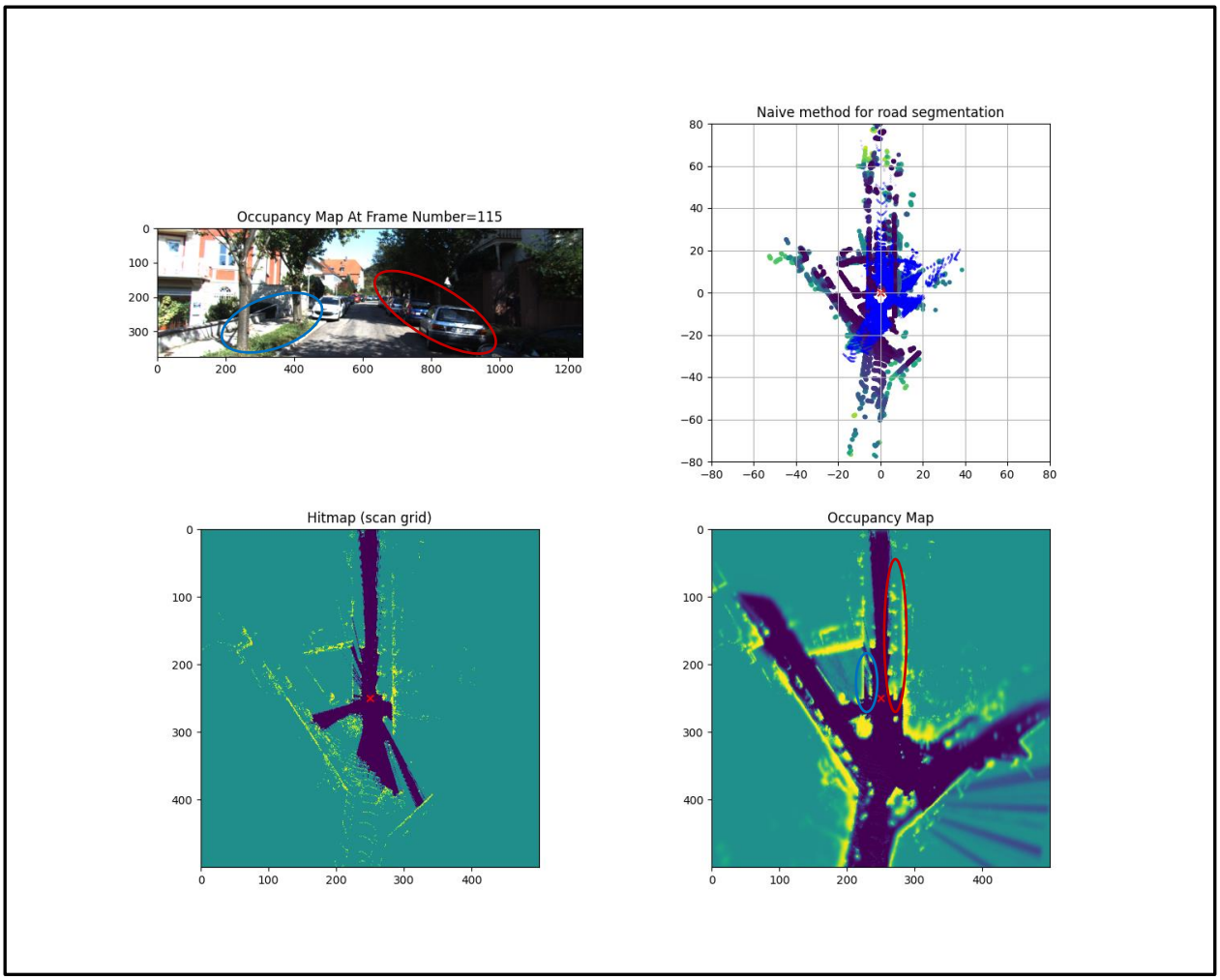

Figure 19 - Frame 115 of the occupancy map

In Figure 19, the LiDAR detects vehicles parked on the right (marked in red) and an avenue of trees (marked in blue). This is just one of many examples, as I have selected and displayed various frames from the dataset.

Here are some bad examples:

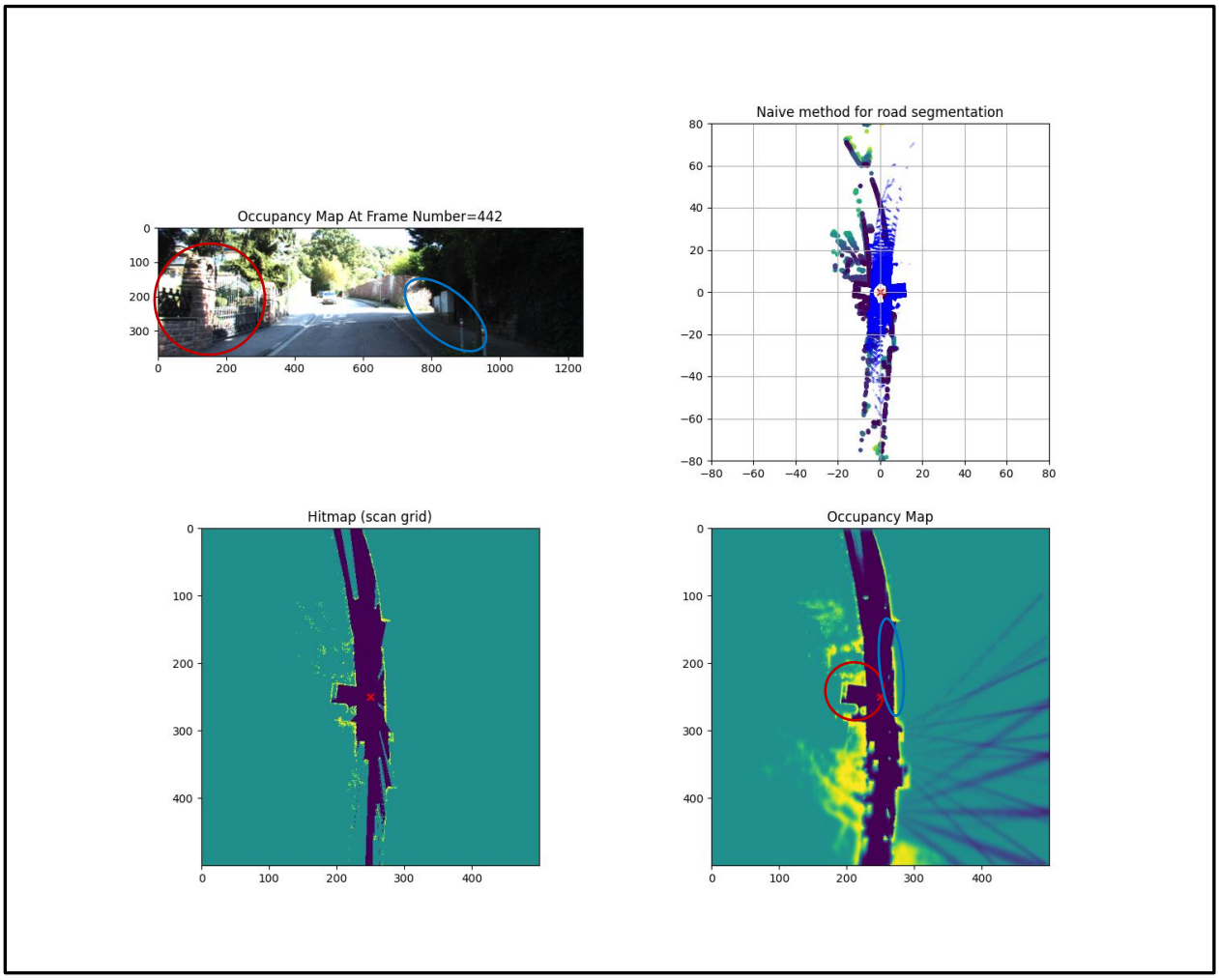

Figure 20 - Frame 442 of the occupancy map

In Figure 20, the LiDAR detects a clear road in a residential area (marked in red). The scene is relatively dark and shaded, and it appears that the apartment fence is low and not sealed, allowing the LiDAR rays to pass through it. Additionally, the sidewalk is recognized as part of the road due to the residential neighborhood's layout, which includes low curbs to allow access to private driveways from the street. This makes it difficult to distinguish the curb's condition. It is important to take this into account and adjust the height of the point filter in the system to accurately handle these situations (marked in blue).

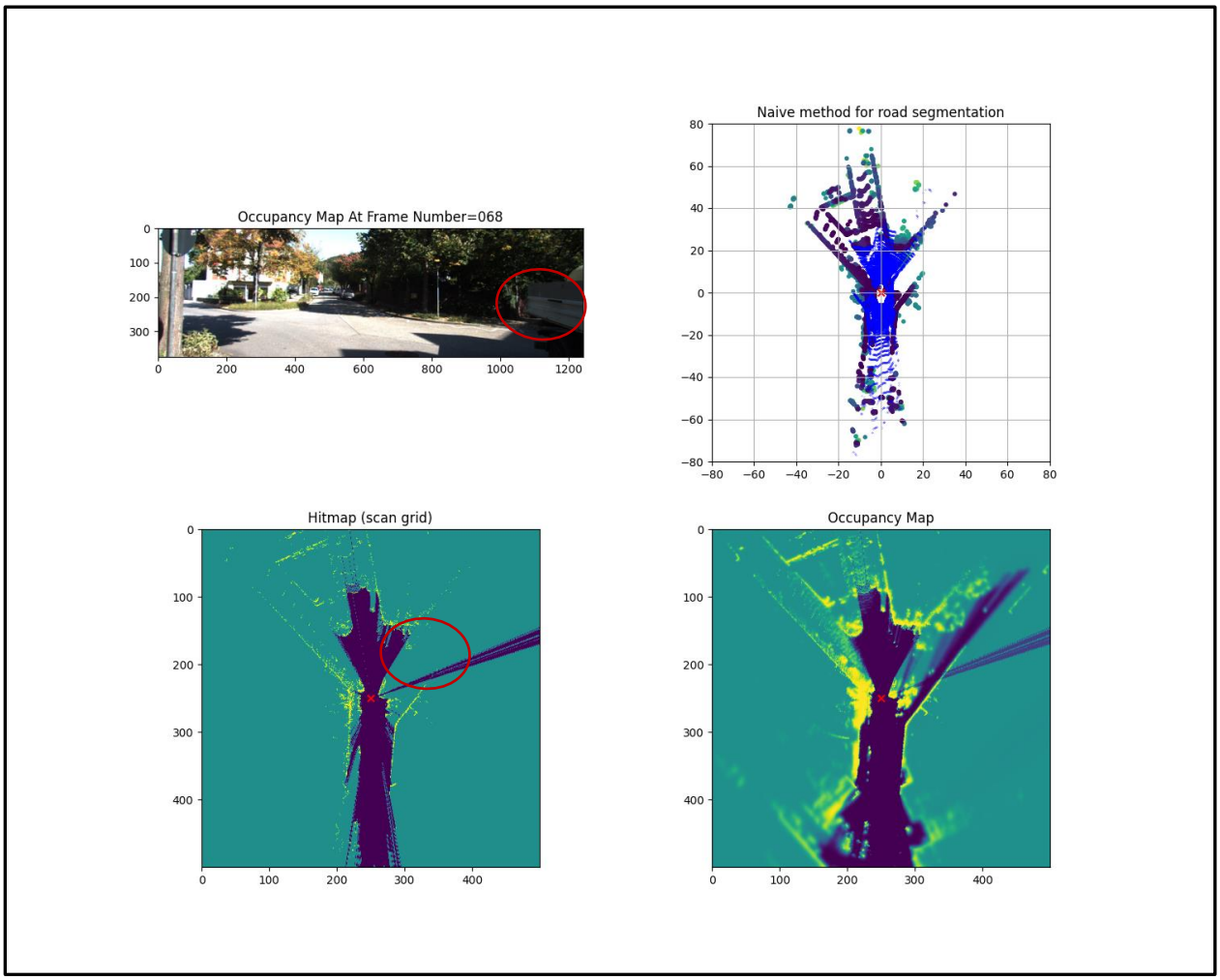

Figure 21 - Frame 68 of the occupancy map

In Figure 21, the LiDAR emits rays on the right side of the image where there is a truck present (marked in red). This occurs due to the minimum threshold value defined in the cloud filtering function. The threshold is set to 2.5 meters, and it is clear that the distance of the truck from the vehicle is less than 2.5 meters. This means that the points belonging to the truck are filtered out from the point cloud, giving the appearance that there is no obstacle in this direction.

b.

The change I made is that I lowered alpha from 0.2 degrees to 0.1 degrees and beta from 0.1 degrees to 0.05 degrees. The goal of this change was to improve resolution in both range and angle to clarify obstacles in the occupancy grid map, allowing them to occupy more pixels or cells.

Alpha (α) = 0.1: This reduced alpha value means that each new measurement now has less influence when updating the occupancy grid map compared to the previous setting.

Beta (β) = 0.05: This reduced beta value means that the past measurements have a slower decay, making them more influential in the OGM over time compared to the previous setting.

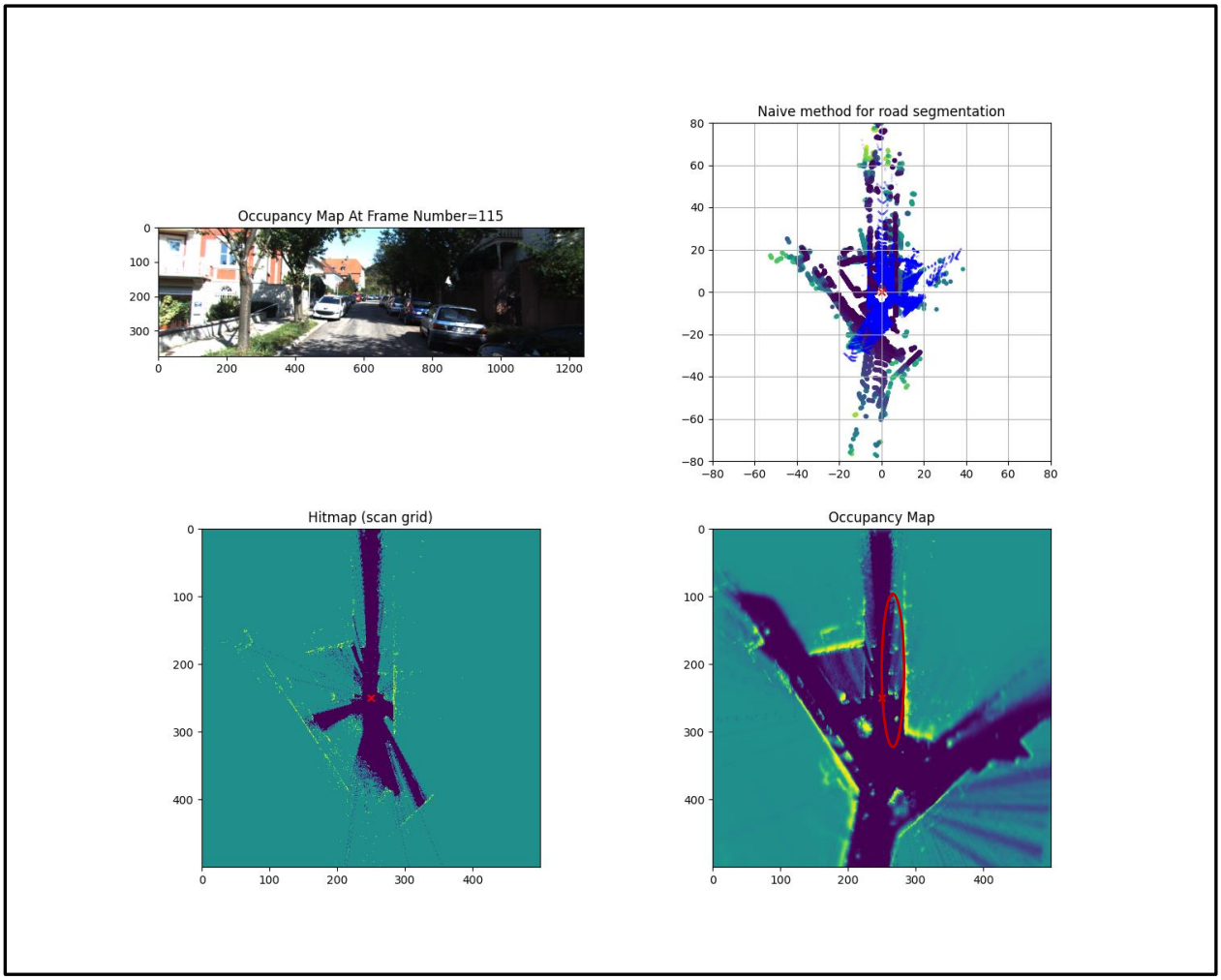

Figure 22 - Frame 115 of the occupancy map (α=0.1 ,β=0.05)

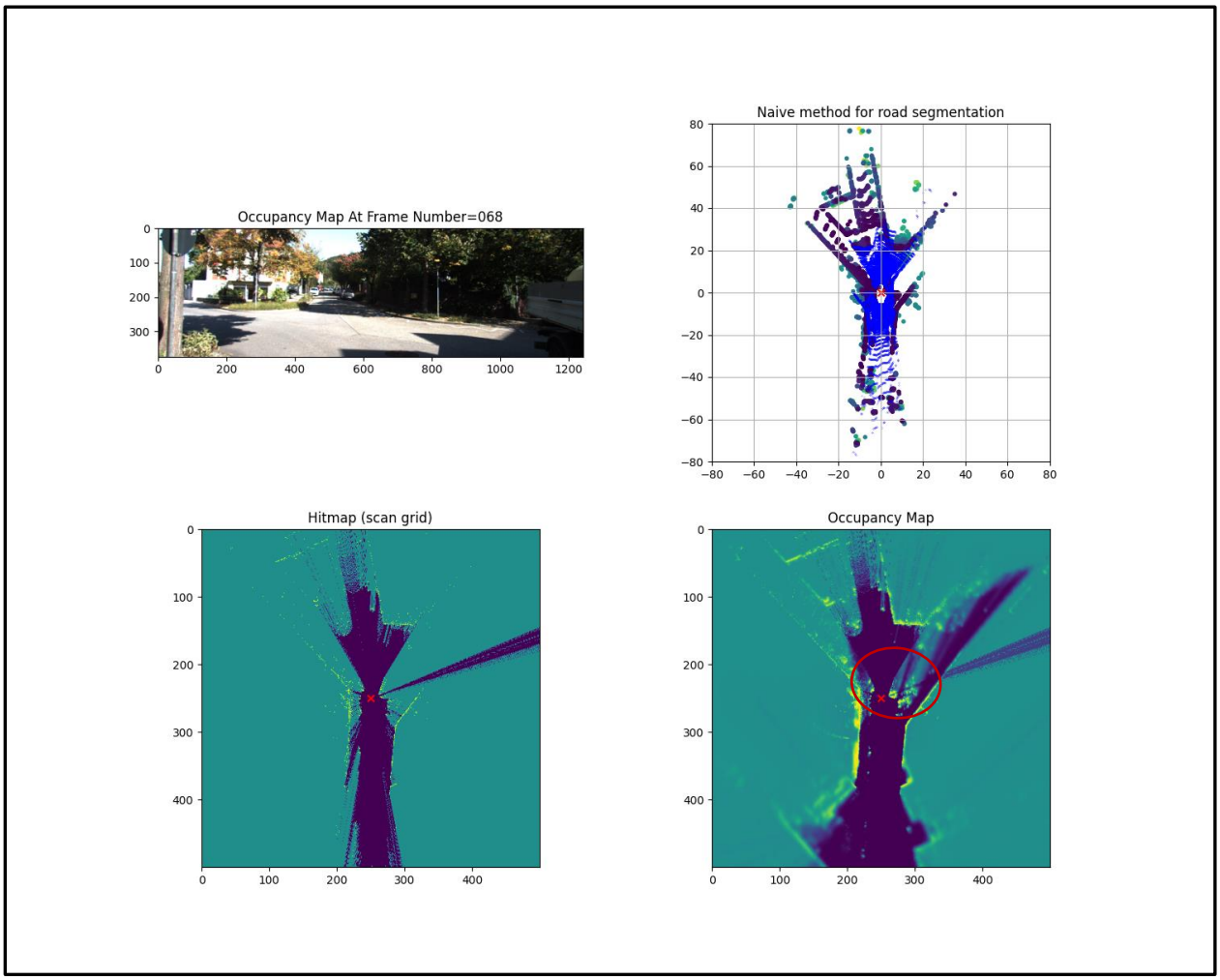

Figure 23 - Frame 68 of the occupancy map (α=0.1 ,β=0.05)

Comparing figures 22 and 23 with those from the previous section, it's clear that the expected improvement in clarity and precision did not materialize as anticipated. The mechanics of parked vehicles are less distinct, resembling an image of a tree-lined avenue based on the provided values. In addition, the area where the truck is visible is less good than the original values on the OGM.

In practice, the results from this change have been contrary to the original intention. It seems that the increased angle resolution causes the same object to appear at varying angles consistently, without achieving sufficient clarity to confidently distinguish between obstacles and free space. There isn't enough cumulative "energy" over time in specific cells to clearly indicate the presence of an obstacle.

c. The mapping is affected by objects of different types:

Dynamic objects: These can include moving vehicles, pedestrians, or animals. Their presence introduces variability in occupancy maps over time. Dynamic objects might appear and disappear from frames, leading to transient occupancy changes that reflect their movement patterns. These examples can be seen in Figure 21. At the beginning of the recording, a truck approached from the front, blocking the right-turn lane, which was not visible on the map initially. Only after it moved away, it became possible to identify that there was a lane there.

Parked cars/trees: Stationary objects like parked cars or trees typically create consistent occupancy patterns. They can obstruct LiDAR or camera views and appear as persistent obstacles in the occupancy grid map. Trees, for instance, might show up as solid obstacles due to their density and structure. Examples can be seen in figures 18 and 19.

Sparse/dense objects: Objects vary in their spatial density. Sparse objects might not fully occupy their surrounding grid cells, resulting in intermittent occupancy readings. Dense objects, on the other hand, occupy more cells densely and can be detected more reliably across frames. In Figure 18, you can see the small columns on the edge of the sidewalk marked in blue.

Transparent materials (such as glass): Transparent or semi-transparent materials pose challenges to LiDAR and camera sensors. They can cause reflections or refractions that distort sensor readings, leading to inconsistent occupancy map representations.

Different colored objects of the same type: Objects of the same type but different colors can affect how they are perceived by sensors. For instance, a black car might reflect light differently compared to a white car, altering LiDAR intensity returns or affecting camera-based segmentation algorithms. I searched for examples of black cars causing issues in the map, but did not find any. the LiDAR managed to detect them, likely due to differences in lighting conditions caused by sunlight in those brightly lit areas.

Part C: Senor fusion and semantic segmentation

1.

a. Work Process Description:

1. Loading the image and point cloud data of the current frame.
2. Filtering points within a distance less than 2.5 meters from the sensor and discard points with heights higher than the sensor's height. This filtering is practical as higher points are less relevant as obstacles for the vehicle.
3. Coordinate the point cloud from lidar's coordinate system to camera 2's coordinate system using the extrinsic calibration matrix.
4. Filtering points from the points cloud that are behind the camera, as they are irrelevant for processing.
5. Conversion of the points from the point cloud to the image's coordinate system using the intrinsic calibration matrix to associate pixels in the image with points in the point cloud.
6. Filtering out points from the point cloud that exceed the boundaries of the image due to the camera's limited and specific field of view.

b. Examples of projection onto image coordinates:

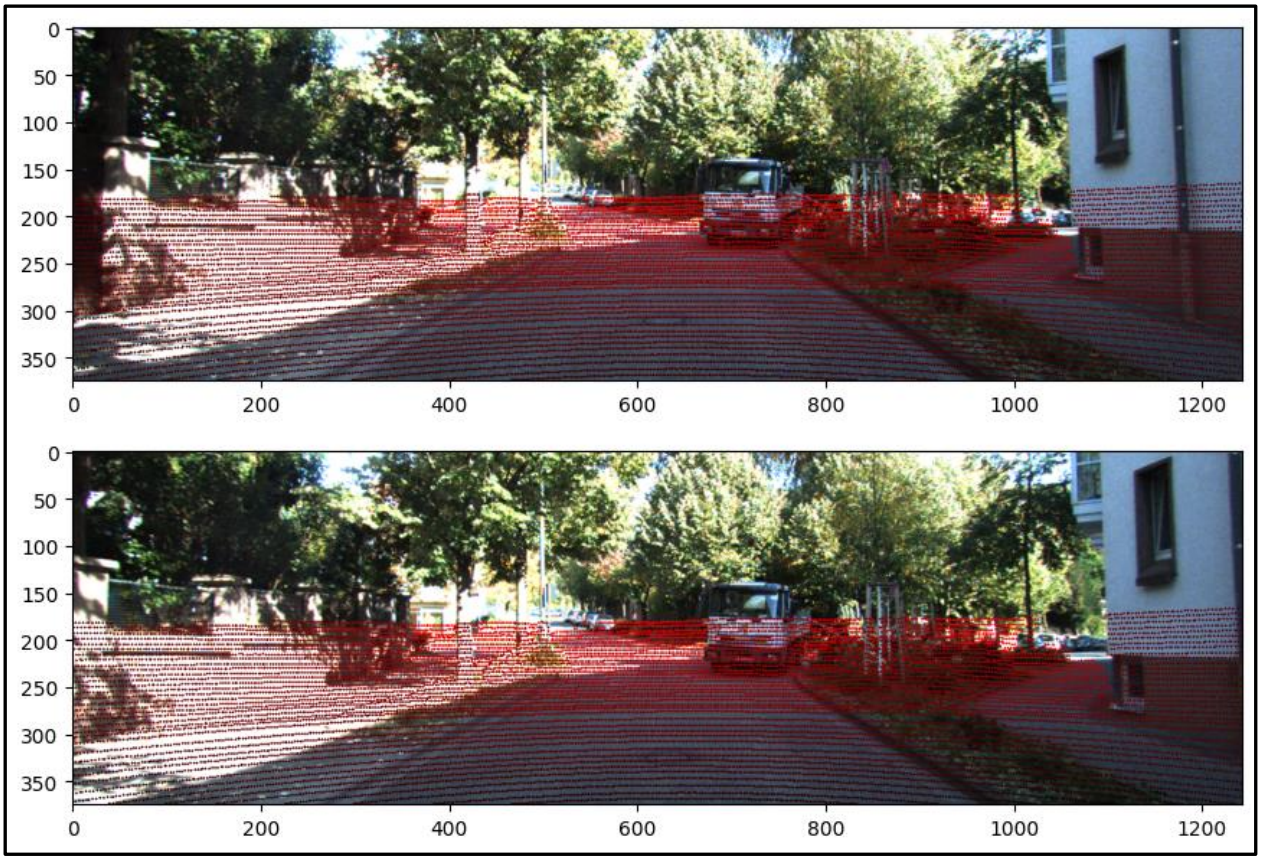

Figure 24 - Projection onto image coordinates frame 40 and frame 41

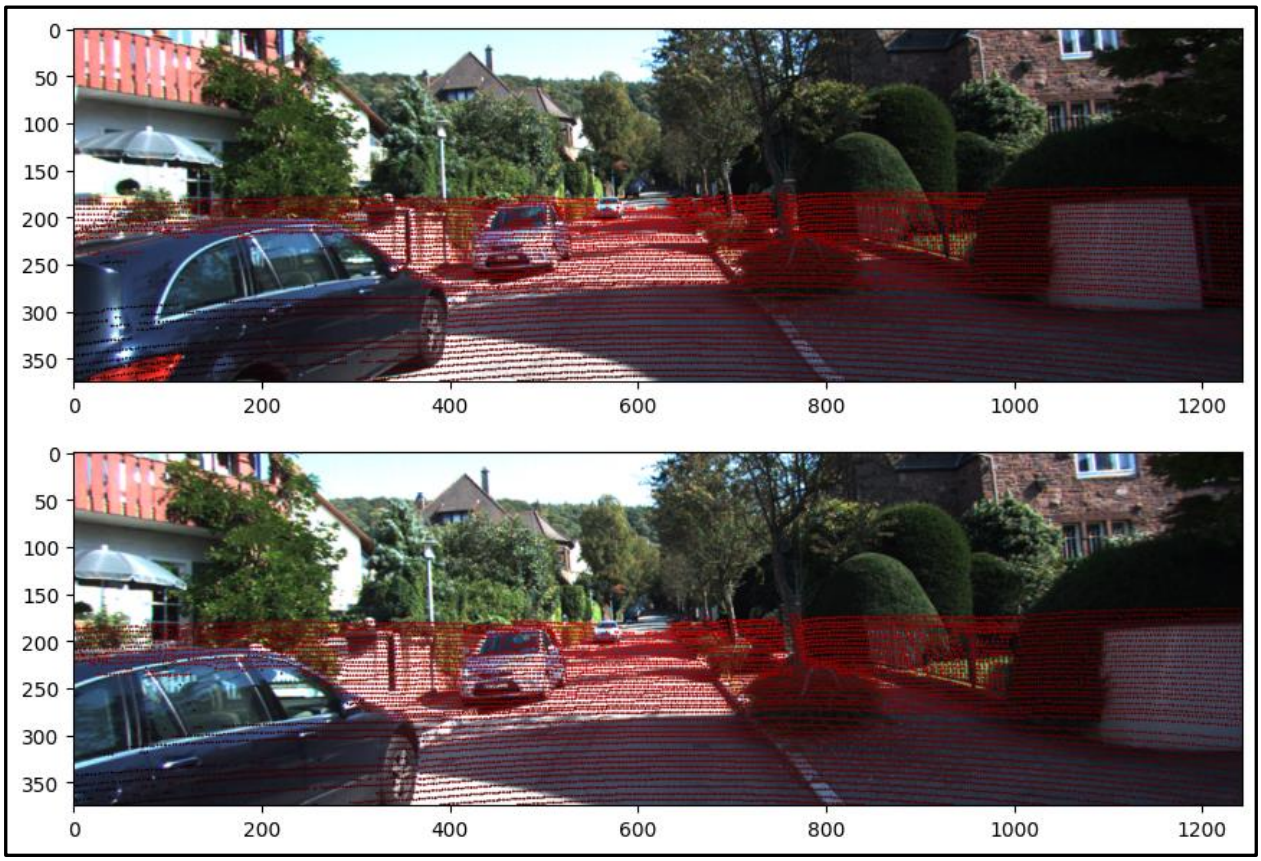

Figure 25 - Projection onto image coordinates frame 240 and frame 241

2. a. Cropping the image:

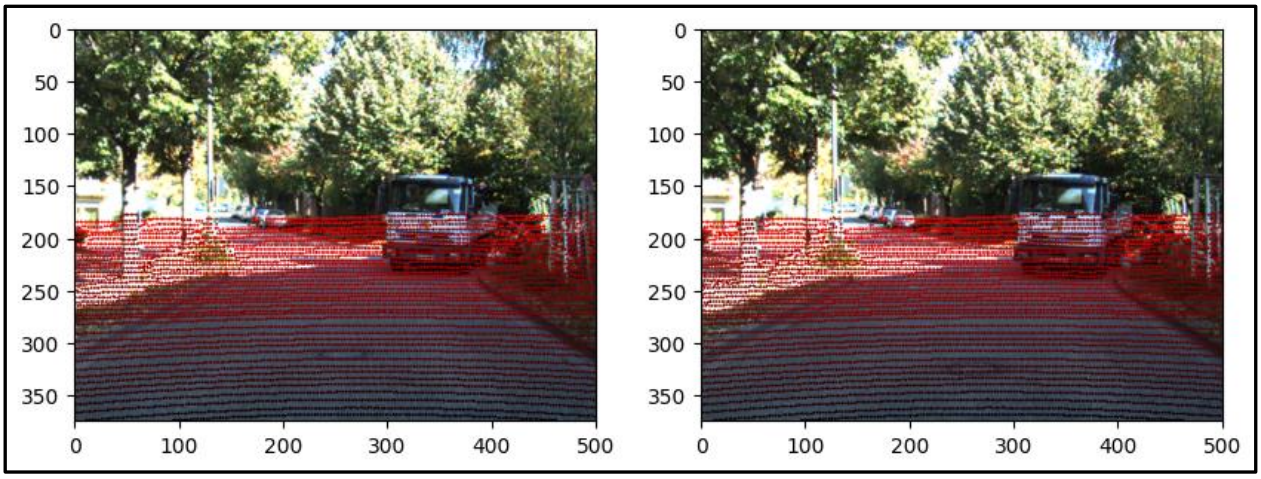

Figure 26 - Cropped image frame 40 and frame 41

b. In the version we use, the DeepLabV3+ network is trained on images with a 4:3 aspect ratio that have been resized to 513x513 pixels. To use it, we follow these steps:

1. Adjust the images and point cloud to fit the network by cropping them around their center to achieve a 4:3 aspect ratio.
2. Filtering the point cloud so that only points within the image pixel boundaries are retained.
3. Converting the color space from BGR (the default format in OpenCV) to RGB, which is the format in which the network is trained.
4. Resize the image to 513x513 pixels to match the network input size.
5. Running the network on the resized image.
6. Convert the resulting output to a "probability" matrix the size of the original image, with values ranging from 0 to 255.
7. Applying a threshold to the matrix so that values above the threshold are set to 1 and values below are set to 0.
8. Filtering areas that are not part of the main road to reduce false results.

The following result was obtained:

Figure 27 - DeepLabV3+ on frame 40

c. The integration between the prediction image generated by the network and the point cloud is done as follows: For each point in the point cloud, the prediction value from the network at that point is checked. If the value is 1, a red circle is drawn. If the value is 0, the original color of the point is used, as it appears in the image before being associated with the road. The resulting image can be seen in figure 28.

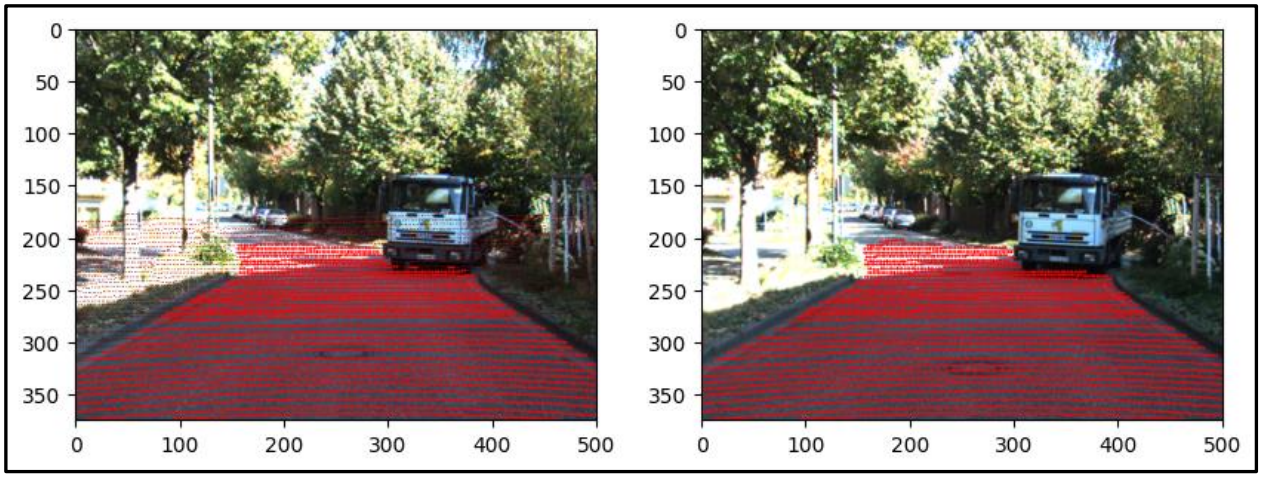

Figure 28 - point cloud only the points that correspond to the road frame 40

3.

a. In this section, the plane representing the road is estimated based on the points in the point cloud that have already been associated with the road using the DeepLabV3+ network. This allows for the inclusion of additional points that lie on the same plane but may not be visible in the camera image.

b.

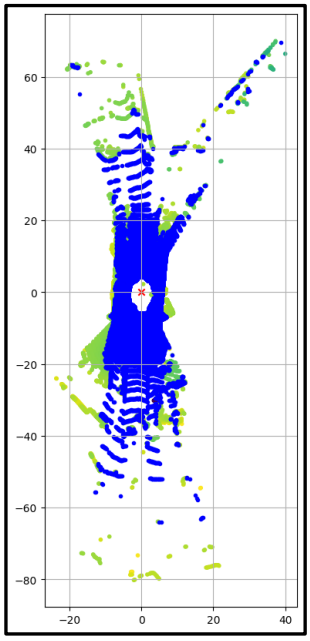

Figure 29 - Drivable path prediction using plane model from LiDAR point clouds frame 40

4.

c. Work Process Description:

1. Loading the image and point cloud data of the current frame.
2. Filtering points within a distance less than 2.5 meters from the sensor and discard points with heights higher than the sensor's height. This filtering is practical as higher points are less relevant as obstacles for the vehicle.
3. Coordinate the point cloud from lidar's coordinate system to camera 2's coordinate system using the extrinsic calibration matrix.
4. Cropping and preparing images for network processing
5. Running the neural network on the image.
6. Estimating road plane using RANSAC model.
7. Assigning nearby points to the road based on a specified threshold.
8. Projecting point cloud excluding non-road points to the camera coordinate system, achieved by multiplying with the inverse intrinsic and extrinsic matrices to transform from camera to lidar and then back from lidar to IMU.
9. Calculating vehicle speed and accumulated yaw angle change since the previous frame.
10. Updating the Occupancy Grid Map (OGM) with the current scan.
11. Applying the resulting matrix from the current scan.

d.

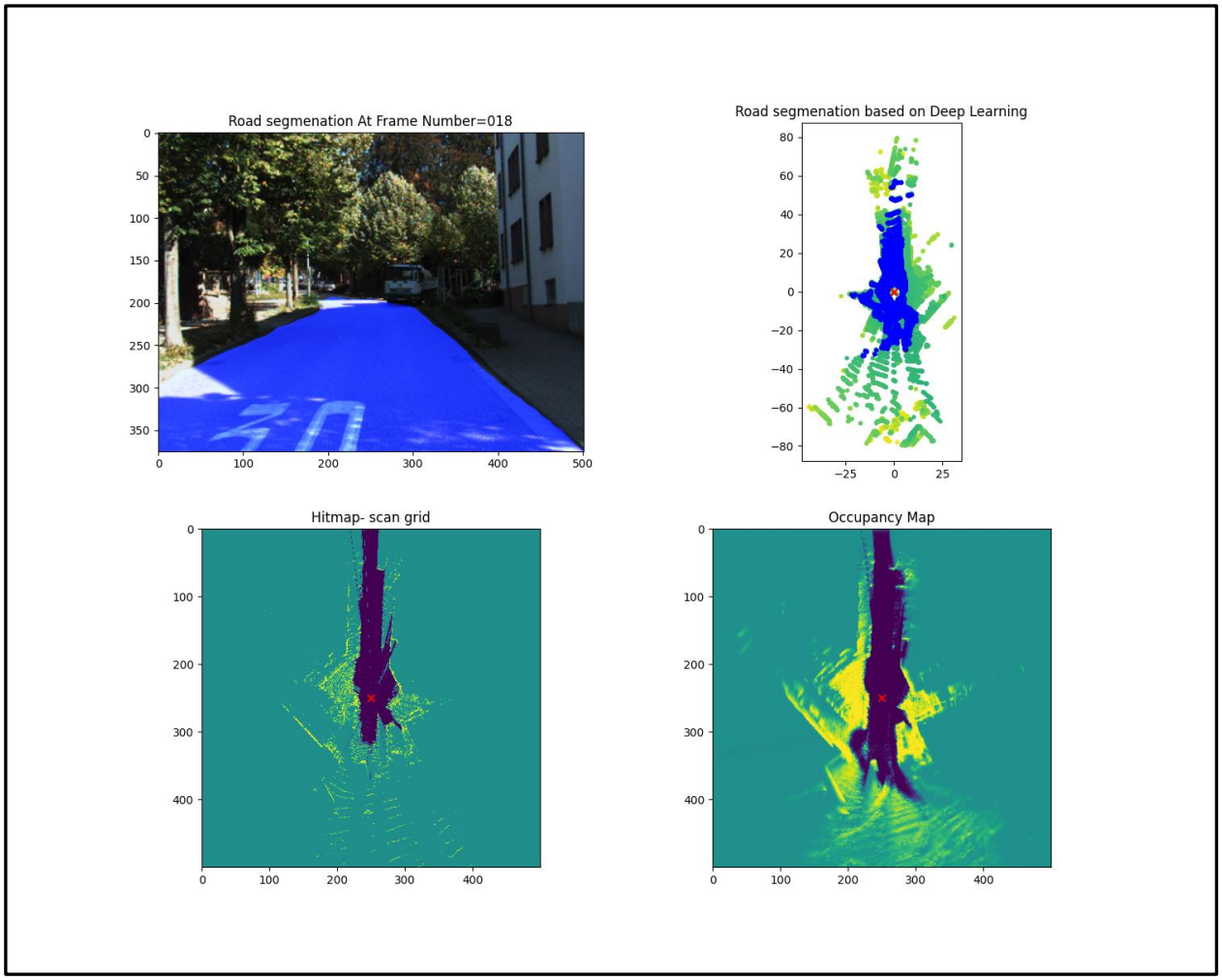

Figure 30 - Frame 18 of the occupancy map (using DeepLabV3+ network)

5.

a. In the naive approach, the LiDAR generates a 360-degree map, continuously adding and reinforcing points with each frame. This method allows for a comprehensive understanding of the environment, including the area behind the vehicle, which is crucial for maneuvers such as reversing. However, this approach requires processing a large amount of data, as it considers the entire 360-degree field of view.
In contrast, the deep learning approach focuses only on the area in front of the vehicle. This results in a more blurred map for the area behind the vehicle, as the accumulated map does not account for objects that may be present there, such as other vehicles. Although this approach processes less data by limiting its scope to the forward-facing view, it produces a more refined segmentation of the drivable path and effectively separates the road from the sidewalk. This targeted processing enhances the accuracy of road navigation while reducing computational complexity.

   Ultimately, while the naive approach provides a more holistic view of the surroundings, the deep learning method offers improved path differentiation and obstacle identification in the vehicle's forward trajectory.

b. Here are some good examples:

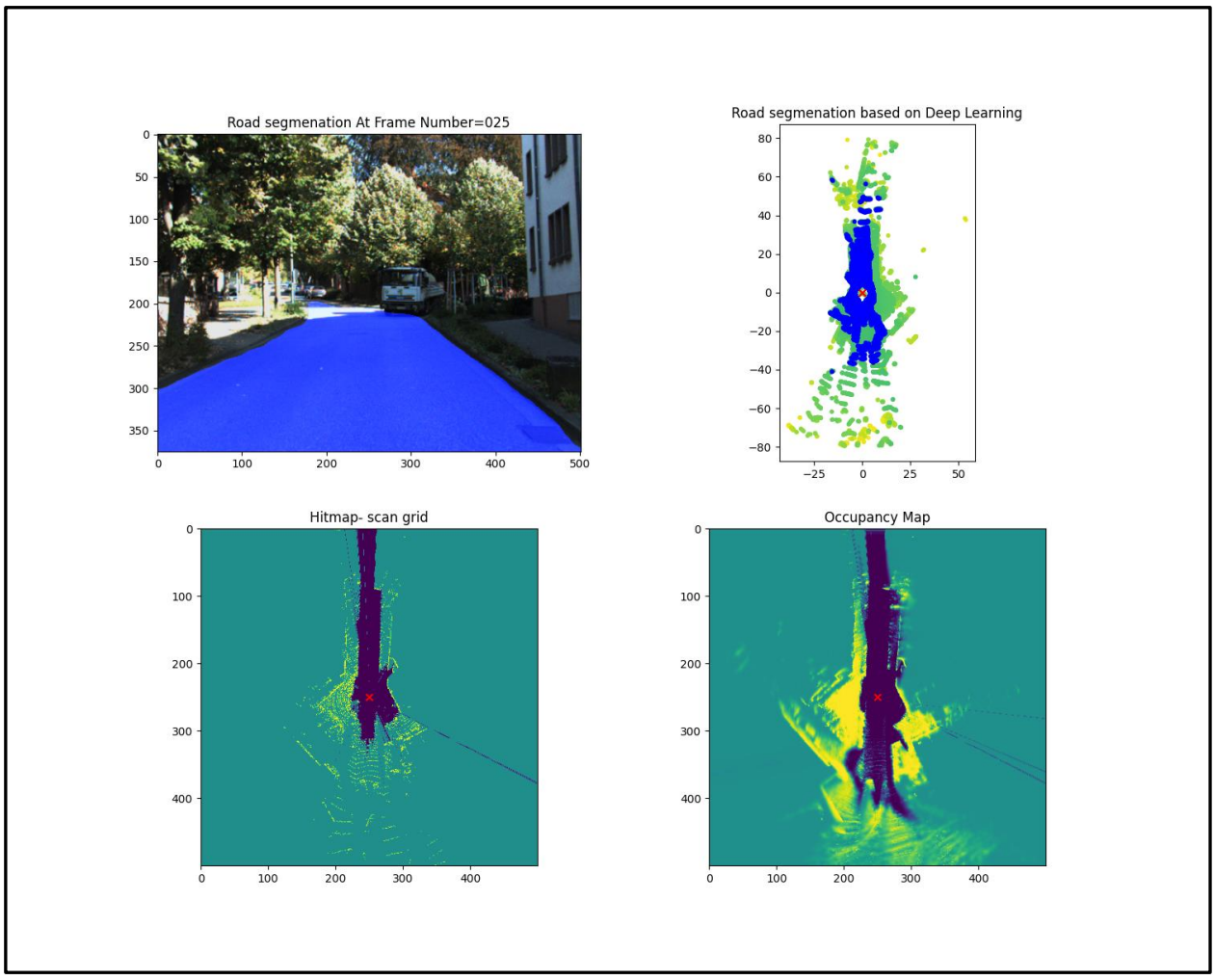

Figure 31 - Frame 25 of the occupancy map (using DeepLabV3+ network)

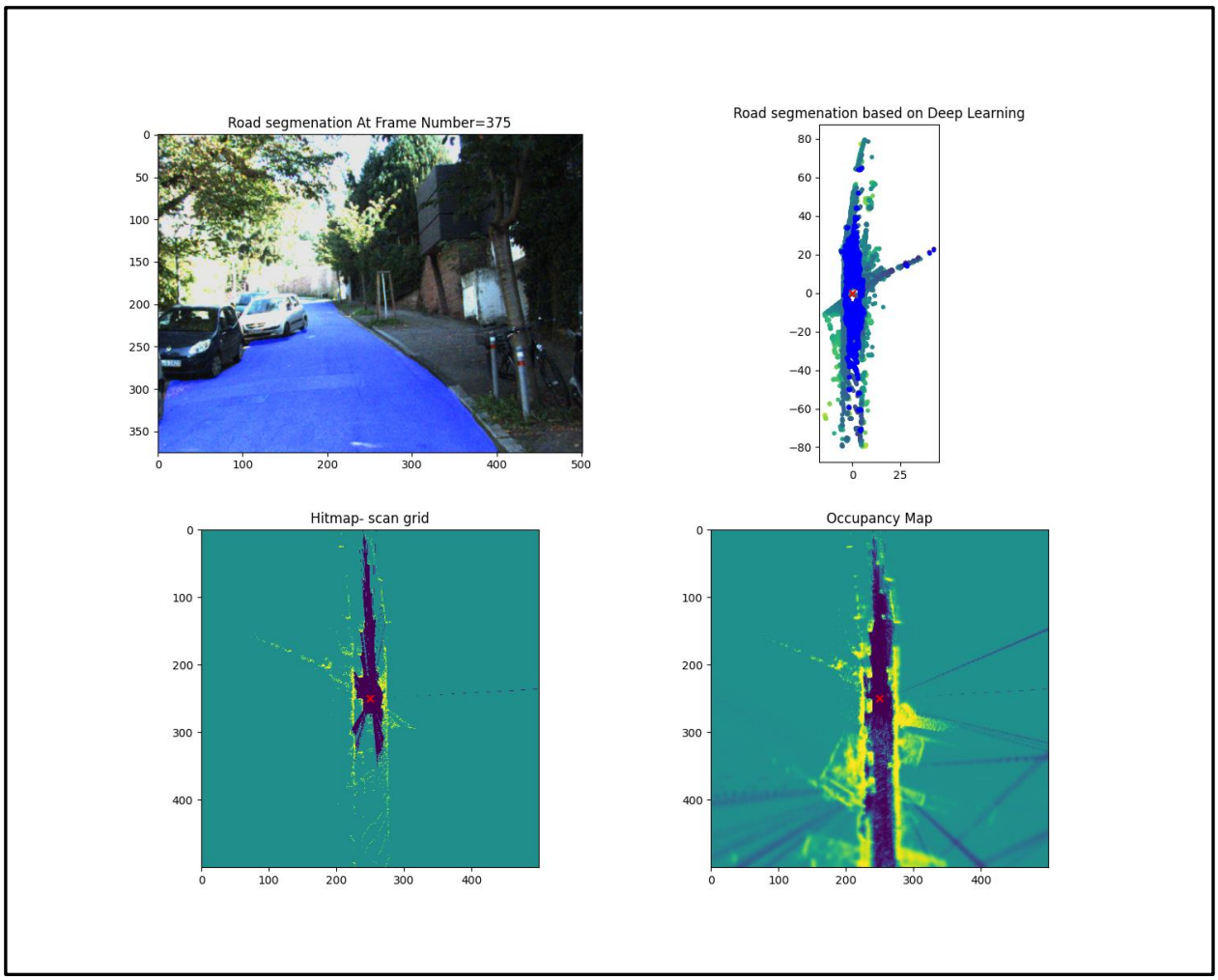

Figure 32 - Frame 375 of the occupancy map (using DeepLabV3+ network)

In these examples, we see the road marked in blue on the image, separated very clearly. The OGM map is hardly blurred and successfully delineates the curb, which is a low curb allowing access to private home driveways. In contrast, the naive method failed to define clear boundaries for the curb in the same areas.

Here are some bad examples:

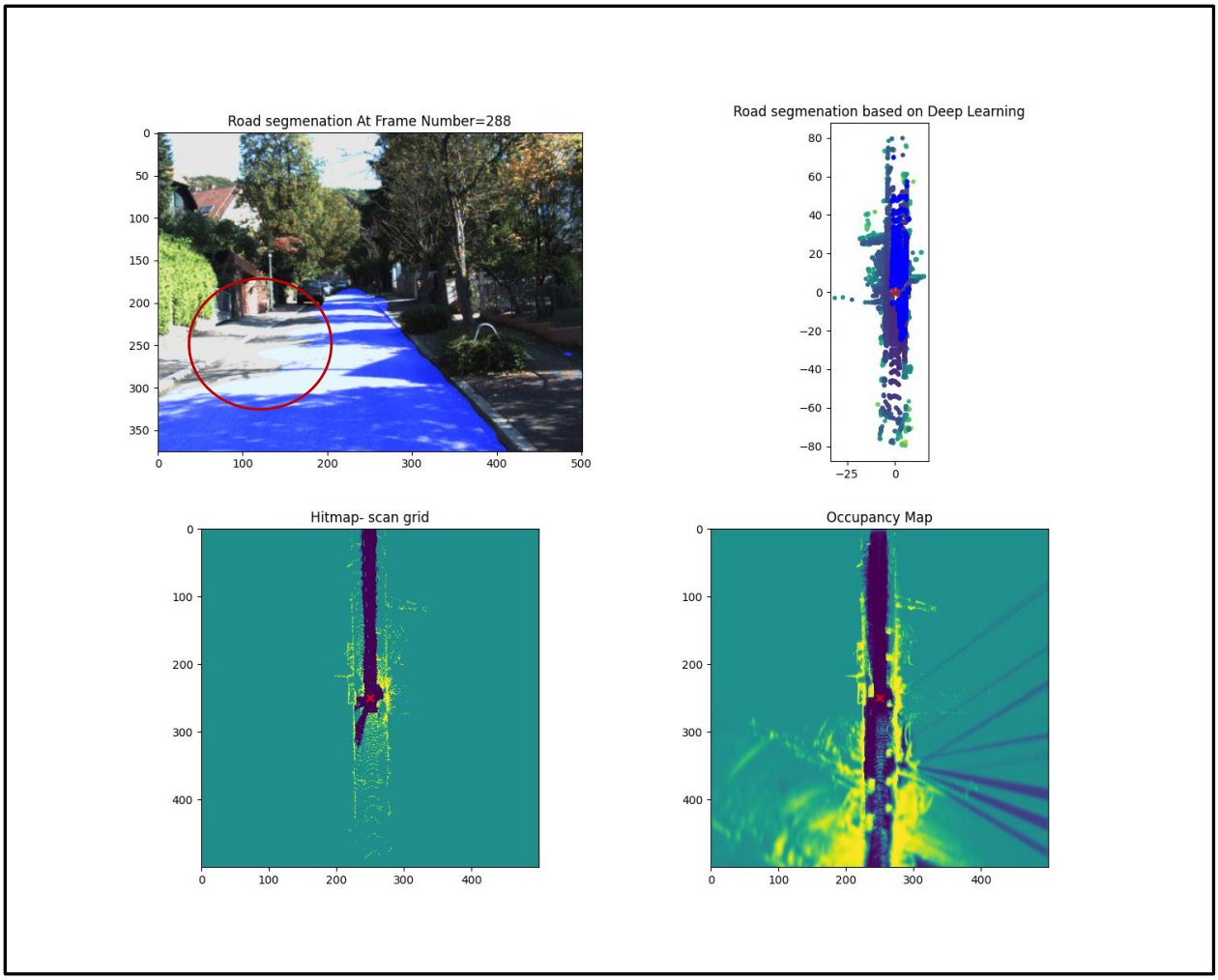

Figure 33 - Frame 288 of the occupancy map (using DeepLabV3+ network)

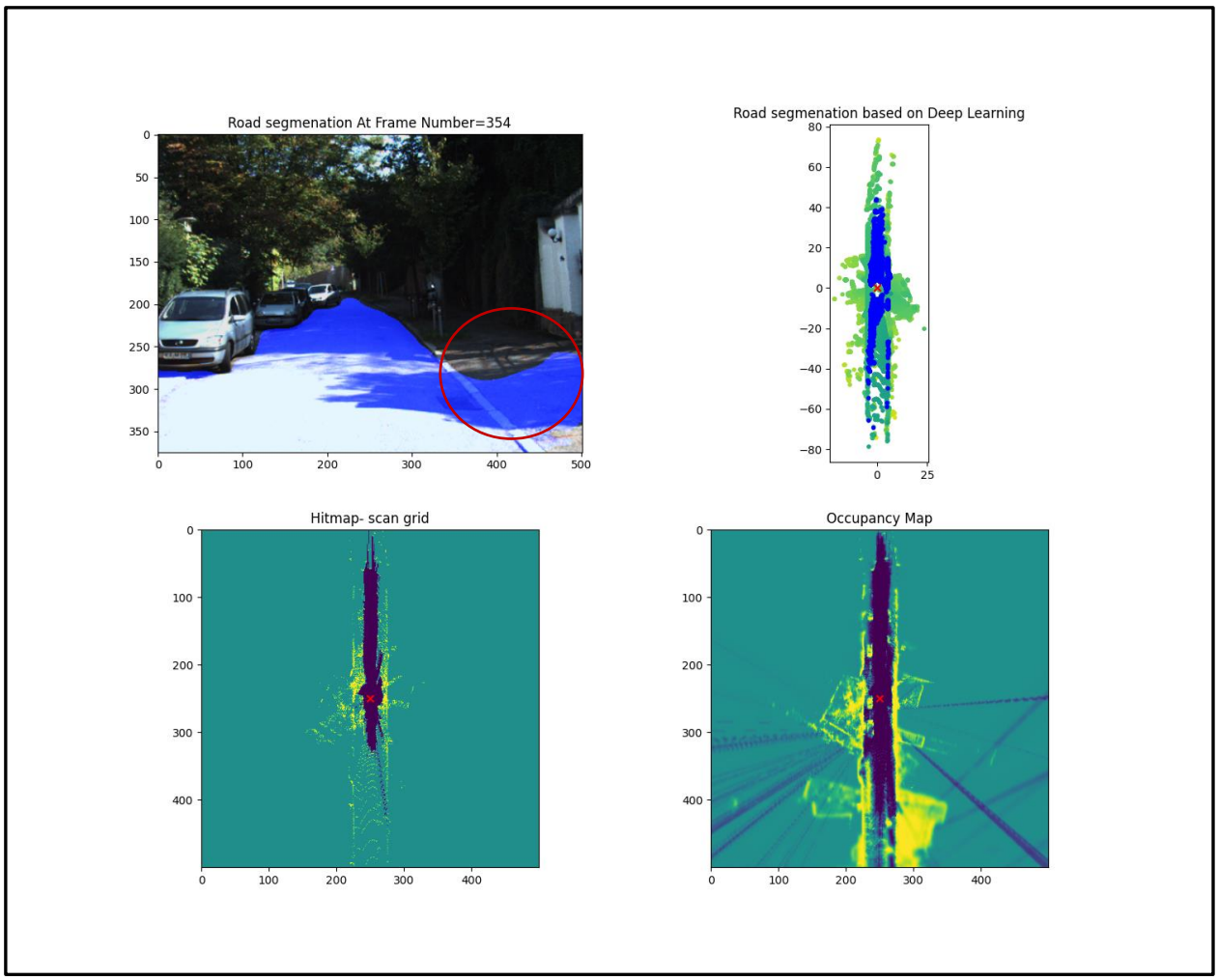

Figure 34 - Frame 354 of the occupancy map (using DeepLabV3+ network)

In these examples, we observe a missed detection of the road marking in blue on the map. It is evident where sunlight interfered with the LiDAR, affecting its ability to classify the road accurately. Additionally, in another example, part of the sidewalk leading to a driveway is marked. Although the classification is technically correct, it lacks the clarity one would expect. This might indicate that the dataset requires slight adjustments to the height offsets due to the low curb.

c. Comparing deep learning with naive mapping, pros and cons:

Pros of Mapping Based on Deep Learning:

- **Accuracy and Generalization:** Deep learning models can learn complex patterns and features from data, potentially leading to more accurate mappings. They can generalize well to diverse environments and conditions, making them robust in various scenarios.

- **Automation:** Once trained, deep learning models can automate the mapping process. They can handle large volumes of data efficiently and continuously improve with more data and fine-tuning, reducing the need for manual intervention.

- **Feature Learning:** Deep learning models can automatically extract relevant features from raw sensor data, which can be challenging to define manually. This ability to learn features can enhance the understanding and representation of the environment.

Cons of Mapping Based on Deep Learning:

- **Complexity:** Deep learning models are often complex and require significant computational resources for training and inference. They may also involve complex architectures that are difficult to interpret and debug.

- **Data Dependency**: Deep learning models require large amounts of labeled data for training, which can be costly and time-consuming to collect and annotate. The quality and diversity of the data directly influence the model's performance.

- **The area** where the vehicle has traveled becomes cluttered in the OGM image, making it appear blocked by obstacles, even if this is not necessarily the case in reality. If the vehicle wants to reverse, it may mistakenly assume it cannot because the OGM indicates the path is obstructed.

d. The algorithmic challenges of segmenting offroad roads:

- **Terrain Variability and Lack of Clear Boundaries:** Off-road environments often lack well defined road boundaries or consistent visual cues that distinguish roads from surrounding terrain. This variability makes it challenging for segmentation algorithms to accurately delineate road areas from non-road areas.

- **Handling Complex Backgrounds and Clutter:** Off-road environments frequently contain complex backgrounds such as rocks, vegetation, puddles, and uneven surfaces. These elements can create clutter in sensor data (e.g., LiDAR or camera images), making it difficult for segmentation algorithms to distinguish between road surfaces and surrounding objects.

Appendix

Attached is a directory named "Figures" containing all the images added to the report.

The dataset from Section B was analyzed using 550 frames due to data processing issues that necessitated the reduction in the number of frames processed, This reduction was implemented to mitigate complications encountered during data processing and frame extraction.

The corresponding animation for part B, Question 2 is labeled as "occupancy_map_video_1".

The corresponding animation for part B, Question 3 is labeled as "occupancy_map_video_2".

The corresponding animation for part C, Question 4 is labeled as "occupancy_map_video_3".