Signal Processing Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
https://sps.ewi.tudelft.nl/

SPS-2023-00

M.Sc. Thesis

LiDAR and Radar-Based Occupancy Grid Mapping for Autonomous Driving Exploiting Clustered Sparsity

Çağan Önen

Abstract

Occupancy grid maps are fundamental to autonomous driving algorithms, offering insights into obstacle distribution and free space within an environment. These maps are used for safe navigation and decision-making in self-driving applications, forming a crucial component of the automotive perception framework. An occupancy map is a discretized representation of a chosen environment that is constructed using point cloud information obtained from sensor modalities like LiDAR and radar. In this project, we formulate the problem of estimating the occupancy grid map using sensor point cloud data as a sparse binary occupancy value reconstruction problem. We utilize the inherent sparsity of occupancy grid maps commonly encountered in automotive scenarios. Besides, the spatial dependencies between the grid cells are exploited to provide a better reconstruction of the boundaries of the objects inside the range of the map and to suppress the false alarms emerging from the reflections coming from the road. To address sparsity and spatial correlation jointly, we propose an occupancy grid estimation method that is based on pattern-coupled sparse Bayesian learning. The proposed method shows enhanced detection capabilities compared to two benchmark methods, based on qualitative and quantitative performance evaluation with scenes from the automotive datasets nuScenes and RADIal.

LiDAR and Radar-Based Occupancy Grid Mapping for Autonomous Driving Exploiting Clustered Sparsity

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Electrical Engineering

by

Çağan Önen
born in Ankara, Turkey

This work was performed in:

Signal Processing Systems Group
Department of Microelectronics
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Microelectronics

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics, and Computer Science for acceptance a thesis entitled **"LiDAR and Radar-Based Occupancy Grid Mapping for Autonomous Driving Exploiting Clustered Sparsity"** by **Çağan Önen** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 22/08/2023

Chairman: _____
prof.dr.ir. A.J. van der Veen

Advisors: _____
dr. ir. G. Joseph

_____
dr. ir. A.Pandharipande

Committee Members: _____
dr. ir. N.J. Myers

_____

Abstract

Occupancy grid maps are fundamental to autonomous driving algorithms, offering insights into obstacle distribution and free space within an environment. These maps are used for safe navigation and decision-making in self-driving applications, forming a crucial component of the automotive perception framework. An occupancy map is a discretized representation of a chosen environment that is constructed using point cloud information obtained from sensor modalities like LiDAR and radar. In this project, we formulate the problem of estimating the occupancy grid map using sensor point cloud data as a sparse binary occupancy value reconstruction problem. We utilize the inherent sparsity of occupancy grid maps commonly encountered in automotive scenarios. Besides, the spatial dependencies between the grid cells are exploited to provide a better reconstruction of the boundaries of the objects inside the range of the map and to suppress the false alarms emerging from the reflections coming from the road. To address sparsity and spatial correlation jointly, we propose an occupancy grid estimation method that is based on pattern-coupled sparse Bayesian learning. The proposed method shows enhanced detection capabilities compared to two benchmark methods, based on qualitative and quantitative performance evaluation with scenes from the automotive datasets nuScenes and RADIal.

Acknowledgments

I would like to thank my supervisors Geethu Joseph and Ashish Pandharipande for their assistance and guidance in completing the thesis. Also, I would like to thank Nitin Myers for his assistance and support throughout the project. Finally, I want to thank my dear friends and family for their endless moral support.

Çağan Önen
Delft, The Netherlands
22/08/2023

Contents

List of Figures

List of Tables

1 Introduction

The automotive industry has witnessed a remarkable evolution in recent years, with rapid advancements in perception technologies transforming the way we interact with and drive vehicles. At the forefront of this revolution are two key technologies: Advanced Driver Assistance Systems (ADAS) and Autonomous Driving Systems (ADS).

ADAS includes a range of technologies that assist drivers in making safer and more informed decisions while driving and parking. Under the umbrella of ADAS, the systems are grouped under 6 levels ranging from level 0 to level 5 [1]. Level 0 systems only provide warnings feedback to the driver about potential hazards without controlling the car [2]. The last and most advanced level of autonomy is level 5, which is also referred to as ADS. At this level, the vehicle can perform all driving tasks without driver intervention [3]. There is no distinction between the driver and the passengers, and the interface between the vehicle and the passengers is kept minimal [2]. The vehicles must navigate through complex and dynamic environments without any human intervention or supervision.

Taking the whole responsibility from the driver and giving it to the vehicle seems scary and raises some concerns in society, but given the robustness of the state-of-art self-driving vehicles, ADS offers to increase safety on the roads [3]. To guarantee safety, the autonomous vehicle must obtain an accurate perception and understanding of the surrounding environment [4, 5]. Therefore, automotive perception is one of the most important modules in ADS. The automotive perception systems, which include cameras, light detection and ranging (LiDAR), radar, and ultrasonic sensors capture real-time data about the vehicle's surroundings. Then, this raw data is processed and used by the vehicle to generate an understanding of its environment, allowing it to make informed decisions, detect potential hazards, and execute precise maneuvers for safe and efficient autonomous navigation on the road. Therefore, highly accurate, robust perception algorithms are needed to decrease the number of traffic accidents and casualties via autonomous vehicles.

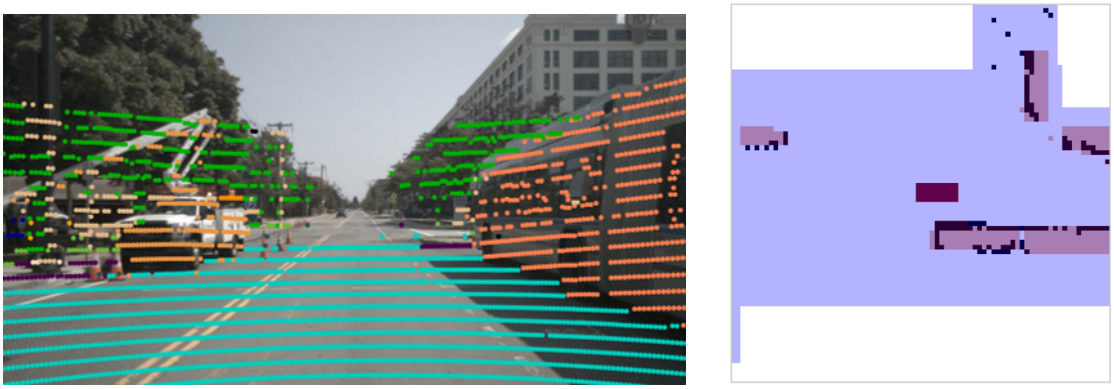
One of the common algorithms used for automotive perception is occupancy grid mapping which provides a grid map representation of the occupancy status of the surrounding space using sensor data [6, 7].

1.1 Occupancy Grid Maps

Occupancy grid mapping entails creating a discretized representation of an environment based on sensor readings, effectively partitioning the chosen area into uniformly sized grid cells [6]. This process is akin to spatial quantization, where the resulting cells, termed grid cells, store the occupancy information. Occupancy is meaningful information and it refers to whether a particular area is occupied by an object or ob-

stacle. Each grid cell is associated with a numerical value denoting the probability of the corresponding area being occupied by objects, whether static, such as buildings, or dynamic, like vehicles or pedestrians. From a probabilistic perspective, the occupancy map can be conceptualized as a binary random field arranged in a uniformly spaced grid, wherein each cell represents a binary random variable indicative of its occupancy state [6]. These occupancy states of the grid cells are calculated using occupancy grid mapping methods.

1.2 Occupancy Grid Mapping Methods



(a) Sample LiDAR Point Cloud [8]

(b) Sample Occupancy Grid Map

Figure 1.1: The occupancy grid mapping algorithms create a discrete environment representation from sensor data. In (a), a LiDAR-generated point cloud is overlaid on the front camera image. Points are labeled and colored by object classification. (b) presents a sample occupancy grid map derived from this LiDAR data. Blue denotes roads, red signifies the ego vehicle, and black marks occupied cells. The vehicle is centered, heading right. Objects' original locations are indicated with red shades.

The occupancy grid mapping algorithms enable the creation of occupancy grid maps from sensor data. The sensor data is mostly in the point cloud format which is a collection of reflection points that are coming from a sensor. Fig. 1.1a demonstrates a sample LiDAR point cloud in which the coordinates of laser hits are labeled with different colors and projected onto the front camera view. The objects the laser hits correspond to are labeled with different colors but in real-life this information is not available. This is obtained via manual annotation to provide a comprehensive representation of the provided points cloud.

The point cloud is fed to occupancy grid mapping algorithms to create a map of the environment. Figs. 1.1a and 1.1b demonstrate a sample occupancy grid mapping scenario in which the two vehicles that are visible on the front view of the vehicle are represented in the sample occupancy grid map (Fig. 1.1b). Among the four vehicles that are labeled with red shades in the sample occupancy map, only two of them are visible in the front camera image. The ego vehicle which is labeled with dark red is traveling toward the right side of the map between the bus and the truck. Therefore, the bus is on the right side of the ego vehicle and it is labeled with the biggest red boundary box in Fig. 1.1b. The black cells that are inside this box indicate the occupied cells

corresponding to the bus. This representation is obtained using LiDAR point, but LiDAR is not the only sensor that is used for occupancy grid map estimation. Various sensors can be used in the estimation process and they all have different advantages over each other.

1.2.1 Sensors Used for Occupancy Grid Mapping

Sensors that can be used for occupancy grid mapping include LiDAR, radar, cameras, sonar, ultrasonic sensors, and infrared sensors. In automotive applications, occupancy grid mapping relies on the utilization of specific sensors that offer distinct capabilities, contributing to safe and reliable navigation.

LiDAR is commonly used in automobiles for occupancy grid mapping and it provides a detailed representation of the environment as demonstrated in Fig. 1.1a. LiDAR sensors operate by emitting laser pulses toward the ground or objects in their field of view. These emitted pulses are then reflected back toward the LiDAR sensor from the surrounding obstacles. Precise timing equipment measures the time it takes for the laser pulses to travel to the target and return, enabling the calculation of very accurate depth information [9]. By emitting thousands of laser pulses per second and using scanning mechanisms, LiDAR creates a dense point cloud representation of the surroundings. The 3D data points (x, y, z coordinates) in the point cloud correspond to specific locations on the ground or objects. In the context of ADAS and ADS, point cloud data provides a detailed representation of the surrounding vehicles, pedestrians, cyclists, and static objects inside or around the road. Scanning lasers are commonly characterized by their relatively high scan rates, high resolution, and lower costs compared to radar systems which make them highly desirable for occupancy grid mapping. It offers a comprehensive and precise understanding of the environment, making it valuable in occupancy mapping, object detection, and SLAM algorithms. However, one of the drawbacks of the LiDAR sensor is that bad weather conditions such as rain, fog, and dust can corrupt the measurements. Moreover, as light can go through transparent objects, LiDAR does not have the ability to detect transparent objects such as glass [10].

Another common sensor used in automobiles is radar. Radar works by emitting radio waves or microwaves, which then travel through the air and reflect from the surrounding objects. The system measures the time it takes for the reflected waves to return, allowing it to calculate the distance, speed, and direction of the objects in its field of view. Radars generally have wider beam widths and lower range resolution compared to LiDAR, resulting in less precise distance measurements and a less detailed representation of the environment. Notwithstanding their inferior resolution, radars are preferred for occupancy grid mapping [11] over LiDAR in certain scenarios due to the specific advantages they offer. Firstly, they are widely used for long-range object detection, making them invaluable for highway driving and adaptive cruise control. By detecting larger objects and providing information about relative speed and distance, radar sensors enhance collision avoidance and situational awareness. Secondly, radar sensors are favored in scenarios that require robustness in adverse weather conditions as they are insensitive to water vapor particles (fog) and dust [10]. Therefore, radar's capability to operate effectively in adverse weather conditions makes it a reliable choice for applications where adverse weather may corrupt LiDAR measurement.

In addition to range sensors that supply point clouds, also cameras can be used for occupancy grid mapping. They provide vital data for road scene understanding, traffic sign recognition, lane detection, and object recognition. The information extracted from cameras is essential for creating a comprehensive occupancy grid map that includes visual information about the environment and surrounding objects. Monocular cameras are unsuitable for occupancy grid mapping due to their inability to provide depth information. However, some existing approaches [12] utilize stereo cameras to address this limitation and effectively tackle the occupancy grid mapping problem.

In this project, our particular focus is on using point cloud information obtained from range sensors, specifically LiDAR and radar as they provide direct distance measurements to objects, enabling accurate representation of the environment and precise mapping of obstacles. With a solid foundation in the role of sensors, our exploration now shifts to the heart of the matter: the algorithms that orchestrate the creation of occupancy grid maps.

1.2.2 Occupancy Grid Map Estimation Algorithms

Before exploring occupancy grid mapping algorithms, it is essential to establish a common notation to ensure clarity and consistency throughout the discussion.

- Let M be the number of points in the point cloud. The matrix $\mathbf{X} \in \mathbb{R}^{M\times 2}$ contains the 2D coordinates of M points, where the location of the mth point is $\mathbf{x}_m = [X[m,1], X[m,2]]^T$, with $X[m,1]$ and $X[m,2]$ denoting the x and y coordinates, respectively. The matrix $\mathbf{X}$ can be represented as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_M^T \end{bmatrix} = \begin{bmatrix} X[1,1] & X[1,2] \\ X[2,1] & X[2,2] \\ \vdots & \vdots \\ X[M,1] & X[M,2] \end{bmatrix}. \tag{1.1}$$

- Let N_x and N_y indicate the number of cells in height and width which makes $N = N_x \times N_y$ cells in total. The matrix $\mathbf{X_c} \in \mathbb{R}^{N\times 2}$ contains the 2D coordinates of the center of N grid cells, where the location of the nth grid cell is $\mathbf{x}_{c,n} = [X_c[n,1], X_c[n,2]]^T$, with $X_c[n,1]$ and $X_c[n,2]$ denoting the x and y coordinates of the center of the nth grid cell, respectively. The matrix $\mathbf{X_c}$ can be represented as:

$$\mathbf{X_c} = \begin{bmatrix} \mathbf{x}_{c,1}^T \\ \mathbf{x}_{c,2}^T \\ \vdots \\ \mathbf{x}_{c,N}^T \end{bmatrix} = \begin{bmatrix} X_c[1,1] & X_c[1,2] \\ X_c[2,1] & X_c[2,2] \\ \vdots & \vdots \\ X_c[N,1] & X_c[N,2] \end{bmatrix}. \tag{1.2}$$

- We use $\mathbf{g} \in \mathbb{R}^N$ to donate the binary occupancy grid map:

$$\mathbf{g} = \begin{bmatrix} g[1] & g[2] & \cdots & g[N] \end{bmatrix}^T. \tag{1.3}$$

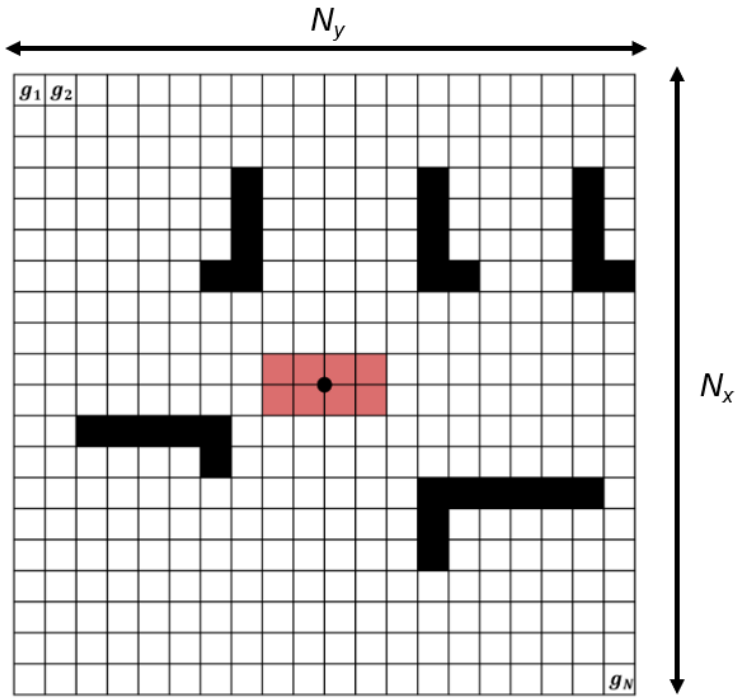


Figure 1.2: Representation of a sample binary occupancy grid map. The black cells are the occupied cells and the white cells are the free ones. The ego vehicle is stationed at the origin and its coverage is shown in red.

Each $g[n]$ holds a binary occupancy value showing the nth cell's occupancy state such that $g[n] = 1$ for occupied and $g[n] = 0$ for free. Then, the notation $p(g[n])$ is used to refer to $p(g[n] = 1)$ which is the probability that the nth grid cell is occupied. Similarly, the probability that it is free is $p(g[n] = 0)$ which is equal to $1 - p(g[n])$.

- The occupancy probabilities of the grid cells are represented by $\mathbf{f} \in \mathbb{R}^N$:

$$\mathbf{f} = \begin{bmatrix} f[1] & f[2] & \cdots & f[N] \end{bmatrix}^T, \tag{1.4}$$

where

$$f[n] = p(g[n] = 1). \tag{1.5}$$

In occupancy grid mapping, the key objective is to estimate the posterior probability distribution $\mathbf{f} = p(\mathbf{g}|\mathbf{X}, \mathbf{p})$, given the sensor data $\mathbf{X}$ collected from the robot's perception system and the sensor location $\mathbf{p}$,and calculate the estimated map $\hat{\mathbf{f}} \in \mathbb{R}^N$. In our case, we are not performing localization, so we always assume that the location of the sensor is known. Therefore, the notation $p(\mathbf{g}|\mathbf{X})$ is used to donate the posterior distribution for the sake of simplicity.

After obtaining $\hat{\mathbf{f}}$, the estimate of the binary occupancy grid map representation $\hat{\mathbf{g}}$ is calculated by applying a threshold to $\hat{\mathbf{f}}$:

$$\hat{g}[n] = \begin{cases} 0, & \hat{f}[n] < \eta_{\text{th}} \\ 1, & \hat{f}[n] \geq \eta_{\text{th}} \end{cases}. \tag{1.6}$$

Fig 1.2 is a representation of a sample binary occupancy grid map. The elements of vector $\mathbf{g}$ are represented using the colors black and white, where black corresponds to $g[n] = 1$ and white corresponds to $g[n] = 0$. To estimate and create this binary representation from $\mathbf{X}$ there are various approaches proposed in the literature.

The diversity in occupancy grid mapping algorithms primarily lies in the approach employed to translate sensor readings into occupancy probabilities. When considering the methodologies involved, the most fundamental approach is the Binary Bayesian Filter Using the Log-Odds Ratio Formulation [13], which stands as the earliest and most straightforward technique in this domain. Despite the emergence of more sophisticated techniques, the simplicity of this method makes it a valuable tool in occupancy mapping research. There are also kernel-based approaches such as Gaussian Process Occupancy Mapping (GPOM) [14] which aims to model and capture the spatial dependencies between the grid cells via the usage of kernel functions. This method allows for more accurate estimation of occupancy probabilities and better incorporation of spatial information, at the expense of increased computational complexity. Recently, an improved version of this approach, Bayesian Generalized Kernel-based mapping (BGK) in [15], addressed the complexity issue and improve the accuracy of occupancy estimates by modeling spatial dependencies using a generalized kernel function. In this section, the application of these methods to two-dimensional occupancy grid mapping is explained.

1.2.2.1 Binary Bayesian Filter Using the Log-Odds Ratio Formulation

Binary Bayesian Filter with the Log-Odds Ratio Formulation [13] is a fundamental and effective occupancy grid mapping algorithm. Here, the aim is to estimate the posterior probability distribution $\mathbf{f}$ given the sensor data $\mathbf{X}$ iteratively using a binary filter. However, the problem is the dimension of the $\mathbf{g}$ vector since the map has 2^N possible states. The number of grid cells in a map can be hundreds of thousands which makes this posterior estimation intractable [13]. Therefore, instead of utilizing an exact solution, a suboptimal solution is proposed which makes two assumptions to simplify the problem: Independence assumption and static world assumption. The independence assumption states that the occupancy state of a grid cell does not influence the occupancy states of its neighboring cells. In mathematical terms, this assumption implies that the occupancy states of the individual grid cells are conditionally independent given the sensor measurements. In reality, assuming independence among occupied cells is not logically justified, as many obstacles span multiple cells, and the occupancy status of cells within the same object tends to be correlated [16], [17]. On the other hand, it is a strong and convenient assumption that divides the map estimation problem into N independent binary estimation problems.

The static world assumption states that the past sensor measurements are conditionally independent given the map $\mathbf{f}$ at any time t. This assumption is valid for static maps with non-changing environments. However, when it is coupled with the independence assumption it becomes a strong and incorrect assumption [16], [17]. Instead of considering conditional independence given the full map knowledge, conditional independence given only a map cell is considered.

This method leverages Bayes' theorem to recursively update the occupancy grid as

new sensor measurements become available. The log-odds ratio formulation is utilized to represent the occupancy probabilities in a log-likelihood domain, which provides computational advantages and numerical stability. By applying the binary Bayes filter defined in [13] and with the underlying assumptions in place, the update rule

$$\ell_m[n] = \ell_{m-1}[n] + \log \frac{p(g[n] \mid \mathbf{x}_m)}{1 - p(g[n] \mid \mathbf{x}_m)} - \ell_0. \tag{1.7}$$

is obtained to incorporate mth measurement in the point cloud to the map and update the states of the grid cells accordingly. Here, $\ell_m \in \mathbb{R}^N$ denotes log-odds representation of the occupancy probabilities of the N cells after processing the mth measurement. The first term in (1.7) is the previous state of the corresponding cell and the last term ℓ_0 is the initial state of the cells. The second term is known as the "inverse sensor model (ISM)" and it is used to update the states of the corresponding grid cells due to mth measurement. This model assumes the shape of a LiDAR beam as a cone that is called "the measurement cone". The shape of the cone is characterized using the beam-width Ω and the obstacle thickness Δ which are illustrated in Fig. 1.3. Following this structure, the measurement cone is divided into two regions: Region 1, which includes the dotted cells, and Region 2, which includes the dashed cells. The algorithm assigns to the cells in Region 1 high occupancy values ℓ_{occ} and to the cells in Region 2 lower occupancy values ℓ_{free}.

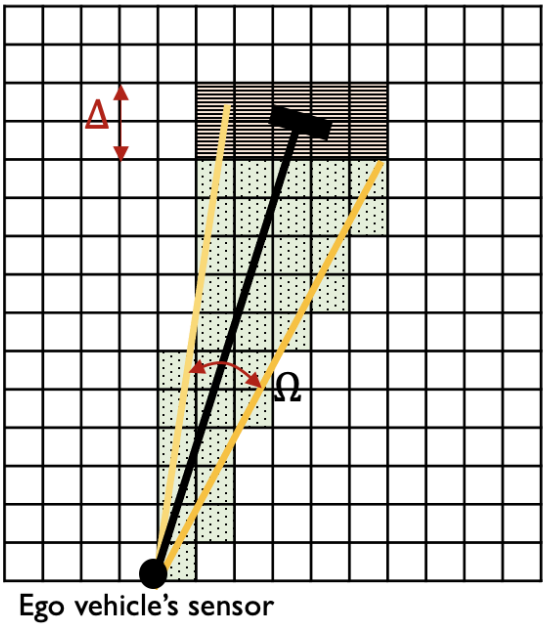


Figure 1.3: Illustration of the sensor model used in the study. The figure depicts the measurement cone, whose shape is determined by the parameters Ω and Δ.

After processing M measurements and updating the states of the cells using the defined sensor model, the log-odds representation of the environment is obtained. Then, the transformation

$$\hat{f}[n] = 1 - \frac{1}{1 + \exp \ell[n]} \tag{1.8}$$

is applied to go back to probability domain and obtain $\hat{\mathbf{f}}$. The binary occupancy grid map $\hat{\mathbf{g}}$ is calculated using (1.6) and the final occupancy grid map is obtained. The overall procedure is summarized in Algorithm 1.

To conclude, Binary Bayesian Filter Using the Log-Odds Ratio Formulation [16] is a fundamental mapping technique that is popular due to being straightforward and computationally efficient. On the other hand, it is important to highlight its drawbacks, primarily rooted in the two assumptions: independence assumption and static world assumption. Although these assumptions make both the derivation and implementation of the algorithm very simple, they lead to neglecting the spatial correlations between the grid cells and restricting the algorithm's ability to model obstacles accurately in complex environments. One way to utilize the spatial correlations between the grid cells is by using kernel functions to model their dependencies which is the fundamental of the Gaussian process occupancy mapping algorithm [14].

Algorithm 1 Occupancy Grid Mapping Using Binary Bayesian Filter

Input: Reflection point coordinates $\{\mathbf{x}_m\}_{m=1}^{M}$
Parameters: Beam width Ω, object thickness Δ, ISM parameters ℓ_{free}, ℓ_{occ} and ℓ_0
Output: Binary occupancy grid map: $\hat{\mathbf{g}}$

For each measurement m **do**
  For each grid cell n **do**
    if n is inside the conical beam of $\mathbf{x}_m$
      $\ell_m[n] \leftarrow \ell_{m-1}[n] + \ell_{\text{free}} - \ell_0$
    if n is one of the conical beam's terminal points
      $\ell_m[n] \leftarrow \ell_{m-1}[n] + \ell_{\text{occ}} - \ell_0$

Compute $\hat{f}[n]$ using (1.8) for each grid cell
Compute $\hat{\mathbf{g}}$ from (1.6)

1.2.2.2 Gaussian Process Occupancy Mapping

Gaussian process occupancy mapping (GPOM) [14] is a probabilistic occupancy mapping technique that uses Gaussian processes to model the occupancy of each cell in an occupancy map. The basic idea of GPOM is to model the occupancy of each grid cell as a function of its location and sensor measurements, using a Gaussian process (GP). GP is a flexible probabilistic model that can be used to make predictions about a function. In essence, a GP is a distribution over functions, where any finite set of function values has a joint Gaussian distribution. This joint Gaussian assumption enables the algorithm to capture complex correlations and uncertainties in the data [14].

In the context of GPOM, a GP $f(.)$ is employed to model occupancy probabilities for spatial locations, with $f(\mathbf{x}_{c,n})$ representing the occupancy value of the nth cell. To compute $f(\mathbf{x}_{c,n})$ for a given cell, a training set $\mathbf{X_T}$ is formed by combining reflection points from point cloud data with acquired free space points. For every reflection point, a free space line is defined which is the line between the sensor and the reflection point. The closest point on the free space line to $\mathbf{x}_{c,n}$ is added to the training set as a free space

point. Moreover, the vector $\mathbf{y} \in \mathbb{R}^{M\times 2}$ is crafted which serves as the noisy indicators of occupancy states at training points. It employs a value of -1 for free space points and 1 for reflection points. This GP $f(.)$ is characterized by its mean $\mu(.)$ and covariance functions $k(.,.)$, wherein $k(\mathbf{X_T}, \mathbf{X_T})$ denotes the covariance among training points and the vector $k(\mathbf{x_{c,n}}, \mathbf{X_T})$ signifies the covariance between the training set and $\mathbf{x}_{c,n}$. In GPOM, the neural network covariance function is chosen as the kernel function for its ability to capture complex and non-linear relationships in occupancy data. According to this model, the central assumption is the joint Gaussian distribution of points in the training set $\mathbf{X_T}$ and $\mathbf{x}_{c,n}$, which gives

$$f(\mathbf{x}_{c,n}) = N(\mu, \sigma) \tag{1.9}$$

where

$$\hat{f}[n] = \mu(\mathbf{x}_{c,n}) = k(\mathbf{x}_{c,n}, \mathbf{X_T})(k(\mathbf{X_T}, \mathbf{X_T}) + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}, \tag{1.10}$$

$$\sigma(\mathbf{x}_{c,n}) = k(\mathbf{x}_{c,n}, \mathbf{x}_{c,n}) - k(\mathbf{x}_{c,n}, \mathbf{X_T})(k(\mathbf{X_T}, \mathbf{X_T}) + \sigma_n^2\mathbf{I})^{-1}k(\mathbf{X_T}, \mathbf{x}_{c,n}). \tag{1.11}$$

Here, σ_n^2 is the measurement noise. Traditionally, $\mu(\mathbf{x}_{c,n})$ serves as the MAP estimate for occupancy values, but in GPOM one additional step is taken to perform classification.

The "Probabilistic Least Squares" approach from [14], detailed in [18] is employed to transform the predicted mean and variance through the sigmoid function $\Phi(.)$, resulting in:

$$\tilde{f}[n] = p(g[n]|\mathbf{X_T}, \mathbf{x}_{c,n}) = \Phi\left(\frac{\theta_1\mu(\mathbf{x}_{c,n}) + \theta_2}{1 + \theta_1^2\sigma^2(\mathbf{x}_{c,n})}\right). \tag{1.12}$$

In this context, the hyperparameters of the sigmoid function, denoted as θ_1 and θ_2, can be learned offline during a prior training phase, as mentioned in [14]. The final binary occupancy map representation $\hat{\mathbf{g}}$ can be obtained by applying a threshold to $\tilde{f}[n]$. The overall estimation procedure is summarized in Algorithm 2.

In conclusion, GPOM aims to derive the occupancy probability of the queried points. The primary contribution [14] lies in its adoption of GP for addressing the occupancy mapping problem and leveraging the spatial correlation through kernel functions. Employing a large kernel in point cloud analysis enables accurate predictions of occupancy states for any location in the map area, including unobserved regions. However, it also introduces computational complexity. When making predictions for each query point $\mathbf{x}_{c,n}$, the covariance between the query point and all measurements in the point cloud needs to be updated using the chosen kernel function. To create a complete occupancy map, this prediction process must be repeated N times, which can lead to significant computational demands. Moreover, the need for a training phase is another downside of GPOM which increases the overall complexity of the algorithm. During the training phase, the model must optimize the hyperparameters to best fit the sensor data. To address the complexity issues and make the methodology more feasible for driving scenarios or other applications where lower complexity is essential, an approach is proposed in [15]. This alternative method involves employing a smaller kernel function and adopting a different inference approach to reduce the computational burden while still providing reliable predictions.

Algorithm 2 Occupancy Grid Mapping Using GPOM

Input: Reflection point coordinates $\{\mathbf{x}_m\}_{m=1}^{M}$, grid center Coordinates $\mathbf{X_c}$
Parameters: Hyperprameters for the chosen kernel function and hyperparameters for classification function θ_1, θ_2
Output: Binary occupancy grid map: $\hat{\mathbf{g}}$

For each grid cell n **do**
  For each measurement m **do**
    $\mathbf{x}_{\text{free,m}} \leftarrow$ the closest point on the free space line to the center of nth grid cell
    $(2m-1)$th row of $\mathbf{X_T} \leftarrow \mathbf{x}_{\text{free,m}}$
    $(2m)$th row of $\mathbf{X_T} \leftarrow \mathbf{x}_m$
    $\mathbf{y}[2m-1] \leftarrow -1$
    $\mathbf{y}[2m] \leftarrow 1$
  Compute $\mu(\mathbf{x}_{c,n})$ and $\sigma(\mathbf{x}_{c,n})$ from (1.10)
  Compute $\tilde{\mathbf{f}}[n]$ from (1.12)

Compute $\hat{\mathbf{g}}$ from (1.6)

1.2.2.3 Bayesian General Kernel Inference

Bayesian General Kernel Inference-based occupancy mapping [15] is one of the alternative methods to GPOM. Similar to GPOM, this method tries to predict the occupancy probabilities of the chosen query points by considering the correlation between the grid cells. This method utilizes a nonparametric Bayesian generalized kernel inference [19] for mapping applications to create a simple algorithm that offers accurate predictions for occupancy states.

In this framework, $f[n]$ is represented as a realization of a Beta-distributed random variable which is characterized by the hyperparameters $\alpha[n]$ and $\beta[n]$. Here, the goal is to estimate this parameter and obtain the probability of occupancy of nth grid cell. To achieve this, a nonparametric Bayesian inference model for exponential families [19] is employed. According to this model, the estimated mean and variance of $f[n]$ can be expressed as

$$\mu(\mathbf{x}_{c,n}) = \frac{\alpha[n]}{\alpha[n]+\beta[n]}, \tag{1.13}$$

$$\sigma(\mathbf{x}_{c,n}) = \frac{\alpha[n]\beta[n]}{(\alpha[n]+\beta[n])^2(\alpha[n]+\beta[n]+1)}. \tag{1.14}$$

Once a training set $\mathbf{X_T}$ is extracted from a point cloud, the hyperparameters $\alpha[n]$ and $\beta[n]$ are calculated as follows:

$$\alpha[n] = \alpha_0 + \sum_{m=1}^{\tilde{M}} k\left(\mathbf{X_T}[m], \mathbf{x}_{c,n}\right) y[m] \tag{1.15}$$

$$\beta[n] = \beta_0 + \sum_{m=1}^{\tilde{M}} k\left(\mathbf{X_T}[m], \mathbf{x}_{c,n}\right) (1 - y[m]) \tag{1.16}$$

where $\tilde{M}$ is the size of the training set, α_0 and β_0 are the initial values for the hyperparameters. $k\left(\mathbf{X_T}[m], \mathbf{x}_{c,n}\right)$ is the kernel function that captures the similarity between the query point $\mathbf{x}_{c,n}$ and the training point $\mathbf{X_T}[m]$. The spatial correlation between the grid cells is captured using the sparse kernel function [20] whose diameter can be controlled using the kernel scale parameter l. Via the usage of l, only the training points within this radius to the query point are used to predict its occupancy state unlike the utilization of whole training points to calculate the covariance in [14]. Here, the training set $\mathbf{X_T}$ and $\mathbf{y}$ vector are created using a line-based measurement model: The free space line between the sensor and the reflection point is sampled with a predefined sampling frequency to obtain the free space points. The free space points and the reflection points are stacked to form $\mathbf{X_T}$. For each free space point the corresponding entry of $\mathbf{y}$ is set to 0 whereas for each reflection point, it is set to 1.

Here, the predictive mean (1.13) is accounted as the best estimate for $\hat{\mathbf{f}}$. Similar to the previously defined approaches, the occupancy state of nth grid cell can be determined by applying a threshold. By following the outlined procedure which is summarized in Algorithm 3, the occupancy states of N grid cells can be estimated using this method.

To sum up, this method offers a simple model with lower complexity than GPOM. Firstly, there are no hyperparameters to be learned. Therefore, no training phase is required. Secondly, the usage of Bayesian generalized kernel instead of Gaussian processes provides simpler update equations (1.13), (1.14) which does not contain any matrix inversion. Finally, the sparse kernel allows the utilization of only training points within a predefined radius, which simplifies the algorithm in terms of computational complexity.

Algorithm 3 Occupancy Grid Mapping Using BGK

Input: Reflection point coordinates $\{\mathbf{x}_m\}_{m=1}^{M}$, grid center Coordinates $\mathbf{X_c}$
Parameters: Beta distribution parameters α_0, β_0, kernel parameters σ_0, l
Output: Binary occupancy grid map: $\hat{\mathbf{g}}$

For each measurement m **do**
  $\mathbf{X}_{\text{free,m}} \leftarrow n_{free}$ sampled cells between the sensor and $\mathbf{x}_m$
  Concatenate $\mathbf{X_T}$ with $\mathbf{X}_{\text{free,m}}$ and $\mathbf{x}_m$
  Add n_{free} 0 entries to $\mathbf{y}$
  Add a single 1 to $\mathbf{y}$

For each grid cell n **do**
  Compute $\alpha[n]$, $\beta[n]$ from (1.15), (1.16)
  Compute $\mu(\mathbf{x}_{c,n})$, $\sigma(\mathbf{x}_{c,n})$ from from (1.13), (1.14)
  $\hat{\mathbf{f}}[n] \leftarrow \mu(\mathbf{x}_{c,n})$

Compute $\hat{\mathbf{g}}$ from (1.6)

This chapter focuses on the significance of occupancy grid mapping in automotive perception. Three widely used occupancy grid mapping methods are introduced and analyzed, outlining their respective advantages and limitations. By identifying the

drawbacks of these existing methods, a research problem is formulated, motivating the development of a novel occupancy grid mapping approach that effectively addresses these limitations.

1.3 The Main Problem

The above mapping algorithms [13–15,17] developed for robotic applications do not exploit contextual information and the underlying sparse structure in LiDAR and radar sensor measurements jointly when estimating occupancy maps for automotive perception.

The sparse structure arises as a consequence of the operating principles of LiDAR and radar sensors. These sensors have limited line-of-sight capabilities. The measurements represent only the locations at which the sensor receives reflections from the environment. In most cases, it is only possible to obtain measurements from the visible edges. Therefore, even in very occluded scenarios, only the borders of the obstacles are reconstructed in the occupancy maps which leads to a sparse representation. This phenomenon is also reflected in Fig 1.2 where each obstacle is represented using only the boundaries which are facing toward the ego vehicle at the $(0, 0)$ location. Moreover, only the objects in the line-of-sight of the sensor are reconstructed. The objects which are obstructed or hidden behind another object are not reconstructed in occupancy maps which decreases the density of the resultant map promoting sparse structure.

Another issue to address is the spatial correlation among the grid cells. The mapping algorithms mentioned earlier [14, 15] address this issue by incorporating kernel functions that express the global relationships between grid cells. However, while these approaches capture some spatial dependencies, they may encounter challenges in accurately representing the complex driving environment. Kernel-based methods tend to rely on smooth kernel functions, which can cause issues when dealing with small obstacles or objects in close proximity to each other. These smooth kernels may lead to missing detections of small obstacles, as the sparse nature of sensor measurements can obscure them within the global model. Moreover, in some cases, usage of kernel functions cases limits the method's capability to represent narrow spaces between obstacles that are in close proximity to each other representing them as a single combined obstacle can cause an oversimplification of the environment. This representation does not have a negative effect on defining the borders of the drivable area. However, when it is intended to identify the objects from the generated occupancy grid maps, this issue becomes problematic. In reality, objects in proximity may have distinct boundaries and individual characteristics, but a global kernel function might not be able to capture such fine-grained details.

The final issue to address is the dimension of the map vector $\mathbf{f}$. The complexity of the occupancy grid mapping algorithms increases with the number of grid cells and it is not restricted in the existing algorithms. In the generated occupancy maps, the mapping area is defined as a big square or rectangular area around the vehicle without excluding any areas which are not related to driving scenarios. Generally, big static structures like buildings, trees; parking areas; parks, and big empty fields take place in the range of the map. However, their representation in the occupancy maps is not

crucial since the aim is to define the driving space accurately and detect the existence of obstacles. Reconstructing the non-related static parts increases the dimension and complexity of the problem significantly whereas the complexity is one of the things to be optimized firstly for autonomous driving applications as the existence of the obstacles should be reported in real-time.

1.3.1 Problem Definition

To address the mentioned issues our research question is:

- **How to construct occupancy grid maps of a defined reconstruction area of any shape from map priors that encapsulate local spatial dependencies and promote sparsity?**

The main problem, can be partitioned into the following sub-problems:

- identifying drivable/non-drivable areas in the pre-processing step to decrease complexity
- constructing a prior that encapsulates local spatial correlation information between neighboring cells
- constructing sparse occupancy maps using this prior
- coming up with quantitative metrics to evaluate the resultant occupancy grid maps

1.3.2 Contribution

- We design a linear measurement model for the occupancy grid mapping problem which is applicable for both LiDAR and radar point clouds.
- We use digital map information to define the area to be reconstructed and decrease the dimension of the map
- We propose a new occupancy grid mapping approach that outperforms the existing benchmark algorithms in terms of resolving the obstacles better and eliminating road reflections in some scenes.
- We introduce two quantitative metrics to evaluate the performance of occupancy grid mapping algorithms evaluate our method using real-world LiDAR and radar point cloud data obtained from comprehensive datasets.

1.4 Outline

In this section, the role of occupancy grid maps in automobile perception is described which is followed by the introduction of the existing occupancy grid mapping algorithms and the definition of the problem. Following the proposed problem, a new occupancy grid mapping approach is proposed whose derivation and implementation are explained

in Section 2. Then real-world sensor data that is used to test the algorithms, the method to generate the ground truth maps, and the defined quantitative metrics are described in Section 3, which is followed by the evaluation of the performance and comparisons with the performance with the chosen benchmark algorithms.

1.5 List of Publications From This Thesis

Conference Proceeding

- **C. Onen**, A. Pandharipande, G. Joseph, and N. J. Myers, "LiDAR-based occupancy grid map estimation exploiting spatial sparsity," in Proc. IEEE Sens. Conf., Accepted.

Journal Paper

- **C. Onen**, A. Pandharipande, G. Joseph, and N. J. Myers, "Occupancy Grid Mapping for Automotive Driving Exploiting Clustered Sparsity," IEEE Sensors Journal, In preparation.

2 Methodology

In this chapter, we present the methodology for constructing an occupancy map measurement model, utilizing point cloud data acquired by the ego vehicle through LiDAR or radar sensors [4], [21] and for estimating the occupancy states of the grid cells from the measurement model. Firstly, the pre-processing step is described which aims to decrease the redundancy in the obtained sensor data and define the reconstruction area. Then, the sensor measurement models for LiDAR and radar sensors are described which are used to transform the point cloud data into a linear sensor model. This description is followed by the introduction of the proposed occupancy grid estimation framework which aims to estimate sparse occupancy grid maps from the proposed linear measurement model.

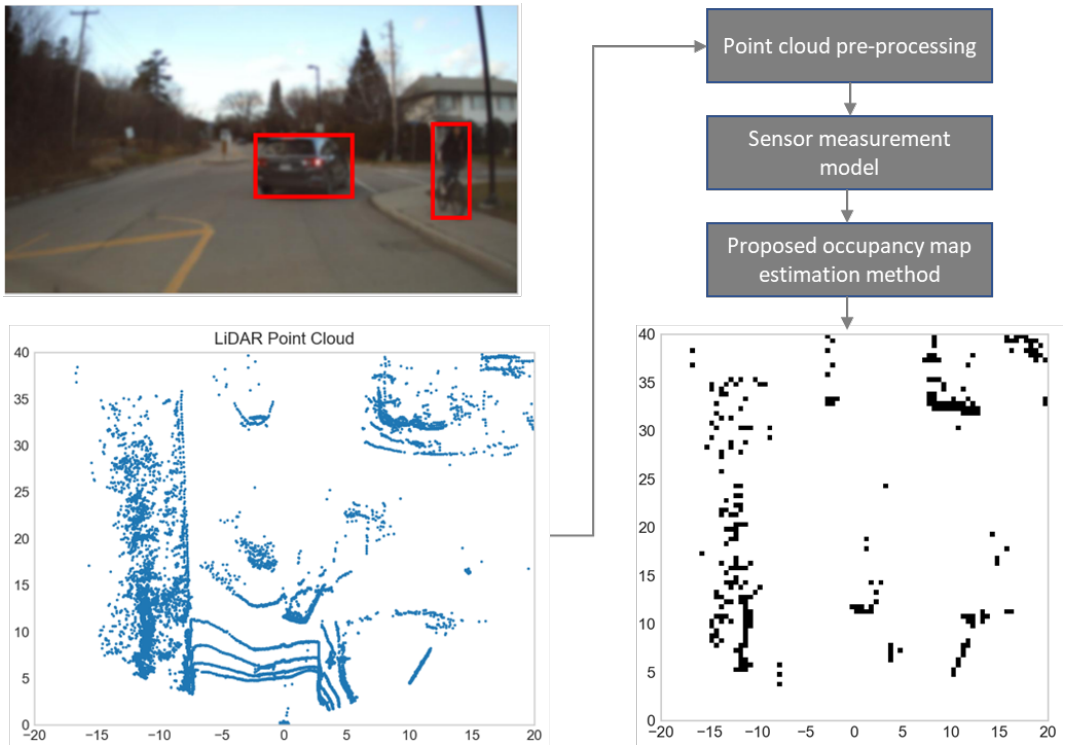
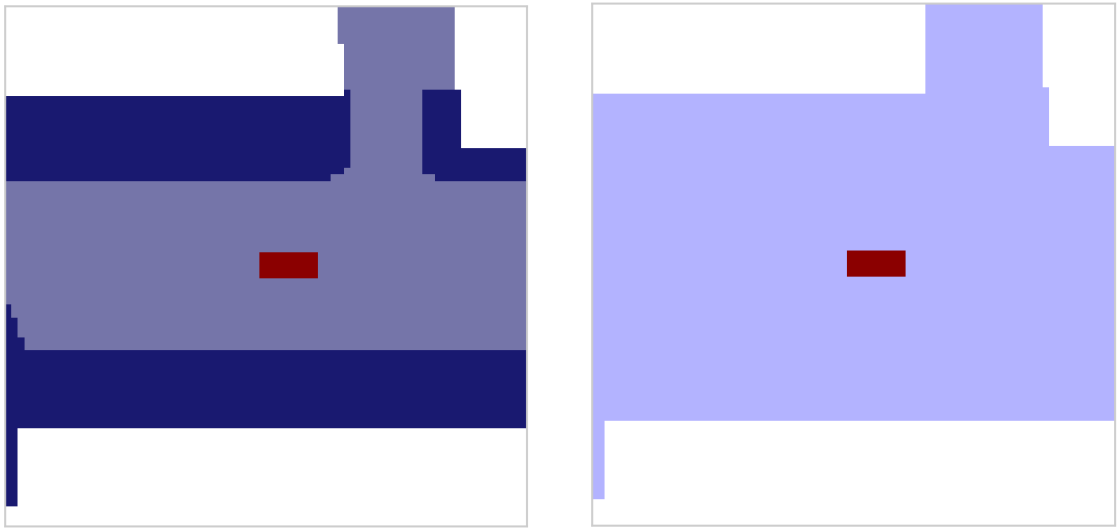


Figure 2.1: The proposed approach involves preprocessing the obtained sensor data, creating a sensor measurement model based on the filtered data, and then estimating the occupancy values using this model.

2.1 Preprocessing and Usage of Digital Map Information

When a complete point cloud representation of a scene is obtained, a brief preprocessing step is applied to eliminate points that lie outside the map's range and those that fall below or exceed the desired height threshold. The motivation for this step is to eliminate reflection points that do not correspond to the objects within the designated area of interest and to reduce the dimensionality of the occupancy grid estimation problem. The inclusion of height thresholds proves beneficial in mitigating the impact of reflections originating from the road surface, as well as tall objects such as trees and traffic signs that extend over the road. Let M be the number of remaining points, referred to as reflection points, which indicate potential occupied locations within the designated area of interest. Each point represents 3D cartesian coordinates of a reflection coming from a potential obstacle and they are represented in a global coordinate system using the ego vehicle's position as the origin. We aim to generate an occupancy grid representation of the area of interest by using these reflection points.



(a) Extend of Road and Walkway Areas

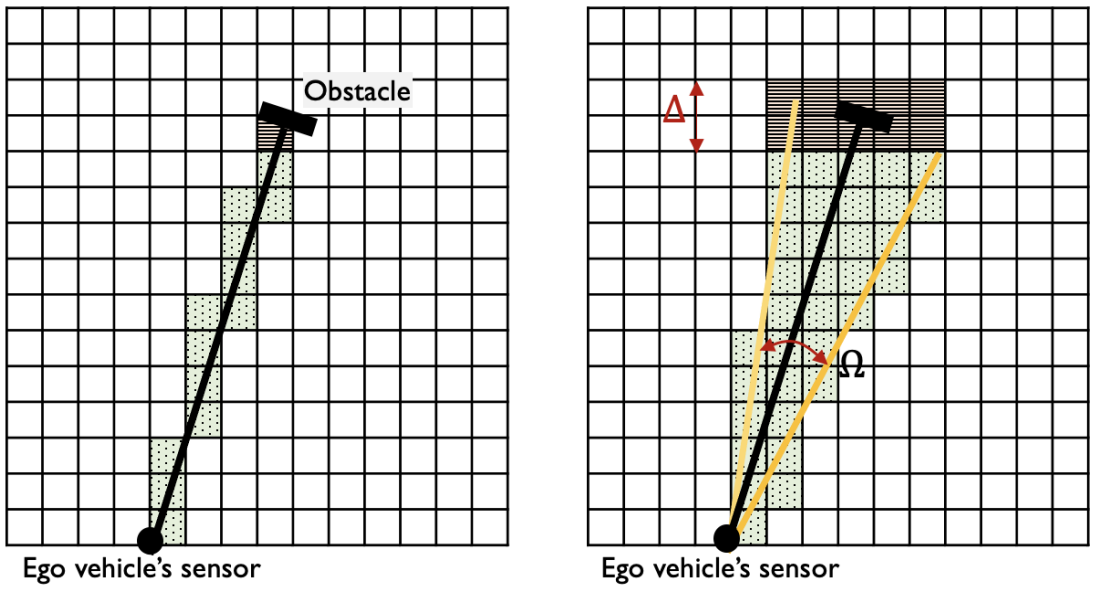
(b) Focus Area

Figure 2.2: The focus area is defined as the combination of road and walkway areas. (a) shows the walkway area in dark blue and the road area in gray. The combined focus area is shown with light blue in (b). The red rectangle indicates the location of the ego vehicle.

We further process the measurements to reduce the dimension of the occupancy map to be estimated. To this end, we include contextual information from digital maps [22] in the model by defining a *focus area*. The focus area is the area of interest whose map is reconstructed and it depends on the application. Our specific emphasis is on driving applications, wherein the area of interest is determined by two constraints. Firstly, the defined area must encompass the road on which the ego vehicle is presently traversing. Secondly, it should also encompass the pavements adjacent to the road. Hence, the objective is to detect all occupancies occurring both on the road and the pavement area [23], primarily consisting of vehicles and pedestrians, although not exclusively limited to them. To be able to use this information in the occupancy grid mapping

algorithms, the contextual segmentation information must be extracted and integrated into the algorithm in real-time. Besides, the precise location of the ego vehicle must be tracked. Therefore, open-source digital mapping services like Google Maps and Open Street Maps can be used for this purpose which provides segmentation information of the surroundings of a chosen GPS location. The remaining area may encompass parking areas, buildings, fields, and other elements that are not crucial for driving scenarios.

Therefore, the focus area is defined as the combination of road and walkway areas as shown in Fig. 2.2. After the focus area is defined, the sensor measurement models for LiDAR and radar are designed to estimate the occupancy states of the grid calls that fall inside the focus area.



(a) LiDAR Sensor Model

(b) Radar Sensor Model

Figure 2.3: For LiDAR point clouds, our method labels all the cells along the line between the sensor and a reflection point as free. For sparse radar point clouds, the cells within a beam width of Ω are labeled as free and those in a region of width Δ around the reflection point are labeled as occupied.

2.2 Sensor and Signal Model

2.2.1 LiDAR Measurement Model

We now construct a linear measurement model of $\mathbf{f}$ using a LiDAR point cloud. For every LiDAR measurement, it can be inferred that the associated reflection point, (discretized to the nearest grid cell) represents occupancy, while the cells located along the free space line connecting the ego vehicle and the reflection point are considered unoccupied and labeled as free space points. If the cell corresponding to the nth entry of $\mathbf{f}$ is a reflection point, we set $f[n] = y_{\text{occ}}$. Here, y_{occ} is a parameter chosen to be close to 1 to indicate that the cell is likely to be occupied. In compact form,

$$\mathbf{e}_n^{\mathsf{T}}\mathbf{f} = y_{\text{occ}} \tag{2.1}$$

when a reflection is observed from a cell indexed n, where $\mathbf{e}_n$ denotes the n^{th} column of the $N \times N$ identity matrix.

We define $\mathcal{F}_n$ as a set of grid cells that are sampled along the line from the ego vehicle to the nth cell (which is a reflection point). The probability that the cells in $\mathcal{F}_n$ are unoccupied is

$$\prod_{k \in \mathcal{F}_n} (1 - f[k]). \tag{2.2}$$

$f[k]$ is expected to be small as the corresponding cell is likely to be unoccupied. Under this assumption, this expression can be approximated to

$$1 - \sum_{k \in \mathcal{F}_n} f[k] = 1 - \sum_{k \in \mathcal{F}_n} \mathbf{e}_k^{\mathsf{T}} \mathbf{f}. \tag{2.3}$$

We set this approximation equal to $1 - y_{\text{free}}$, where y_{free} is a parameter chosen to be close to 0. Therefore,

$$\sum_{k \in \mathcal{F}_n} \mathbf{e}_k^{\mathsf{T}} \mathbf{f} = y_{\text{free}}. \tag{2.4}$$

Observe that each LiDAR point cloud reflection results in two linear measurements of the occupancy map as given in (2.1) and (2.4). A collection of M point cloud reflections thus leads to $2M$ linear equations of $\mathbf{f}$ given by

$$\mathbf{y} = \mathbf{C}\mathbf{f} + \mathbf{w}. \tag{2.5}$$

Here, $\mathbf{w} \in \mathbb{R}^{2M}$ is modeled as independent and identically distributed Gaussian measurement noise with zero mean and unknown variance σ^2, and $\mathbf{C} \in \{0, 1\}^{2M \times N}$ is a selection matrix defined by (2.1) and (2.4). It indicates the presence of free-labeled cells and reflection points for each sensor measurement. Finally, $\mathbf{y} \in \mathbb{R}^{2M}$ is the noisy observations of the occupancy values of the N grid cells. Specifically, if the mth reflection point corresponds to the nth entry of $\mathbf{f}$, we set $y[2m-1] = y_{\text{occ}}$ and $y[2m] = y_{\text{free}}$. Also, the $(2m-1)$th and $2m$th rows of $\mathbf{C}$ are $\mathbf{e}_n^{\mathsf{T}}$ and $\sum_{k \in \mathcal{F}_n} \mathbf{e}_k^{\mathsf{T}}$, respectively.

2.2.2 Radar Measurement Model

We observed that the radar point cloud provided in the datasets we utilize exhibits sparsity when compared to the LiDAR point cloud. The use of such low-density points results in poor occupancy grid map estimates than the LiDAR-based approach. To address this issue, we employ an artificial enhancement technique to enrich the point cloud using a sensor model influenced by the "Inverse Sensor Model" which is proposed in [13]. Following this approach, a conical beam originating at the ego vehicle's sensor is constructed for each point cloud measurement. This beam has a width of Ω and terminates in a region of width Δ about the reflection point, as shown in Fig. 2.3 (b). The region where the beam terminates is marked using dashed cells, while the remaining cells within the conical beam are represented as dotted cells. We use $\mathcal{O}_n$ and $\mathcal{F}_n$ to denote the sets comprising the indices of the dashed cells and dotted cells respectively. Similar to our LiDAR-based measurement model, we define two measurements of the

occupancy map using these sets:

$$\sum_{k \in \mathcal{F}_n} \mathbf{e}_k^\mathsf{T} \mathbf{f} = y_{\text{free}}, \tag{2.6}$$

$$\sum_{k \in \mathcal{O}_n} \mathbf{e}_k^\mathsf{T} \mathbf{f} = y_{\text{occ}}. \tag{2.7}$$

The key distinction between the LiDAR sensor model and the radar sensor model lies in the labeling of occupied cells. While our LiDAR-based model labels a single cell as occupied for a reflection point, our radar-based model marks all the cells within $\mathcal{O}_n$ as occupied for the same point. Here, the motivation is to fill the gaps between the sparse radar point data and to represent the obstacles in the region using several cells. A linear measurement model similar to (2.5) can be constructed, by putting together (2.6) and (2.7) for all the M measurements. In the radar case, the measurement matrix $\mathbf{C}$ is such that its $(2m-1)$th and $2m$th row are $\sum_{k \in \mathcal{O}_n} \mathbf{e}_k^\mathsf{T}$ and $\sum_{k \in \mathcal{F}_n} \mathbf{e}_k^\mathsf{T}$ respectively.

After defining our sensor measurement model as a linear Gaussian model, our occupancy mapping problem reduces to estimating the map vector $\mathbf{f}$ from the linear measurement vector $\mathbf{y}$ by exploiting sparsity and spatial correlation in the map. The final thing to be specified is the estimation framework.

The estimation framework depends on the structure and characteristics of the unknown vector. In Chapter 1, the sparse structure of the occupancy grid maps was explained, and the research problem was defined accordingly. To address the sparsity aspect of the occupancy grid maps, Sparse Bayesian Learning (SBL) can be used as the estimation framework.

SBL is a statistical framework that can be used for estimating sparse signals or models from noisy data [24]. In SBL, the goal is to find a sparse solution that best explains the observed data by assuming that most of the coefficients are zero or close to zero. It is commonly used in linear regression problems, where the number of predictors or features is much larger than the number of observations, resulting in an underdetermined system. SBL can accurately estimate the nonzero coefficients and their locations, making it a popular approach for many signal processing and ML applications. SBL is a tool that can be used to estimate the occupancy probabilities in occupancy maps as the occupancy maps have sparse structures.

In addition to the inherent sparsity, in occupancy maps, we also observe that the occupied cells occur in clusters due to the physical constraints and layout of the environment. This means that the presence of an obstacle usually results in multiple adjacent cells being occupied as most of the obstacles that exist in traffic are larger than a single grid cell (this assumption may not hold for occupancy grid maps with large grid cells e.g. 1m x 1m). This is due to obstacles on the road being larger than a single grid cell. Thus, it is reasonable to assume that the occupancy maps have a block-sparse structure, where the occupied cells occur mostly in clusters. Various algorithms were introduced in signal processing literature to estimate block sparse vectors. The SBL algorithms Block-OMP [25], Group Basis Pursuit [26], Model-CoSamMP [27], and block sparse Bayesian learning (BSBL) [28] assume the availability of knowledge on block partitions. On the other hand, algorithms like pattern-coupled sparse Bayesian learning (PC-SBL) [29], learned sparse Bayesian learning (L-SBL) [30], expanded block

sparse Bayesian learning (EB-SBL) [28] and alternative EB-SBL [31] does not assume any prior knowledge of the block partitioning. In a rapidly changing driving scenario, the knowledge of the occupancy patterns is not available, so we focus particularly on the second set of algorithms. Taking this approximation into account, PC-SBL is a promising method to estimate the structure and location of these occupancy patterns from the available sensor data. By exploiting the inherent sparsity and structure in the occupancy maps, PC-SBL can accurately estimate the occupancy probabilities in sparse regions, making it a suitable approach for occupancy mapping applications in robotics and autonomous systems. The description of the conventional PC-SBL algorithm and its usage in occupancy map estimation is provided in section 2.2.3.

2.2.3 Occupancy Grid Map Estimation Algorithm

The PC-SBL [29] algorithm is designed to estimate the sparse coefficients of a linear regression problem. It is a variation of SBL that is specifically designed to recover signals with block-sparse structures whose structure is not known. To be more specific, this algorithm is applicable for the scenarios in which the signal to be estimated has nonzero values in clusters. The pattern dependencies are introduced by using pattern-coupled hierarchical Gaussian prior in which a set of hyperparameters are assigned to control the sparsity of each coefficient. The coefficients are coupled to each other by making the prior of each coefficient controlled not only by its own hyperparameter but also by the hyperparameters of its neighbors. This prior structure encourages the solution to be blocked-sparse. The associated hyperparameters are learned by maximizing their posterior given the noise-corrupted measurements and the posterior mean and variance of the coefficients are estimated using Bayesian inference.

The problem of recovering block sparse signal from noisy corrupted measurements is in the form of a linear Gaussian model which is the same as (2.5). In the conventional SBL, $\mathbf{f}$ is assigned a prior

$$p(\mathbf{f}|\boldsymbol{\alpha}) = \prod_{n=1}^{N} p(f[n]|\boldsymbol{\alpha}) = \prod_{n=1}^{N} \mathcal{N}(0, (\alpha[n]^{-1}). \tag{2.8}$$

Here, $\alpha[n]$ is the hyperparameter that controls the sparsity of $f[n]$. It is called the precision of $f[n]$ and equals to the inverse of its variance. When $\alpha[n]$ goes to infinity, its variance goes to zero and it approaches its mean value of 0. In this model, the hyperparameters for each coefficient are independent and it does not promote any clustered structure. To promote it, motivated by the PC-SBL framework, we use a different type of Gaussian hierarchical prior on the sparse vector $\mathbf{f}$,

$$p(\mathbf{f}|\boldsymbol{\alpha}) = \prod_{n=1}^{N} p(f[n]|\boldsymbol{\alpha}) = \prod_{n=1}^{N} \mathcal{N}(0, (\alpha[n] + \beta \sum_{j \in \mathcal{L}_n} \alpha[j])^{-1}). \tag{2.9}$$

Here, $\boldsymbol{\alpha} \in \mathbb{R}^N$ denotes the unknown hyperparameters, $\beta \in [0, 1]$ is the prefixed coupling parameter, and $\mathcal{L}_n$ denotes the set of neighbors of the nth cell. In [29], the neighbors of nth grid cell are defined as $(n-1)^{th}$ and $(n+1)^{th}$ coefficients but this choice is only applicable when the vector being estimated is one-dimensional. However, in our

scenario, the vector $\mathbf{f}$ represents the occupancy values of the grid cells, which is two-dimensional. Therefore, it is more appropriate to couple the coefficients in a 2D manner similar to the implementations in [32], [33]. In other words, for 2D cases, the neighbors of $f[n]$ can be defined as the immediate neighbors (the adjacent left, right, above, and below cells) of the nth cell as follows,

$$p(\mathbf{f}|\boldsymbol{\alpha}) = \prod_{n=1}^{N} p(f[n]|\boldsymbol{\alpha}) = \prod_{n=1}^{N} \mathcal{N}(0, (\alpha[n] + \beta \sum_{j \in \mathcal{L}_n} \alpha[j])^{-1}). \tag{2.10}$$

According to this Gaussian prior, the diagonal covariance matrix $\boldsymbol{D}$ of $\mathbf{f}$ can be constructed as

$$\mathbf{D} = \begin{bmatrix} \alpha[1] + \beta(\alpha[2] + \alpha[N_y + 1])^{-1} & & & \\ & (\alpha[2] + \beta(\alpha[1] + \alpha[3] + \alpha[N_y + 2]))^{-1} & & \\ & & \ddots & \\ & & & (\alpha[N] + \beta(\alpha[N-1] + \alpha[N - N_y]))^{-1} \end{bmatrix}. \tag{2.11}$$

In this context, we have selectively presented the variance values solely for the first two and the last grid cells. The reason for this selection is to illustrate specific aspects related to the variance matrix, as defined in Equation (2.11) and to visualize how the cells are coupled together. To further clarify the corresponding grid cells mentioned in the equation, we have visualized them in Fig. 2.4.

In order to complete the characterization of the hierarchical prior structure, we place a distribution over the precision parameters $\boldsymbol{\alpha}$ and the inverse of the noise variance $\gamma = \sigma^{-2}$. The conventional SBL approach [24], uses Gamma prior distribution on the hyperparameters with prior parameters a, b, c, $d > 0$ as follows,

$$p(\boldsymbol{\alpha}) = \prod_{n=1}^{N} \Gamma\left(\alpha[n] \mid a, b\right) = \prod_{n=1}^{N} \Gamma(a)^{-1} b^a \alpha[n]^a e^{-b\alpha[n]} \tag{2.12}$$

$$p(\gamma) = \Gamma(\gamma \mid c, d) = \Gamma(c)^{-1} d^c \gamma^c e^{-d\gamma}. \tag{2.13}$$

This particular choice of hyperprior leads to a two-layer Gaussian-inverse Gamma hierarchical prior which encourages eliminating most of the irrelevant coefficients during the training phase leading to sparse solutions. This mechanism is called "Automatic Relevance Determination" and it aims to choose the relevant features in a learning process automatically. In our case, our features are the coefficients and the aim is to eliminate the irrelevant ones in the training phase by choosing appropriate values for the parameters a, b, c, $d > 0$. Here, a and c are the shape parameters that control the skewness of the gamma distribution, and b and d are the rate parameters that determine the spread or concentration of the gamma distribution. In the conventional SBL approach, very small values (e.g., 10^{-4}) are assigned to the rate and scale parameters, resulting in an uninformative prior that manifests as a flat distribution. This choice does not impose any prior belief into the framework making every value equally probable. However, in PC-SBL approach a different choice is made for a: by setting $a = 0.5$ a mild preference

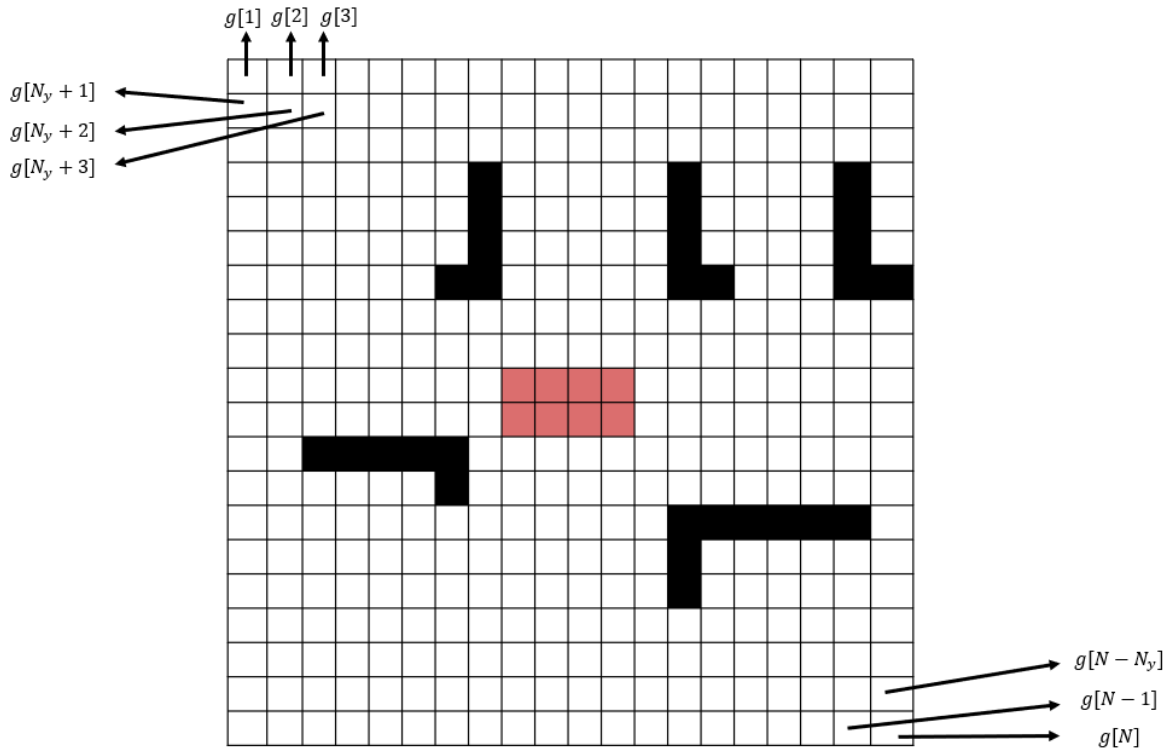


Figure 2.4: The figure displays a sample occupancy grid representation of the environment, where the cells that are used in (2.11) are annotated with their corresponding index.

towards smaller values is imposed while keeping the distribution spread. It implies a slight preference for sparsity and creates a pruning effect for the irrelevant coefficients. Fig. 2.5 demonstrates the difference in the shape of the Gamma distribution when a is chosen as 0.5 and 10^{-4}. Larger a values increase the peak of the distribution as shown in Fig. 2.5 and make it more likely for the hyperparameters to take larger values. Intuitively, when a is larger, a mild preference for a larger hyperparameter is imposed leading to variance values going to zero making the associated coefficients zero.

Next, we use the Bayesian estimation framework to build a bridge between the prior and the observed sensor data to estimate the occupancy states of the grid cells. Using the above hierarchical prior model, we use type II maximum likelihood (ML) estimation of $\mathbf{f}$, where we first estimate the parameters $\boldsymbol{\alpha}$ and γ from the measurement $\mathbf{y}$. The usage of type II maximum likelihood estimation is motivated by its ability to effectively handle hierarchical models, where the estimation of certain parameters relies on the outcomes of previous estimations. In our context, employing type II ML allows us to iteratively estimate the parameters $\boldsymbol{\alpha}_t$ and γ_t from the measurements $\mathbf{y}$ at each iteration t, and then use the estimated parameters to compute the MAP estimate of $\mathbf{f}$, which is the mean of the Gaussian distribution $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\alpha}_t, \gamma_t)$. This estimation relies on the iterative expectation-maximization method with $\mathbf{f}$ being the unobserved latent variable. In EM, a sequence of estimates for $\boldsymbol{\alpha}_t$ and γ_t are generated by maximizing a lower bound of the joint posterior probability $p(\boldsymbol{\alpha}, \gamma \ |, \mathbf{y})$ iteratively. The lower bound is the mean of the joint log-posterior of $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$, i.e., $E_{\mathbf{f}|\mathbf{y},\boldsymbol{\alpha}_t,\gamma_t}[\log p(\boldsymbol{\alpha}, \gamma \mid \mathbf{f}, \mathbf{y})]$

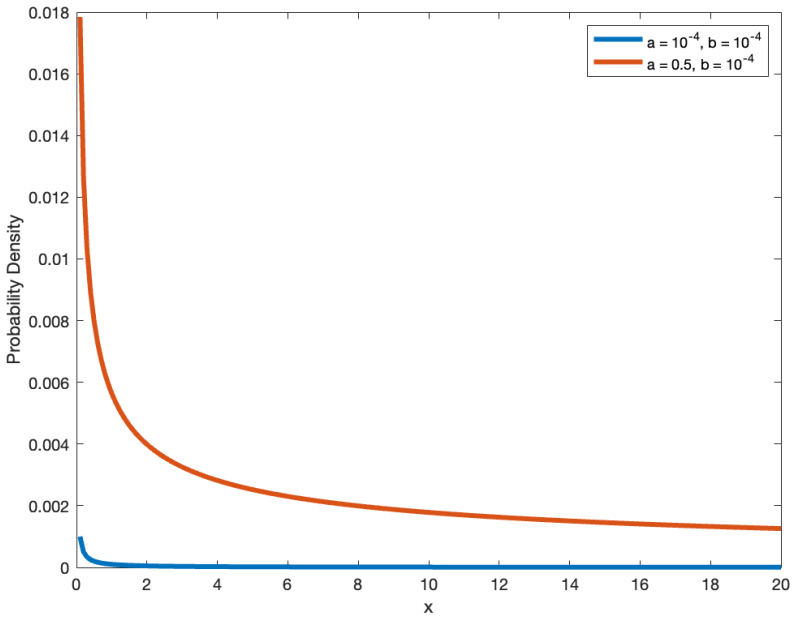


Figure 2.5: Illustration of the effect of shape parameter on Gamma distribution. The blue plot represents the distribution of $\boldsymbol{\alpha}$ when the shape and scale parameters (a and b) are chosen as 10^{-4}. On the other hand, the red plot shows the distribution when the shape parameter (a) is incremented to 0.5 while keeping the scale parameter the same.

which is also called the Q function. The maximization is carried out in two successive steps: Expectation Step (E-Step) and Maximization Step (M-Step). In E-Step, the Q function is calculated given the $\boldsymbol{\alpha}_{t-1}$ and γ_{t-1} from the previous iteration and the sensor measurements.

E-Step:

$$Q\left(\boldsymbol{\alpha}, \gamma \mid \boldsymbol{\alpha}_{t-1}, \gamma_{t-1}\right) = E_{\mathbf{f}|\mathbf{y},\boldsymbol{\alpha}_{t-1},\gamma_{t-1}}[\log p(\boldsymbol{\alpha}, \gamma \mid \mathbf{f}, \mathbf{y})] \tag{2.14}$$

Using Bayes' Theorem, the posterior $p(\boldsymbol{\alpha}, \gamma \mid \mathbf{f}, \mathbf{y})$ can be decomposed as

$$p(\boldsymbol{\alpha}, \gamma \mid \mathbf{f}, \mathbf{y}) \propto p(\boldsymbol{\alpha})p(\mathbf{f} \mid \boldsymbol{\alpha})p(\gamma)p(\mathbf{y} \mid \mathbf{f}, \gamma), \tag{2.15}$$

which allows us to write the Q function as a sum of two terms $Q_\alpha(\boldsymbol{\alpha}_{t-1})$ and $Q_\gamma(\gamma_{t-1})$ as

$$\begin{aligned} Q\left(\boldsymbol{\alpha}, \gamma \mid \boldsymbol{\alpha}_{t-1}, \gamma_{t-1}\right) &= Q_\alpha(\boldsymbol{\alpha}_{t-1}) + Q_\gamma(\gamma_{t-1}) \\ &= E_{\mathbf{f}|\mathbf{y},\boldsymbol{\alpha}_{t-1},\gamma_{t-1}}[\log p(\boldsymbol{\alpha})p(\mathbf{f} \mid \boldsymbol{\alpha})] \\ &+ E_{\mathbf{f}|\mathbf{y},\boldsymbol{\alpha}_{t-1},\gamma_{t-1}}[\log p(\gamma)p(\mathbf{y} \mid \mathbf{f}, \gamma)] \end{aligned} \tag{2.16}$$

where

$$Q_\alpha(\boldsymbol{\alpha}_{t-1}) = \sum_{n=1}^{N} \Bigg(a \log \alpha[n] - b\alpha[n] + \frac{1}{2} \log\Big(\alpha[n] + \beta \sum_{j \in \mathcal{L}_n} \alpha[j]\Big) - \frac{1}{2}\Big(\alpha[n] + \beta \sum_{j \in \mathcal{L}_n} \alpha[j]\Big)\Big(\hat{\mu}[n]^2 + \hat{\phi}[n,n]\Big)\Bigg), \tag{2.17}$$

and

$$Q_\gamma(\gamma_{t-1}) = \frac{M}{2} \log \gamma - \frac{\gamma}{2} E_{\mathbf{f}|\mathbf{y},\boldsymbol{\alpha}_{t-1},\gamma_{t-1}} \left[\|\mathbf{y} - \mathbf{C}\mathbf{f}\|_2^2\right] + c \log \gamma - d\gamma. \tag{2.18}$$

Here, $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Phi}}$ are the mean and covariance matrix of the Gaussian posterior distribution $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\alpha}_{t-1}, \gamma_{t-1})$. At iteration t, the posterior of $\mathbf{f}$ is calculated using the hyperparameters $\boldsymbol{\alpha}_{t-1}, \gamma_{t-1}$ which are calculated at the M-step of the $(t-1)^{th}$ iteration. Specifically,

$$\hat{\boldsymbol{\mu}}_t = \gamma_{t-1} \hat{\boldsymbol{\Phi}}_t \mathbf{C}^T \mathbf{y}, \tag{2.19}$$

$$\hat{\boldsymbol{\Phi}}_t = (\gamma_{t-1} \mathbf{C}^T \mathbf{C} + \mathbf{D}_t)^{-1}. \tag{2.20}$$

We define $\mathbf{D}_t$ as a diagonal matrix with its (n,n) th entry as

$$D_t[n,n] = \alpha_{t-1}[n] + \beta \sum_{j \in \mathcal{L}_n} \alpha_{t-1}[j]. \tag{2.21}$$

The proposed decomposition in (2.16) partitions the joint estimation problem into two distinct subproblems, facilitating the independent update of hyperparameters during the M-Step. By maximizing the Q function in the M-Step, new estimates for the hyperparameters are obtained, effectively optimizing the model to better fit the observed data while considering latent variables.

M-Step:

$$\boldsymbol{\alpha}_t = \arg\max_{\boldsymbol{\alpha}} \; Q_\alpha(\boldsymbol{\alpha}_{t-1}) \tag{2.22}$$

$$\gamma_t = \arg\max_{\gamma} \; Q_\gamma(\gamma_{t-1}) \tag{2.23}$$

It is not straightforward to maximize $Q_\alpha(\boldsymbol{\alpha}_{t-1})$ and find a closed-form analytical solution as the hyperparameters are coupled to each other due to the logarithm term in (2.18). However, a suboptimal solution with a fast convergence guarantee to the true solution is proposed in [29]. According to this approach, an interval for a suboptimal solution is obtained. By setting the first derivative of $Q_\alpha(\boldsymbol{\alpha}_{t-1})$ to zero and using the fact that hyperparameters $\alpha[n]$ and β are non-negative, the interval

$$\alpha_t[n] \in \left[\frac{a}{0.5\hat{\nu}_t[n] + \beta \sum_{j \in \mathcal{L}_n} \hat{\nu}_t[j] + b}, \frac{a + c_0}{0.5\hat{\nu}_t[n] + \beta \sum_{j \in \mathcal{L}_n} \hat{\nu}_t[j] + b}\right] \quad \forall n = 1, \ldots, N \tag{2.24}$$

is obtained. Here, $c_0 = 1.5$ for $n = 2, \ldots, N-1$, and $c_0 = 1$ for $n = \{1, N\}$, and $\hat{\nu}_t \in \mathbb{R}^N$ is the vector of second moments of entries of $\mathbf{f}$ given $\gamma_{t-1}, \boldsymbol{\alpha}_{t-1}$ and $\mathbf{y}$,

$$\hat{\nu}_t[n] = \hat{\mu}_t[n]^2 + \hat{\Phi}_t[n, n]. \tag{2.25}$$

The lower bound in (2.24) is proposed as the sub-optimal analytical solution to this optimization problem. Therefore, the simple and closed-form update equation for α_t is

$$\alpha_t[n] = \frac{a}{0.5\hat{\nu}_t[n] + \beta \sum_{j \in \mathcal{L}_n} \hat{\nu}_t[j] + b}. \tag{2.26}$$

The second parameter to estimate is γ_t and and computing the update equation for γ_t follows a straightforward procedure. By setting the first derivative of $Q_\gamma(\gamma_t)$ with respect to γ_t to zero, we obtain the optimal value for γ_t as

$$\gamma_t = \frac{\|\mathbf{y} - \boldsymbol{C}\hat{\boldsymbol{\mu}}_t\|_2^2 + \gamma_{t-1}^{-1} \sum_n \hat{\Phi}_t[n, n] D_t[n, n] + 2d}{M + 2c}. \tag{2.27}$$

For a more comprehensive derivation of the EM algorithm, we refer the interested reader to [29]. Based on the provided EM algorithm, at iteration t, the mean $\hat{\boldsymbol{\mu}}$ and the covariance $\hat{\boldsymbol{\Phi}}$ of the posterior distribution of $\mathbf{f}$ is calculated using the estimates $\boldsymbol{\alpha}_{t-1}$ and γ_{t-1} of the hyperparameters from the previous time step using (2.20) and (2.20). Then, the new estimates for $\boldsymbol{\alpha}_t$ and γ_t are calculated using (2.26) and (2.27). These EM updates above are iteratively performed until convergence. After convergence, the vector $\hat{\boldsymbol{\mu}}_t$ is the MAP estimate of the unknown occupancy map $\mathbf{f}$. This corresponds to the mean of the converged Gaussian distribution $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\alpha}_t, \gamma_t)$. Finally, a binary occupancy map $\hat{\mathbf{g}}$ is obtained by thresholding the estimated probabilities in $\hat{\mathbf{f}}$ with $\eta_{\text{th,pcsbl}}$. Specifically,

$$\hat{g}[n] = \begin{cases} 0, & \hat{f}[n] < \eta_{\text{th,pcsbl}} \\ 1, & \hat{f}[n] \geq \eta_{\text{th,pcsbl}} \end{cases}. \tag{2.28}$$

To sum up, a summary of the proposed PC-SBL-based algorithm is given in Algorithm 4. The estimated binary occupancy map provides a sparse representation of the objects around the vehicle. The sparsity aspect of the method plays a role in mitigating false alarms caused by road reflections. By leveraging sparsity, the aim is to filter our unwanted signals originating from the road surface, enabling the system to discern genuine objects of interest from potentially misleading clutter. Moreover, via promoting block structure through the usage of described hierarchical prior, we aim to utilize spatial dependencies and reconstruct the borders of the objects in the focus area accurately. The performance of the proposed occupancy grid mapping algorithm is demonstrated and evaluated in Chapter 3 to see whether the utilization of sparsity and block structure makes the proposed algorithm make accurate predictions.

Algorithm 4 Occupancy map estimation using PC-SBL

Input: Reflection point coordinates $\{\mathbf{x}_m\}_{m=1}^{M}$
Parameters: Coupling parameters β, Gamma distribution parameters $a, b, c, d > 0$
Output: Binary occupancy grid map: $\hat{\mathbf{g}}$

For each measurement m **do**
  if (sensor == LiDAR) **then**
    $n \leftarrow$ index of the reflection point $\mathbf{x}_m$ in $\mathbf{f}$
    $\mathcal{F}_n \leftarrow$ indices of (sampled) cells between vehicle and $\mathbf{x}_m$
    $(2m-1)$th row of $\mathbf{C} \leftarrow \mathbf{e}_n^{\mathsf{T}}$
  if (sensor == Radar) **then**
    $\mathcal{O}_n \leftarrow$ indices inside the radar conical beam for $\mathbf{x}_m$
    $\mathcal{F}_n \leftarrow$ indices of the conical beam's terminal points
    $(2m-1)$th row of $\mathbf{C} \leftarrow \sum_{k\in\mathcal{O}_n} \mathbf{e}_k^{\mathsf{T}}$
  $2m$th row of $\mathbf{C} \leftarrow \sum_{k\in\mathcal{F}_n} \mathbf{e}_k^{\mathsf{T}}$
  $y[2m-1] \leftarrow y_{\text{occ}}$
  $y[2m] \leftarrow y_{\text{free}}$

Initialize $t = 0$, $\alpha_t = \mathbf{1}$ and $\gamma_t = 1$
repeat
  Compute $\mathbf{D}_t$ using (2.21) and ν_t from (2.25)
  Update α_t, γ_t using (2.26), (2.27)
  $t \leftarrow t + 1$
until convergence
Compute $\hat{\mathbf{g}}$ from (1.6)

3 Evaluation and Results

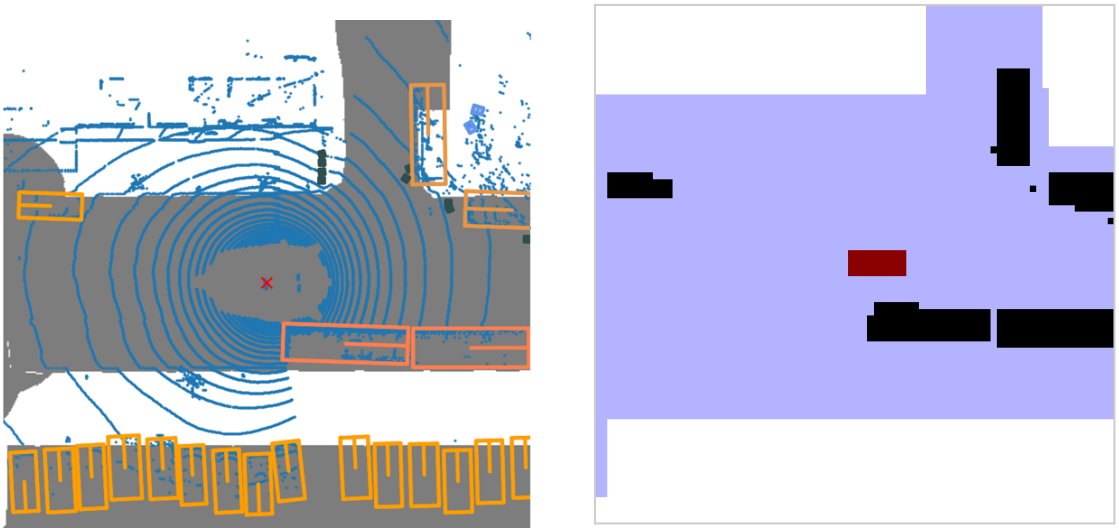
We use real-world LiDAR, radar data, and camera images obtained from the nuScenes [34] and RadIal [35] datasets to test our proposed approach, assess its performance and compare it with the chosen benchmark algorithms [15], [13]. For both of the datasets, we generate results using LiDAR and radar point clouds to show that the proposed approach is applicable for both cases. Both quantitative and qualitative analysis is carried out for performance evaluation. The view of the corresponding scenes obtained from the camera images lets us evaluate the performance of the algorithms quantitatively. By using visual inspection, the detection of the observed obstacles in the camera images is checked. For quantitative analysis, 3D boundary box annotations of the obstacles are needed. These annotations let us create ground truth maps and compare the results using two quantitative metrics we design. The 3D boundary box annotations are provided in the nuScenes dataset but only the locations of the boundary boxes are provided in the RadIal dataset. Therefore, we provide quantitative results for only nuScenes dataset and limit our evaluation to a qualitative comparison of RadIaL dataset. In this section, firstly the method to create the ground truth maps is explained 3.1.1 which is followed by the description of the quantitative metrics 3.1.2. The description of the datasets and the analysis of the chosen scenes are provided in Sections 3.2 and 3.3.

3.1 Performance Evaluation

The performance evaluation includes comparing the results of the proposed approach with the two chosen benchmarks: Binary Bayesian Filter Using the Log-Odds Ratio Formulation [13] and Bayesian General Kernel Inference [15]. We use the generated ground truth maps (3.1.1) to make a quantitative comparison of the outputs of the algorithms.

3.1.1 Ground Truth Maps

In occupancy grid mapping, the accurate localization and representation of objects are vital for analysis and evaluation. To accomplish this, we rely on the dataset and we extract the location and size information of the provided boundary boxes. These boundary boxes represent the spatial extent of the detected vehicles and pedestrians in the LiDAR coordinate frame, as depicted in Fig. 3.1a. To establish ground truth for our algorithm, we compute the footprints of these boundary boxes in the 2D space. This computation involves transforming the 3D coordinates to the 2D plane and capturing the area occupied by the objects which are shown as black boxes in Fig. 3.1b. The resulting footprints serve as essential reference data, enabling us to evaluate the



(a) Provided boundary box annotations and LiDAR point cloud

(b) Sample ground truth map

Figure 3.1: nuScenes dataset provides 2D and 3D boundary box annotations for the labeled objects. Ground truth representation in LiDAR coordinate frame. (a) shows the localization of vehicles and pedestrians in the LiDAR coordinate frame. The enclosed boundary boxes depict the precise spatial extent of the detected objects, while the interior lines elegantly denote the orientations of the respective entities. (b) shows the 2D footprints of the boundary boxes. The focus area is distinctly delineated in blue.

accuracy and effectiveness of our algorithm's performance.

3.1.2 Evaluation Metrics

In assessing the accuracy of the map, two metrics have been employed: the Angular Scan Normalized Mean-Squared Error (NMSE) and the Intersection over Bounding Box (IoBB). The application of these metrics is driven by two primary considerations. Firstly, evaluating the algorithms' proficiency in reconstructing object boundaries facing the EGO vehicle necessitates the use of the NMSE metric. Secondly, a comprehensive assessment of algorithm performance concerning object localization and detection mandates the adoption of the IoBB metric. These metrics provide valuable insights into the effectiveness of the algorithms in the context of our map evaluation.

3.1.2.1 Angular Scan Normalized Mean-Squared Error

NMSE metric provides a comprehensive assessment of the algorithm's ability to accurately reconstruct the environment around the vehicle and identify the closest obstacles in each direction from the vehicle's perspective. This focus on the vehicle's perspective is making it particularly well-suited for autonomous driving applications in which the vehicle's immediate surroundings are of paramount importance. The ability to detect and respond to obstacles directly impacts the overall safety of driving technologies. This capability is particularly crucial in scenarios involving dynamic objects such as

pedestrians, and cyclists where a timely response can prevent accidents and save lines. Moreover, the accuracy in detecting the closest target in each direction provides an indication of the drivable space. For the case of self-driving vehicles, if the drivable space is characterized correctly in each direction, the vehicle can navigate itself safely through the obstacles and plan its trajectory accordingly.

To define NMSE, we denote $\mathbf{d} \in \mathbb{R}^{N_{as}}$ as a vector constructed from the ground truth, which comprises the distances from the ego vehicle to the first occupied cell along each direction of the angular scan. Here, N_{as} is the number of directions and it is equal to

$$N_{as} = \frac{360}{r_{as}}, \tag{3.1}$$

where r_{as} is the chosen resolution for angular scan in degrees. If no occupied cell is encountered along an angular scan then the distance is taken to be the distance until the end of the focus area boundary. The step size of the scan determines the number of directions. In particular, its ith element $d[i]$ represents the maximum distance the ego vehicle can travel in direction i before encountering an obstacle. A similar vector $\hat{\mathbf{d}}$ is defined for the estimated occupancy map. The angular scan NMSE is defined as

$$NMSE = \frac{\|\mathbf{d} - \hat{\mathbf{d}}\|^2}{\|\mathbf{d}\|^2}. \tag{3.2}$$

The normalization term in the denominator ensures that the angular scan metric remains invariant to changes in scale and provides a standardized and fair evaluation of the algorithm across different scenarios.

3.1.2.2 Intersection over Boundary Boxes

IoBB metric quantifies the degree of spatial overlap between the objects detected by the algorithm and the ground truth objects, which are represented as filled boundary boxes. By calculating it, we can evaluate how well the algorithm captures the spatial coverage and position of the objects in the focus area. A high IoBB score indicates a strong overlap between the detected objects and the ground truth objects, suggesting accurate reconstruction. Conversely, a low IoBB score suggests that the algorithm may have difficulty accurately capturing the objects' boundaries.

To define IoBB, we extend the metric in [36] as the ratio of the overlapping area between the occupied cells and the ground truth boundary boxes to the area of the ground truth boundary boxes. Note that, IoBB lies between 0 and 1, and a score larger than 0 indicates the detection of the corresponding obstacle. However, because of the nature of the LiDAR sensor, the objects are mostly represented by their boundaries in the occupancy grid maps, so when the intersection is calculated over the whole boundary boxes the resultant outcome of this metric is not expected to be close to zero for large obstacles. For obstacles that cover 1-2 grid cells like traffic cones or pedestrians, it is possible to represent the full object in the maps.

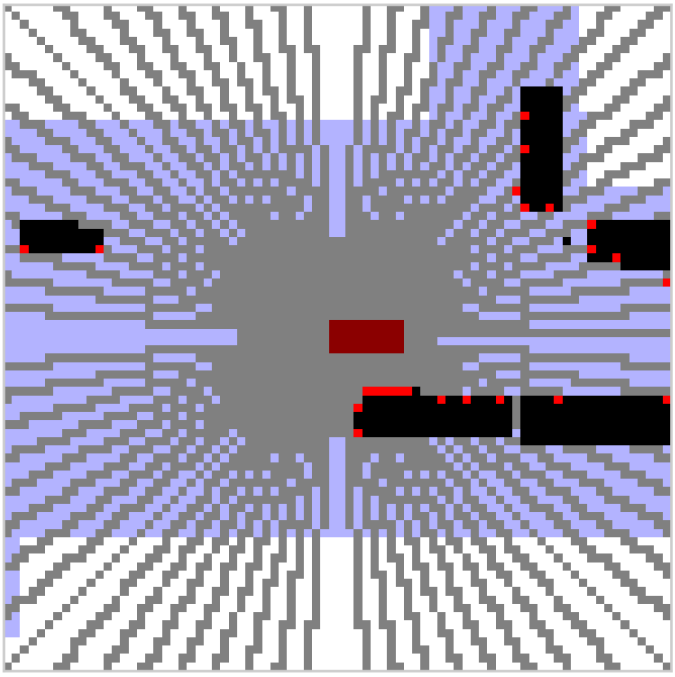


Figure 3.2: Illustration of the angular scan for the ground truth map. The gray lines indicate the scan directions and the red dots are the intersects with the obstacles and the focus area boundaries. The location of the ego vehicle is indicated using dark red.

3.2 Results Based on LiDAR and Radar Point Cloud from nuScenes Dataset

3.2.1 Dataset Description

The nuScenes dataset is a large-scale autonomous driving dataset created by nuTonomy, a self-driving car startup that was acquired by Aptiv in 2017 [8]. The dataset was created to enable research in the field of autonomous driving and to facilitate the development of advanced perception algorithms for self-driving cars.

The nuScenes dataset contains a diverse range of sensor data, including 360-degree camera imagery, LiDAR point clouds, and radar data [8]. The data was collected from a fleet of autonomous vehicles equipped with 6 cameras, 1 LiDAR sensors, and 5 radar sensors, driving around the greater Boston and Singapore areas. The variety in the type of sensors provides a rich and comprehensive set of inputs for perception algorithms [8]. The dataset contains over 1,000 scenes, each of which is about 20 seconds long and captures a variety of traffic scenarios, such as urban driving, highway driving, and off-road driving.

Moreover, in July 2019, nuScenes published a map expansion pack that provides a more comprehensive and detailed set of maps that can be used to train and test autonomous driving algorithms [8]. Actually, the pack is a set of additional semantic layers that were added to the existing high-definition maps provided in the nuScenes dataset. Specifically, it includes 11 semantic layers, which are: crosswalks, sidewalks, curb stones, traffic lights, traffic signs, stop lines, lanes, lane markings, drivable areas,

road dividers, and road edges [8]. These layers are added to the existing high-definition maps provided in the scenes dataset so this segmentation information can be used together with the sensor data [8]. This expansion makes it possible to extract the round boundaries and specify the exact area whose occupancy map will be created [8]. Moreover, semantic labels can be used for motion prediction of vehicles and pedestrians which is a crucial element in creating dynamic occupancy maps [8].

In addition to the sensor data, the nuScenes dataset also contains a comprehensive set of annotations, including object labels and 3D boundary boxes for 23 object classes. This information lets us create ground truth maps as described in 3.1.1. However, it is important to note that the ground truth maps only represent the objects listed in Table 3.1. As a consequence, other objects present in the focus area, such as trees, lamps, traffic signs, or fences, are not accounted for in the ground truth annotations. This limitation has an impact on scene selection as the absence of these additional objects might impact the overall performance and generalization of the occupancy mapping algorithm. When we select the scenes whose results are shown and discussed in the upcoming section, we show pay attention to not using these which include not annotated objects.

Class	**Object ID**
Animal	animal
Human	adult child construction worker personal mobility police officer stroller wheelchair
Movable Object	barrier debris pushable pullable traffic cone

Class	**Object ID**
Static Object	bicycle rack
Vehicle	bicycle bendy bus rigid bus car construction ambulance police motorcycle trailer truck

Table 3.1: 3D Annotated Object Classes in nuScenes

3.2.2 Sensor Suit and Coordinate Frame Convention

Data from three different types of sensors are employed: LiDAR, radar, and camera. It should be noted that these sensors are situated at distinct locations relative to each other. To ensure consistency, a global coordinate frame is established, assuming that the LiDAR sensor is always positioned at the origin, as illustrated in Fig. 3.3. The point clouds obtained from the 5 radar sensors are gathered, and to ensure compatibility and consistency, they are projected onto the LiDAR coordinate frame. This projection allows for unified analysis and calculation of the resultant maps within the LiDAR coordinate system. By employing this approach, the information captured by the radar sensors can be effectively integrated and analyzed within the reference frame of the LiDAR sensor. In this coordinate system, the ego vehicle is located at the center of the

map and is oriented toward the right. The axis originating from the LiDAR sensor's location and extending towards the right side of the map is designated as the positive y-axis, while the axis extending from the LiDAR sensor's location towards the bottom of the map is defined as the positive x-axis. The point clouds obtained from each radar sensor are projected onto the LiDAR coordinates, and the resulting maps are calculated within this domain.

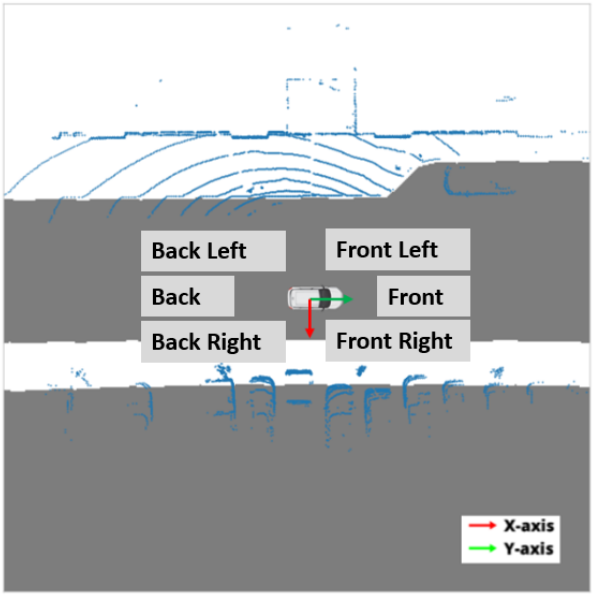


Figure 3.3: LiDAR coordinate frame is chosen as the reference coordinate frame and the maps are constructed accordingly. This figure showcases the EGO vehicle's position and the accompanying LiDAR point cloud. The EGO vehicle's location is represented, and the figure highlights the LiDAR data points that form the point cloud.

3.2.3 Sample Scenes

We choose sample scenes from the nuScenes dataset to demonstrate the performance of our algorithm in complex scenarios and evaluate the obtained results with the benchmark algorithms. To start with, for each scene, NMSE and IoBB values are provided in separate tables, and the corresponding camera images, ground truth map, and the results for 3 algorithms are provided in large combined plots. While choosing these scenes, we pay attention to picking scenes which does not have any unlabeled objects. Certain scenes may contain objects, such as trees, fences, or fire hydrants, not listed in Table. 3.1, which could potentially be misinterpreted as false alarms if detected by the algorithms. To address this, when a large object's location can be accurately determined from camera images and/or sensor measurements, a corresponding bounding box for the object is added to the ground truth map. This allows for precise calculation of quantitative metrics, ensuring a comprehensive evaluation of the algorithms' performance.

In our study, the results were obtained through a process of optimizing the hyperparameters within the algorithms used. The hyperparameters, constants, and thresholds

utilized in our experiments are documented and presented in Table 3.2.

Parameter	Description	Value
N	Number of Grid Cells	6400
$y_{\text{free}}, y_{\text{occ}}$	Labels for the Proposed Measurement Models	$0, 1$
α, β	PC-SBL Prior Parameters	0.5, 1
a, b, c, d	Gamma Prior Parameters	$0.5, 10^{-6}, 10^{-6}, 10^{-6}$
Ω	Beam Width for [13] and Radar Measurement Model	2°
Δ	Obstacle Thickness for [13] and Radar Measurement Model	2 grid cells
$p_{\text{occ}}, p_{\text{free}}$	Occupancy and Non-occupancy Probabilities for [13]	$0.8, 0.2$
σ_0	Kernel Scale for [15]	0.1
α_0, β_0	Beta Prior Parameters for [15]	0.001
l	Kernel Length for [15]	1 m
$\eta_{\text{th,ism}}, \eta_{\text{th,bgk}}, \eta_{\text{th,pcsbl}}$	Thresholds	0.5, 0.5, 0.3
$r_{\text{bgk}}, r_{\text{pcsbl}}$	Sampling Resolution of Free Space	1, 0.5
r_{as}	Resolution for NMSE Metric	5°

Table 3.2: Optimized Parameters for the Benchmark and Proposed Algorithms

We first evaluate the LiDAR results (Figs. 3.4c to 3.4h,) for scene-006, where we examine 7 labeled objects, including 3 cars and 4 traffic cones located at the back-right of the ego vehicle. Due to their small size, the traffic cones are not detected by all the algorithms. Therefore, we have 0 IoBB values for objects with IDs 2, 3, 5, and 7 in Table 3.4 for all the methods. We also notice that [13] and [15] fail to detect the presence of the black car situated at the right-back as depicted in Fig. 3.4f and Fig. 3.4g. Next, we observe that [13] fails to detect objects situated in front of larger obstacles, which is evident from this scene where the car is in front of a large tree. In Fig. 3.4c, the location of the car at the right is completely white which indicates that [13] method makes a very confident decision that this location is not occupied and missed the detection. For the [15] approach, the reason for missed detection is due to road reflections which show up as concentric sectors about the EGO vehicle in Fig. 3.4d. This is because the occupancy values due to road reflections are close to those of the cells corresponding to the car. Actually, the location of the car at the right is labeled with gray in Fig. 3.4dso the method does not say that this location is hundred percent free like [13]. However, the occupancy values are below the chosen threshold

so they do not appear in Fig. 3.4g. This can be solved by choosing a lower threshold but then the road reflections will be detected as occupancies leading to false alarms. This issue highlights the importance of suppressing road reflections. The proposed approach, however, benefits due to a sparse prior that assigns lower occupancy values to the road reflections, enabling the detection of the car as shown in 3.4h. The road reflections can still be recognized in 3.4e. However, the proposed approach successfully assigns very low occupancy values to the cells that the road reflections correspond to and respectively higher occupancy values to the cells corresponding to the car at the right. This is also reflected in Table 3.3 as we observe that our algorithm achieves the lowest NMSE than the benchmarks. Finally, we notice false alarms along the right edge of all the estimated maps, which can be attributed to the branches of trees extending over the road.

The sparse radar point cloud for this scene leads to less informative occupancy maps (Figs. 3.4i to 3.4n). The 4 traffic cones are still undetected, but all methods are successful in detecting 3 cars. The proposed approach and [13] represent the cars with very few detections which can be explained by the poor density of the point cloud. The [15] algorithm, represents the cars with more occupied cells which can be explained by the predictive nature of the algorithm. It is successful in performing extrapolation between sparse data points. Upon analyzing the algorithms' performance in reconstructing the vehicle with ID 11, it is evident that [13] and [15] inaccurately capture the borders of the vehicle. Specifically, these methods detect the front part of the vehicle as closer to the ego vehicle than its actual position. This discrepancy is attributed to the confusion created by the presence of trees extending over the vehicle, causing errors in their estimations. In contrast, the proposed approach demonstrates successful reconstruction of the front border of the obstacle, overcoming any challenges posed by the surrounding trees and achieving more accurate results. It is further illustrated in the last column of Table 3.3, where it attains the lowest NMSE compared to the benchmark methods.

Methods	**NMSE for LiDAR**	**NMSE for Radar**
[13]	1.29	0.33
[15]	1.02	0.31
Proposed	0.71	0.42

Table 3.3: Angular scan NMSE values for scene-006

Object ID	Object Type	IoBB for LiDAR			IoBB for Radar		
		[13]	[15]	Prop.	[13]	[15]	Prop.
1	Car	0.5	0.57	0.29	0.36	0.58	0.43
2	Cone	0	0	0	0	0	0
3	Cone	0	0	0	0	0	0
4	Car	0	0	0.086	0.085	0.32	0.085
5	Cone	0	0	0	0	0	0.
6	Car	0.44	0.38	0.29	0.031	0.19	0.063
7	Cone	0	0	0	0	0	0

Table 3.4: IoBB for scene-006

Next, we consider scene-165 in which the focus area contains 7 labeled objects: 6 cars at the rear and one pedestrian with ID 7 on the front right of the ego vehicle. There is also a fence on the rear right corner which is manually labeled as object 8 and added to the ground truth. Looking at the LiDAR results in Figs. 3.5c to 3.5h, we can say that the pedestrian is only detected by the [15] and the proposed approach. When we look at Figures 3.5f, we see that there are some detections around the pedestrian. Despite the proximity of these occupied cells to the pedestrian, they do not align accurately with their original location, resulting in a missed detection for the approach [13]. In the analysis of the 6 cars (Objects 1 to 6), the proposed approach and [13] successfully detect 5 of them, while [15] manages to detect 4, missing the closest one. The missed detection by [15] can be attributed to the occupancy values of the car being very close to the occupancy values caused by road reflections, leading to confusion in detection. Additionally, all methods fail to detect the car with ID 4, as it is a small car that is shadowed by surrounding cars, resulting in no laser returns from it. Therefore, when the overall detection performance is examined we can say that the proposed method manages to detect 6 out of 7 seven objects while the other methods only detect 5 of them. The better detection performance of our algorithm in detecting objects can be observed from the computed NMSE metric in Table 3.5 and the IoBB values in Table 3.6.

Unlike other scenes, the radar point cloud in this particular scene provides richer information, enabling the detection of cars. As observed in Figs. 3.5i to 3.5n, methods [13] and [15] fail to detect cars 2, 4, 5, and a pedestrian, whereas the proposed method detects them. This promising result highlights the efficacy of our method, even in detecting small obstacles like pedestrians from very sparse point clouds. The superior object detection performance of our algorithm is further supported by the computed NMSE metric in Table 3.5 and the IoBB values in Table 3.6.

Methods	NMSE for LiDAR	NMSE for Radar
[13]	0.41	0.37
[15]	0.37	0.38
Proposed	0.37	0.17

Table 3.5: Angular scan NMSE values for scene-165

Object ID	Object Type	IoBB for LiDAR			IoBB for Radar		
		[13]	[15]	Prop.	[13]	[15]	Prop.
1	Car	0.14	0	0.10	0.10	0.31	0.069
2	Car	0.38	0.59	0.18	0	0	0.059
3	Car	0.28	0.17	0.056	0.028	0	0.083
4	Car	0	0	0	0	0	0.063
5	Car	0.18	0.15	0.091	0	0	0.061
6	Car	0.33	0.15	0.15	0.030	0	0.15
7	Pedestrian	0	0.5	0.5	0	0	1

Table 3.6: IoBB for scene-165

We now consider scene-204 in which the focus area contains 11 labeled objects, including 4 cars, 1 van, and 6 pedestrians to the right of the ego vehicle. As two of the pedestrians (2 and 7) are close to each other, they are represented with a single boundary box leading to 5 boundary boxes for pedestrians in Fig. 3.6b. From the LiDAR results in Figs. 3.6c to 3.6h, we notice that [15] and our approach successfully detect all of the cars, whereas [13] does not detect the car at the right end of the road. Next, we observe that the three methods demonstrate varying performances in detecting pedestrians with IDs 1, 2, 3, 7, and 8, positioned to the right of the car. The IoBB values in Table 3.8 indicate that [13] detects three of the pedestrians, [15] detects 4 of them, and our approach detects 5 of them. The pedestrian with ID 1 remains undetected by all algorithms, since the visibility of this pedestrian is obstructed by a neighboring pedestrian. Also, it is worth mentioning that [15] manages to detect pedestrians 3 and 10 separately by modeling the space between them correctly whereas [13] detects these pedestrians as a combined bigger object. The better performance of our algorithm in detecting the pedestrians is also reflected in the lower NMSE value in Table 3.7.

The results with radar (Figs. 3.6i to 3.6n) are inferior for all three methods, with all the pedestrians undetected. The proposed method performs nominally better than [13] and [15] in detecting the larger objects, e.g. the van is represented by a single occupied cell by the proposed method it is a missed detection for the benchmark methods.

Methods	NMSE for LiDAR	NMSE for Radar
[13]	0.41	0.65
[15]	0.50	0.56
Proposed	0.37	0.46

Table 3.7: Angular scan NMSE values for scene-204

Finally, we consider scene-210 which contains 6 vehicles: 5 cars and 1 construction vehicle. Looking at the LiDAR results in Figs. 3.7c to 3.7h, we observe that all three methods remove the road reflections perfectly which can be explained by the road being very smooth. However, the algorithms' detection performance varies. We observe that all methods are successful in detecting the 2 cars (3 and 4) and the construction vehicle (5) on the left side of the ego vehicle. However, among the three cars on the right, only 2 of them are detected by [15], while all of them are detected by [13] and our proposed

Object ID	Object Type	IoBB for LiDAR			IoBB for Radar		
		[13]	[15]	Prop.	[13]	[15]	Prop.
1	Pedestrian	0	0	1	0	0	0
2	Pedestrian	0	0.5	1	0	0	0
3	Pedestrian	0	1	0.50	0	0	0
4	Car	0.60	0.40	0.27	0	0.012	0.16
5	Van	0.74	0.51	0.28	0	0	0.016
6	Car	0	0.50	1	0	0	0
7	Pedestrian	0.5	0	0	0	0	0
8	Pedestrian	1	0.50	0.5	0	0	0
9	Car	0.48	0.52	0.26	0.083	0.25	0.21
10	Pedestrian	0.25	0.25	0.75	0	0	0
11	Car	0.69	0.39	0.31	0.028	0.028	0.17

Table 3.8: IoBB for scene-204

approach. However, [13] only detects one occupied cell belonging to the car with ID 2, whereas our approach provides a more continuous reconstruction of its border as shown in Fig. 3.7h. Similarly, [15] only detects a few cells belonging to the cars with IDs 2 and 6 while the proposed method provides a more continuous reconstruction of the borders as well. We observe that the benchmark algorithms tend to miss detections of the cars in very close proximity to the ego vehicle either on the left or on the right. In this situation, the windows of the cars are faced towards the ego vehicle decreasing the detection accuracy. Therefore, we can say that our proposed method performs better in detecting close by vehicles and reconstructing their borders without gaps. The better performance of our algorithm is also reflected in NMSE values given in Table. 3.9.

The sparse radar point cloud results in occupancy maps that lack essential details for this scene. Although the IoBB values in Table 3.9 indicate that the proposed method exhibits better detection performance than the benchmark algorithms, it is essential to acknowledge that claiming superior performance in this scenario is not valid. The maps generated from the radar point cloud are considerably sparse due to their limitations, rendering them less informative and hindering precise object detection.

Methods	NMSE for LiDAR	NMSE for Radar
[13]	0.056	0.21
[15]	0.22	0.22
Proposed	0.0078	0.16

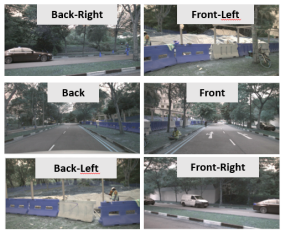
Table 3.9: Angular scan NMSE values for scene-210

To sum up, based on the results obtained from the selected scenes from nuScenes dataset, we see that all algorithms are capable of generating informative 2D occupancy maps using LiDAR data. The LiDAR results prove that the proposed approach performs better than the benchmark algorithms in several aspects. Firstly, it excels in effectively removing road reflections, leading to cleaner and more accurate occupancy maps. Secondly, the proposed approach exhibits superior performance in detecting

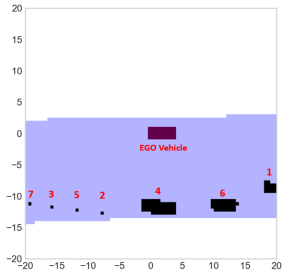
Object ID	Object Type	IoBB for LiDAR			IoBB for Radar		
		[13]	[15]	Prop.	[13]	[15]	Prop.
1	Car	0.028	0	0.17	0.056	0.11	0.17
2	Car	0.27	0.027	0.24	0	0	0.054
3	Car	0.53	0.37	0.32	0	0.026	0.16
4	Car	0.53	0.53	0.29	0.029	0.059	0.059
5	Construction Vehicle	0.55	0.46	0.35	0.10	0.13	0.067
6	Car	0.31	0.22	0.31	0.028	0.028	0.056

Table 3.10: IoBB for scene-210

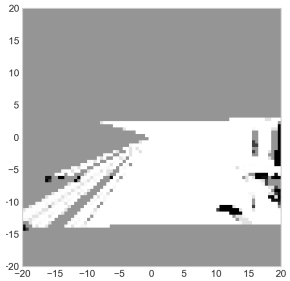
nearby vehicles, precisely identifying and reconstructing their boundaries. Lastly, the method excels in discriminating small obstacles such as pedestrians even in situations where they are in close proximity to each other, showcasing its ability to handle complex and cluttered environments with remarkable accuracy. Unfortunately, the sparse radar data provided in nuScenes do not let us create informative maps in which the boundaries of the obstacles are fully or partially reconstructed. In some of the cases, there are very few occupied cell detections based on sparse radar returns from the obstacles but they do not lead to meaningful representation of these obstacles. The maps generated from the radar point cloud are considerably sparse due to the sensor's limitations, rendering them less informative and hindering precise object detection. However, this issue does not mean that these methods are not capable of generating occupancy maps from radar point clouds. As a result, the comparison of performance must be interpreted cautiously, considering the inherent limitations of the radar data in this specific context. In the next section, it is shown that when an HD radar which is capable of generating denser point clouds is used, also it is possible to create radar occupancy maps whose quality is comparable to LiDAR occupancy maps.



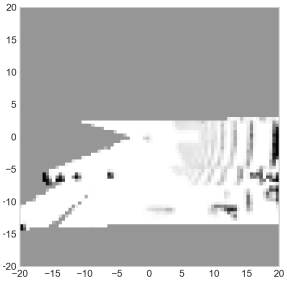
(a) Camera images



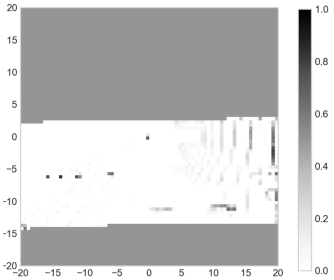
(b) Ground truth map



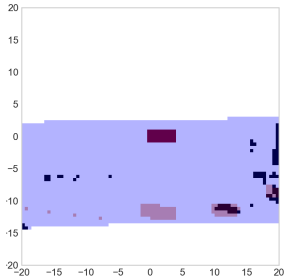
(c) Occupancy values [13]



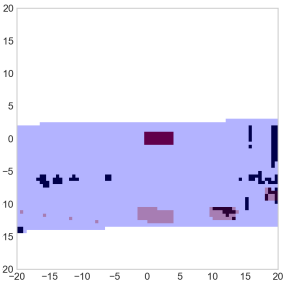
(d) Occupancy values [15]



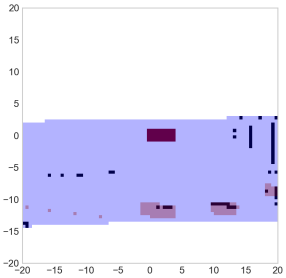
(e) Occupancy values [Proposed]



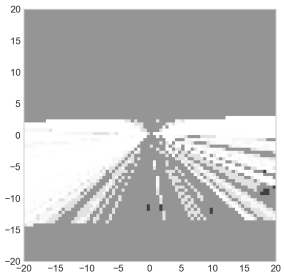
(f) Occupancy map [13]



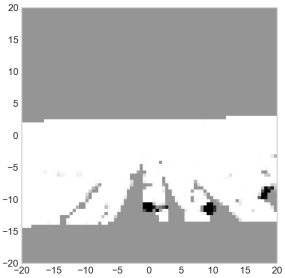
(g) Occupancy map [15]



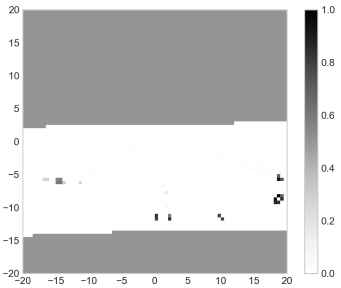
(h) Occupancy map [Proposed]



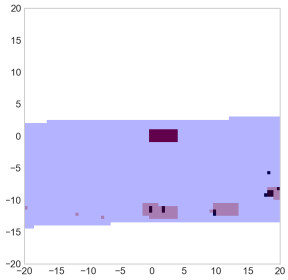
(i) Occupancy values [13]



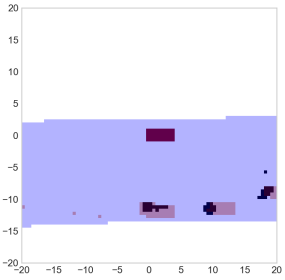
(j) Occupancy values [15]



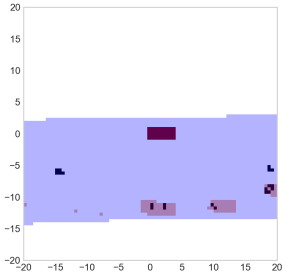
(k) Occupancy values [Proposed]



(l) Occupancy map [13]

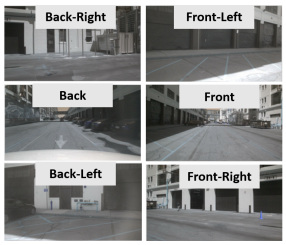


(m) Occupancy map [15]

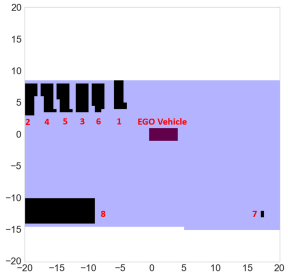


(n) Occupancy map [Proposed]

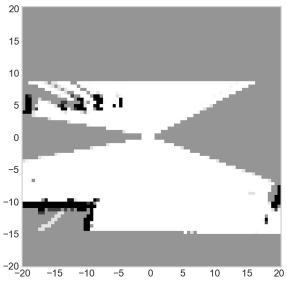
Figure 3.4: LiDAR (c-h) and radar (i-n) occupancy grid mapping results for scene-006 for nuScenes dataset.



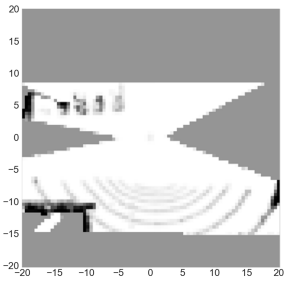
(a) Camera images



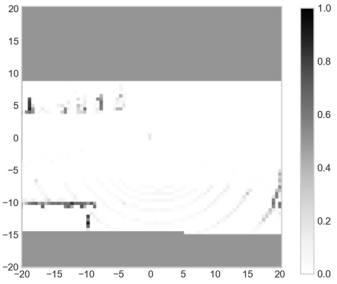
(b) Ground truth map



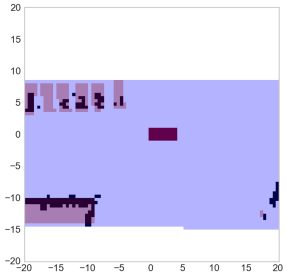
(c) Occupancy values [13]



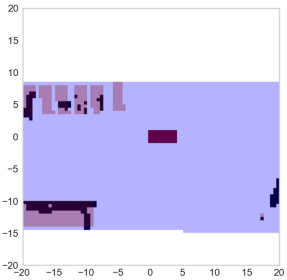
(d) Occupancy values [15]



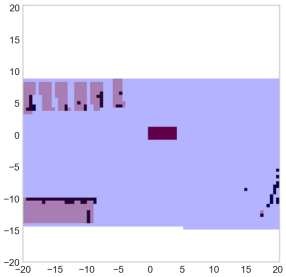
(e) Occupancy values [Proposed]



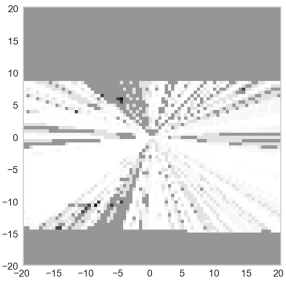
(f) Occupancy map [13]



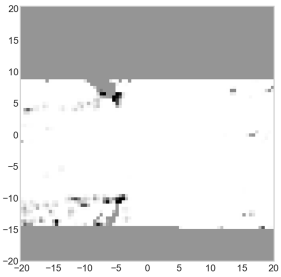
(g) Occupancy map [15]



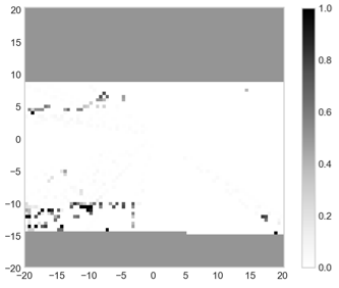
(h) Occupancy map [Proposed]



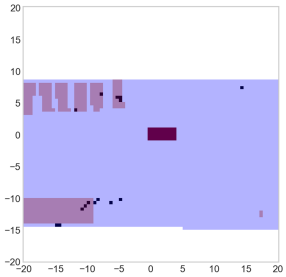
(i) Occupancy values [13]



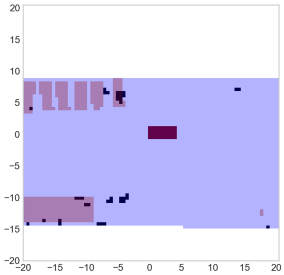
(j) Occupancy values [15]



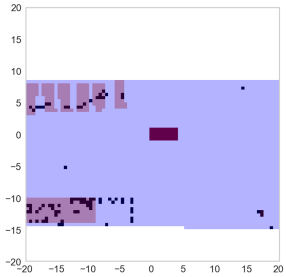
(k) Occupancy values [Proposed]



(l) Occupancy map [13]

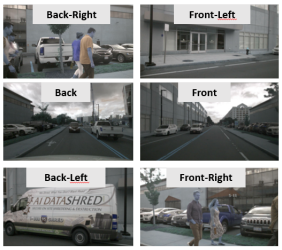


(m) Occupancy map [15]

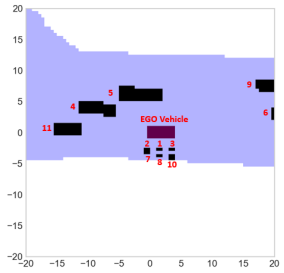


(n) Occupancy map [Proposed]

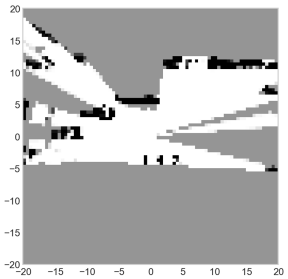
Figure 3.5: LiDAR (c-h) and radar (i-n) occupancy grid mapping results for scene-165 for nuScenes dataset.



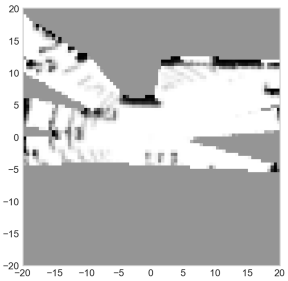
(a) Camera images



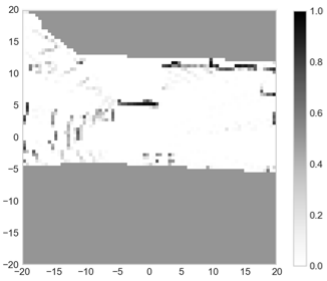
(b) Ground truth map



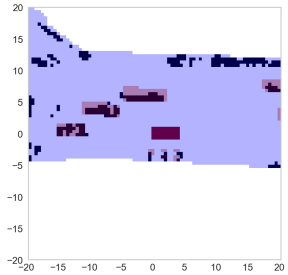
(c) Occupancy values [13]



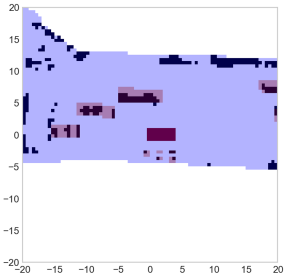
(d) Occupancy values [15]



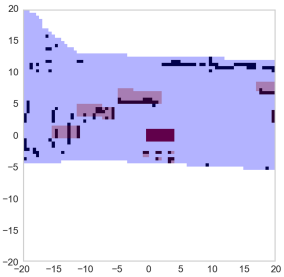
(e) Occupancy values [Proposed]



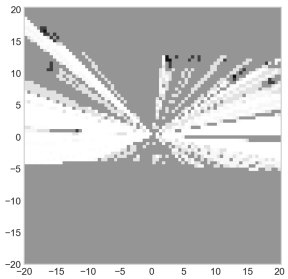
(f) Occupancy map [13]



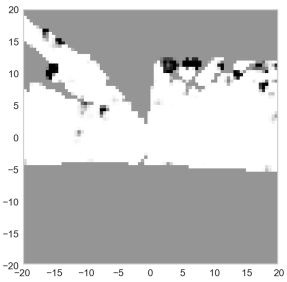
(g) Occupancy map [15]



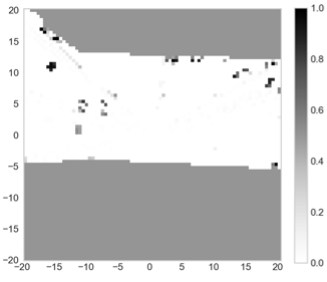
(h) Occupancy map [Proposed]



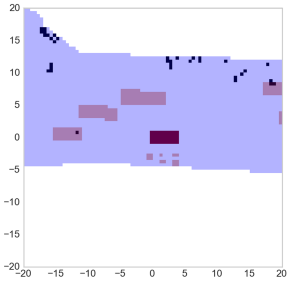
(i) Occupancy values [13]



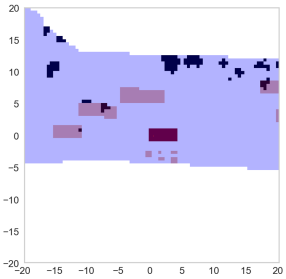
(j) Occupancy values [15]



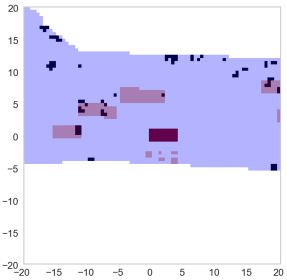
(k) Occupancy values [Proposed]



(l) Occupancy map [13]

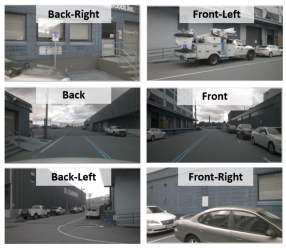


(m) Occupancy map [15]

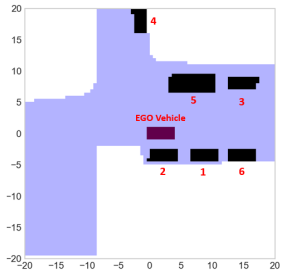


(n) Occupancy map [Proposed]

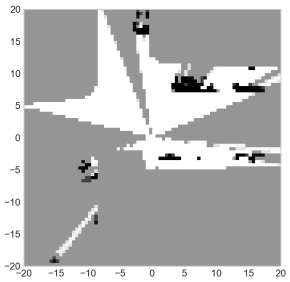
Figure 3.6: LiDAR (c-h) and radar (i-n) occupancy grid mapping results for scene-204 for nuScenes dataset.



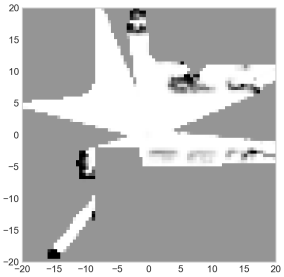
(a) Camera images



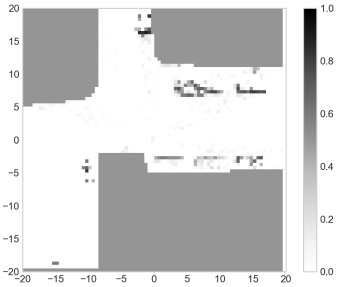
(b) Ground truth map



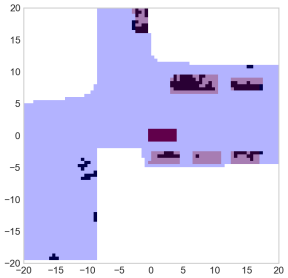
(c) Occupancy values [13]



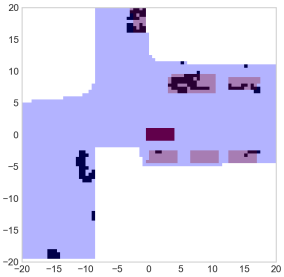
(d) Occupancy values [15]



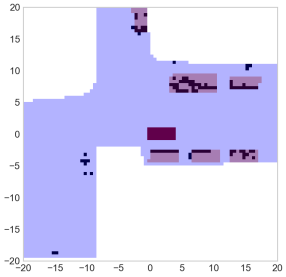
(e) Occupancy values [Proposed]



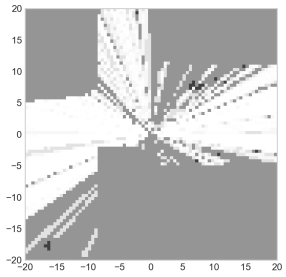
(f) Occupancy map [13]



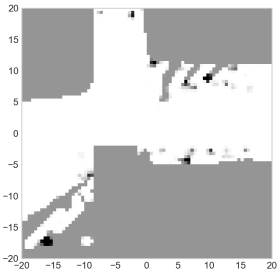
(g) Occupancy map [15]



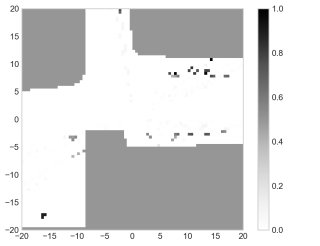
(h) Occupancy map [Proposed]



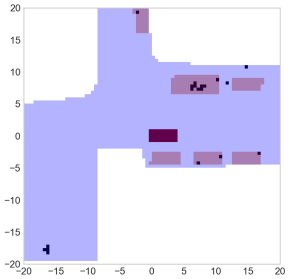
(i) Occupancy values [13]



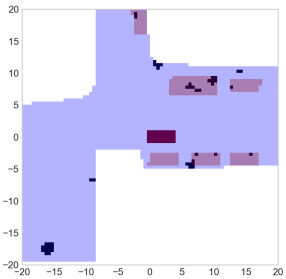
(j) Occupancy values [15]



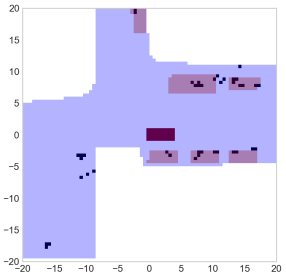
(k) Occupancy values [Proposed]



(l) Occupancy map [13]



(m) Occupancy map [15]



(n) Occupancy map [Proposed]

Figure 3.7: LiDAR (c-h) and radar (i-n) occupancy grid mapping results for scene-210 for nuScenes dataset.

3.2.4 Statistical Performance Analysis Based on Detection Rate

The aspects at which the proposed algorithm performs better than the others are highlighted through the sample scenes from nuScenes dataset. We finally analyze the statistical performance of the proposed method with [13] and [15] using the LiDAR point clouds corresponding to 200 samples. Each scene in the dataset has a duration of approximately 20 seconds, and the scenes are annotated at a frequency of 2Hz, resulting in 40 annotated samples per scene. Within each scene, 20 samples are extracted, obtained by taking every second sample out of the available 40 samples.

As a metric, we consider the ratio of detected objects in the estimated occupancy map to the number of actual labeled objects in the ground truth. From the results shown in Fig. 3.8, we can observe that the median of this metric for the proposed method (0.84) is higher than the other methods (0.74 for [13] and 0.69 for [15]); the variance with the proposed method is 0.02, while [13] has a higher value of 0.03 and a similar value of 0.02 for [15]. A higher mean indicates a higher detection rate hence better detection performance. Therefore, we can claim that our proposed approach provides better detection performance over 200 scenes. The scenes selected for evaluation involve complex and crowded environments with multiple labeled objects. Consequently, numerous occlusions and overlapping obstacles are present, creating challenges in detecting all objects accurately. As a result, the low detection rates in Fig. 3.8 can be attributed to the presence of occlusions and interferences between objects, making it difficult to detect some of them successfully.

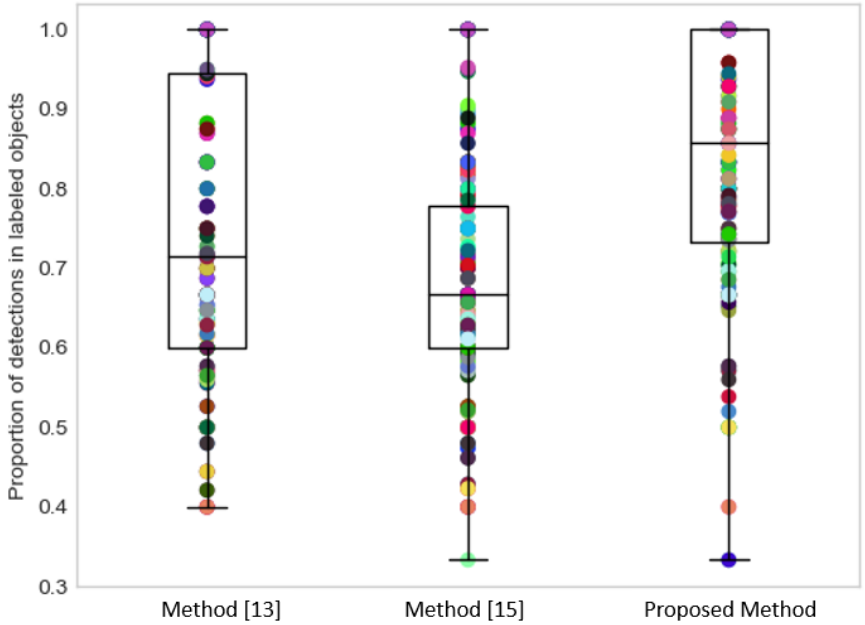


Figure 3.8: The distribution of the proportion of detected objects to labeled objects in the ground truth map. The analysis is carried out on 200 samples from nuScenes.

Methods	Mean	Variance
[13]	0.74	0.03
[15]	0.69	0.02
Prop.	0.84	0.02

Table 3.11: Mean and Variance Values for Different Methods

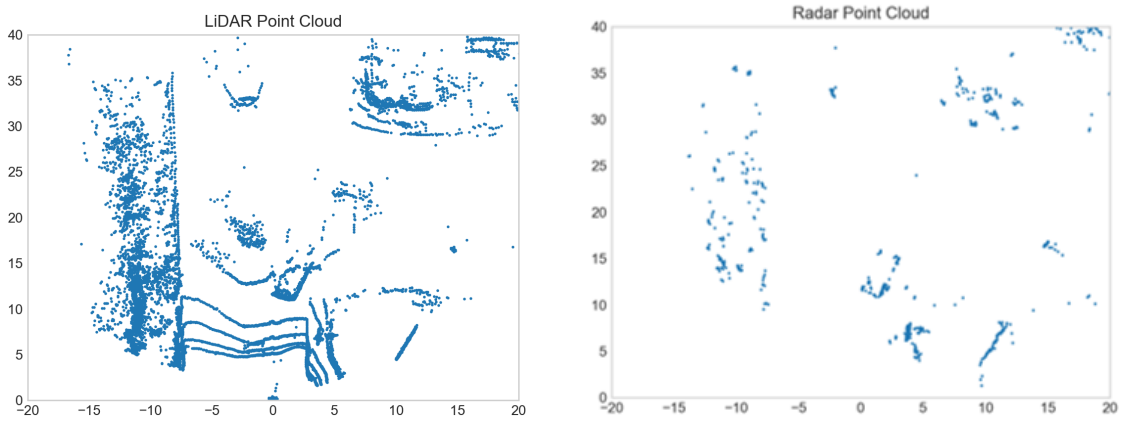
3.3 Results Based on LiDAR and Radar Point Cloud from RadIal Dataset

3.3.1 Dataset Description

The RADIal dataset [37] is a newly developed dataset designed for research on automotive HD radar. It comprises three sensor modalities: camera, radar, and laser scanner, providing opportunities for investigating HD radar fusion with other traditional sensors. The sensor suite used in RADIal includes automotive-grade qualified sensors, along with GPS position and complete CAN bus data of the vehicle, including odometry information. The dataset contains 91 sequences, each ranging from 1 to 4 minutes, totaling approximately 2 hours of data. There are approximately 25,000 synchronized frames in total, with 8,252 frames annotated for vehicle detection [35].

As discussed before, the resolution of the angular radars that are used in the publicly available automotive datasets [8], [38] is poor compared to the resolution of used LiDAR sensors. The highlight of the RadIaL dataset is that it uses an HD (High Definition) radar [35]. It is an advanced radar technology used in automotive applications, particularly in autonomous driving and ADAS. HD radar is designed to provide highly accurate and detailed information about the surrounding environment, including the position, velocity, and shape of objects such as vehicles, pedestrians, and obstacles. Unlike traditional radar systems, which typically have limited resolution and can only detect large objects, HD radar uses multiple antennas and advanced signal processing techniques to achieve higher resolution and precision. This enables HD radar to detect and track smaller objects with greater accuracy, making it a crucial component for enhancing the perception capabilities of autonomous vehicles. The selection of this dataset is primarily driven by the utilization of High-Definition (HD) radar technology, which offers crucial advantages for our research objectives. The HD radar in this dataset provides a radar point cloud with a density comparable to that of LiDAR point clouds as illustrated in Fig. 3.9. This feature enables the integration of data from both sensors to create highly accurate occupancy grid maps. The comparable point cloud densities from HD radar and LiDAR sensors allow for effective usage of our approach to create accurate occupancy grid maps.

Vehicle annotations are included in the dataset as well. 2D boundary boxes are provided in the camera domain for the vehicles. Moreover, the real-world distances of the vehicles to the ego vehicle are provided. Generating vehicle annotations was initially performed using supervision from the camera and laser scanner. A RetinaNet model was used to propose objects in the camera data, and then these proposals were validated using both radar and LiDAR point cloud data. Manual verification was conducted to



(a) Sample LiDAR point cloud

(b) Sample radar point cloud

Figure 3.9: The radar point clouds in the RadIal dataset exhibit comparable density to the LiDAR point clouds. In this particular sample scene, both the radial and LiDAR point clouds offer sufficient information to discern the boundaries and precise locations of the objects present.

confirm or reject the proposed labels. Here, no information about the size and the orientation of the vehicles is provided, we only know a coordinate of a single point from a vehicle in the LiDAR frame and a 2D boundary box in the camera domain [35]. This boundary box only gives information about the size of one side of the vehicle which is facing toward the ego vehicle. It is not possible to extract the 2D boundary box representation of the obstacles from the top view (LiDAR coordinate frame) and create the ground truth maps using the method described in Section 3.1.1. Therefore, quantitative evaluation of the results obtained from this dataset is not possible which limits us to make only qualitative evaluation of the results. Moreover, only the front camera images are provided which limits our qualitative analysis to the front of the car as highlighted in Fig. 3.10.

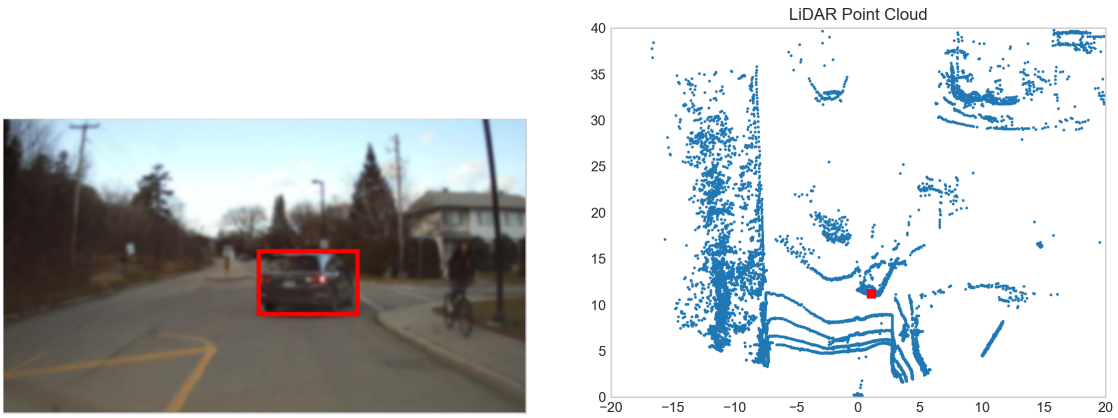


Figure 3.10: Depiction of 2D annotations in Radial dataset.

3.3.2 Sensor Suit and Coordinate Frame Convention

Similar to nuScenes, data from three different types of sensors are employed: LiDAR, radar, and camera. A global coordinate frame was established, assuming that the LiDAR sensor is always positioned at the origin, as illustrated in Fig. 3.11. The radar point clouds are projected onto the LiDAR coordinate frame. In contrast to the coordinate system used in nuScenes, the coordinate system employed in this dataset adopts a different orientation to represent the ego vehicle's location. Here, the ego vehicle is positioned on the x-axis and is oriented upwards. The positive x-axis extends from the LiDAR sensor's location toward the right side of the map, while the positive y-axis extends from the LiDAR sensor's location toward the top of the map. By adopting this coordinate system, the dataset focuses on reconstructing only the area that lies in front of the ego vehicle. This specific choice allows for a more qualitative evaluation of the results by examining the provided front camera images, enabling a better understanding of the perception outcomes in the context of the forward direction of travel. The rationale behind this orientation selection is to facilitate a coherent assessment of the reconstructed environment with respect to the ego vehicle's forward view.

As a result, the dataset provides valuable information to analyze and improve perception algorithms, particularly in scenarios where accurate perception ahead of the vehicle is critical for safe and efficient navigation. Following this convention, tests are carried out on sample scenes chosen from the dataset.

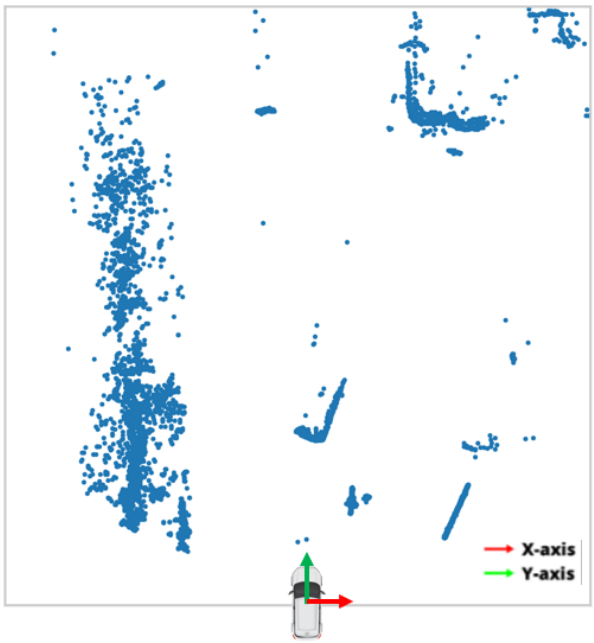


Figure 3.11: Illustration of reference coordinate frame. This figure showcases the EGO vehicle's position and the accompanying LiDAR point cloud.

3.3.3 Sample Scenes

The motivation behind performing tests using data from the RadIaL dataset is to showcase the efficacy of the proposed approach when applied to radar point clouds. To achieve this objective, three scenes are deliberately chosen from the annotated dataset. Subsequently, both LiDAR and radar occupancy maps are generated utilizing benchmark algorithms alongside the proposed approach. This comparison allows for a comprehensive evaluation of the proposed method's performance in handling radar data and underscores its suitability for radar-based mapping applications. The set of hyperparameters and thresholds mentioned in Table 3.2 is applied to obtain the presented results. Due to the unavailability of ground truth maps, a visual evaluation approach is adopted, relying on the provided front-view camera images for assessment.

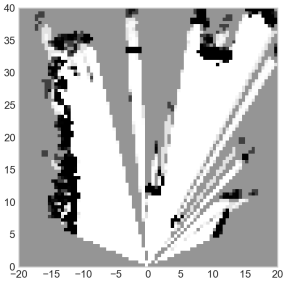
Firstly, we study scene-2681 in which, there is a car and a cyclist in the scene as seen in the camera view Fig. 3.12a. From the LiDAR results in Figs. 3.12b to 3.12g, it is observed that the discrimination between the cyclist and the adjacent street pole is better and our proposed method in comparison to [13] and [15]. The proposed approach successfully reconstructs the space between the cyclist and the pole enabling the detection of two objects separately. All methods based on LiDAR and radar data perform well in detecting both the car and estimating the occupancy map. However, when radar data is used, all methods detect the cyclist and the pole as a single combined object. Although the radar point cloud is dense compared to data provided in nuScenes, still the point cloud density is not enough to discriminate between these two objects and reconstruct the small space between them accurately.

In the following evaluation, we aim to assess the performance of our method in a scene containing an object at an extended range. Up to this point, the occupancy maps created using the lidar point cloud have shown more informative results. However, we now seek to demonstrate that in certain cases, utilizing radar points can be advantageous, leading to more accurate occupancy maps. Instead of claiming superior performance over benchmark algorithms, our focus shifts to investigating a scene that showcases the capabilities of HD radar. One key advantage of HD radars is their ability to detect objects at longer distances. Therefore, we aim to determine whether we can reconstruct a distant object more effectively using a radar point cloud. To explore this, we have selected scene-3218, which features a single car approximately 27 meters away from the ego vehicle, and which is labeled in the camera image (Fig. 3.13). Looking at Figs. 3.13b to 3.13m, we can say that all methods perform well in detecting the car and estimating the occupancy map, while the radar results provide a better representation of the car in comparison to the LiDAR results in Figs. 3.13b to 3.13g. Upon comparing Fig. 3.13g and Fig. 3.13m, it is evident that in the LiDAR results, only a few cells are reconstructed, corresponding to the car. In contrast, the radar map represents the car with a higher number of cells, providing a more detailed representation of its borders and orientation. This difference in representation demonstrates the capability of the radar point cloud to offer a richer and more comprehensive understanding of the car's presence and structure compared to the LiDAR data in this particular scene. This can be attributed to the better detection capabilities of the radar in comparison to a LiDAR for objects that are at longer ranges.

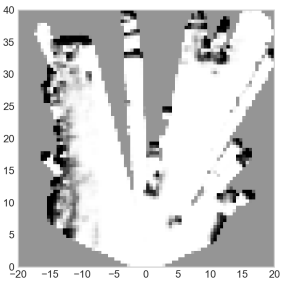
Lastly, we analyze scene-639, which serves as an illustrative instance of a straight-

forward driving scenario. In this scenario, the ego vehicle is positioned on a wide road, facing multiple other cars. Looking at the camera images in Fig. 3.14a, we see 9 labeled cars in this scene, but the extent of the created occupancy maps only covers 5 of them. Our particular focus is on the closest two vehicles as they shape the drivable space in front of the ego vehicle. When LiDAR point cloud is used all methods successfully reconstruct the boundaries of these two vehicles as shown in Figs. 3.14b to 3.14g. When employing a radar point cloud as input, the proposed approach exhibits notable advantages over existing methods in terms of generating smoother and more continuous reconstructions. Notably, [13] demonstrates accurate detection of the back of the car on the right side, providing a continuous reconstruction of its boundary. However, it faces difficulties in reconstructing the front part of the vehicle on the left side. [15] detects some portions of the vehicle on the left side but these identified cells do not accurately correspond to the front part of the vehicle. As a result, it tends to locate the vehicle at a more distant position from its actual location, leading to inaccurate mapping results. Furthermore, [15] offers a reconstruction for the vehicle on the right, facilitating the interpretation of its location and orientation from the generated map. However, a notable issue with this reconstruction is the presence of discontinuities in the vehicle's borders. In contrast, the proposed approach excels in accurately detecting the boundaries of these vehicles, representing them using continuous L-shapes. This innovative representation method leads to more precise and coherent reconstructions for both the front and back parts of the vehicles. By demonstrating these favorable results, the proposed approach highlights its potential for enhancing radar-based mapping tasks.

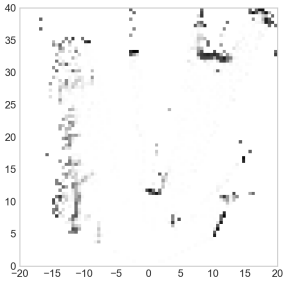
(a) Camera images



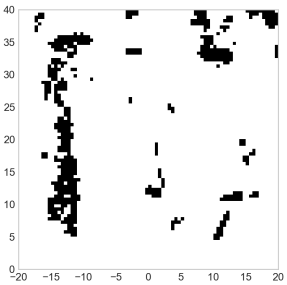
(b) Occupancy values [13]



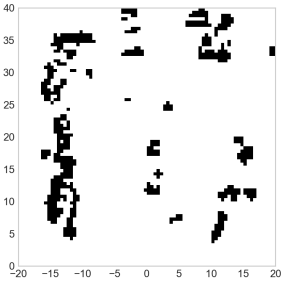
(c) Occupancy values [15]



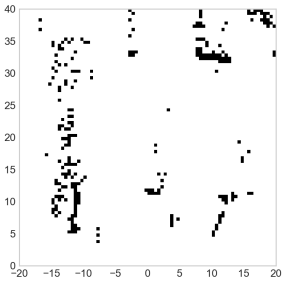
(d) Occupancy values [Proposed]



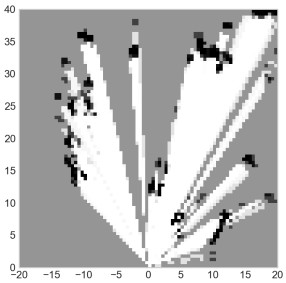
(e) Occupancy map [13]



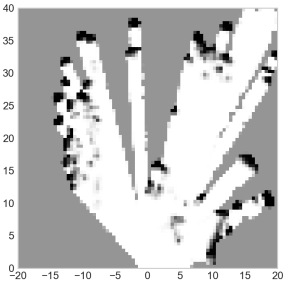
(f) Occupancy map [15]



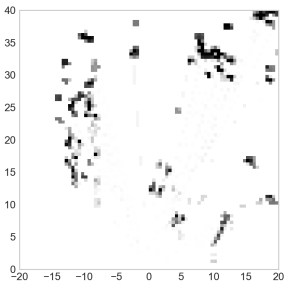
(g) Occupancy map [Proposed]



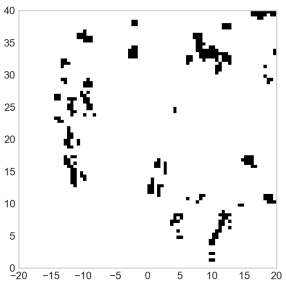
(h) Occupancy values [13]



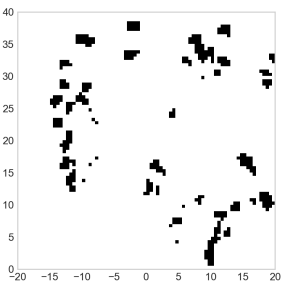
(i) Occupancy values [15]



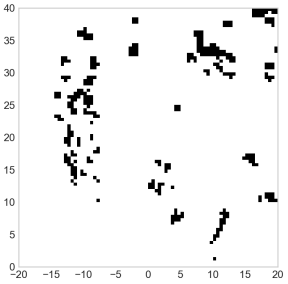
(j) Occupancy values [Proposed]



(k) Occupancy map [13]



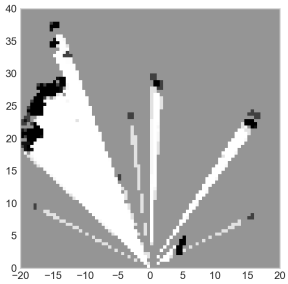
(l) Occupancy map [15]



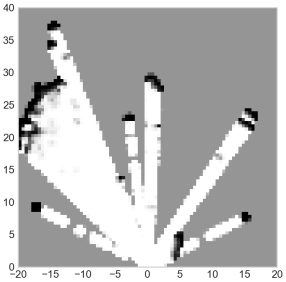
(m) Occupancy map [Proposed]

Figure 3.12: LiDAR (b-g) and radar (h-m) occupancy grid mapping results for scene-2681 from RADIal dataset.

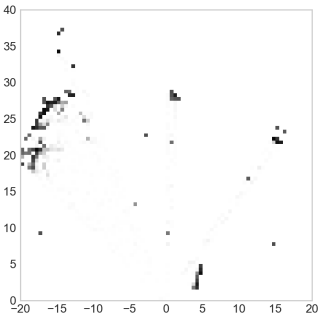
(a) Camera images



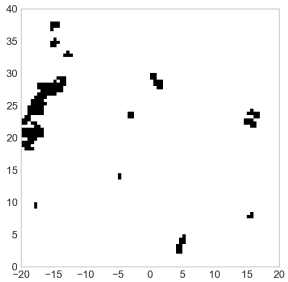
(b) Occupancy values [13]



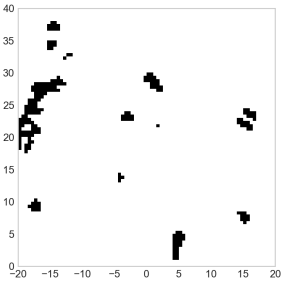
(c) Occupancy values [15]



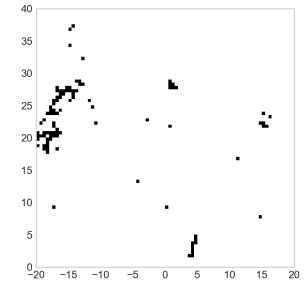
(d) Occupancy values [Proposed]



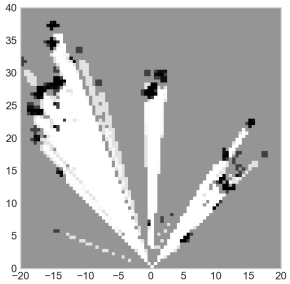
(e) Occupancy map [13]



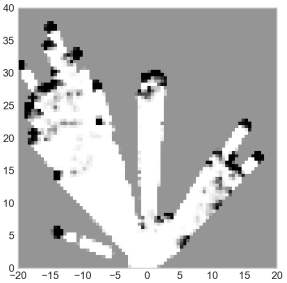
(f) Occupancy map [15]



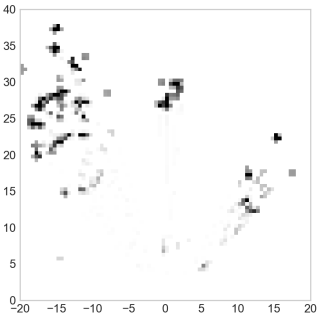
(g) Occupancy map [Proposed]



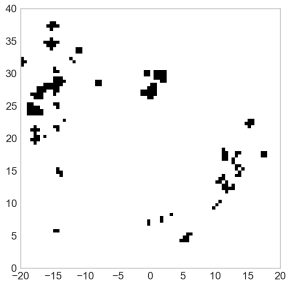
(h) Occupancy values [13]



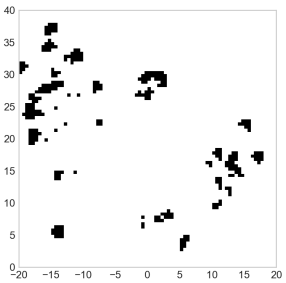
(i) Occupancy values [15]



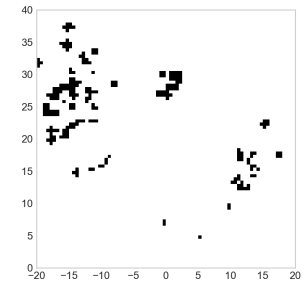
(j) Occupancy values [Proposed]



(k) Occupancy map [13]

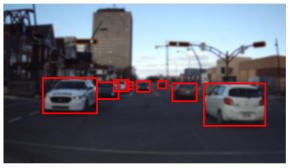


(l) Occupancy map [15]

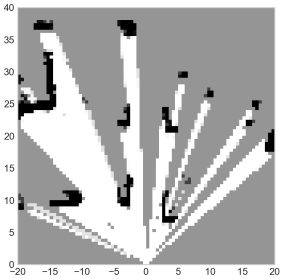


(m) Occupancy map [Proposed]

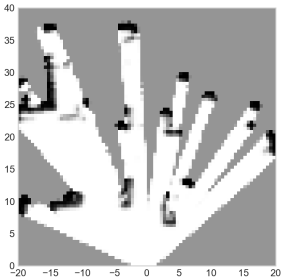
Figure 3.13: LiDAR (b-g) and radar (h-m) occupancy grid mapping results for scene-3218 from RADIal dataset.



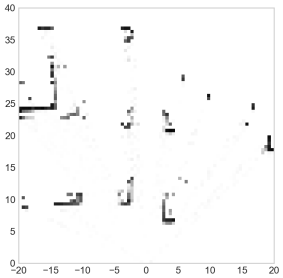
(a) Camera images



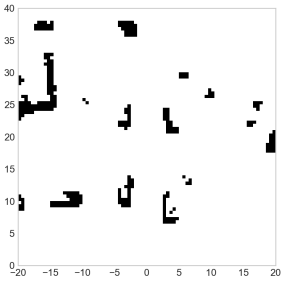
(b) Occupancy values [13]



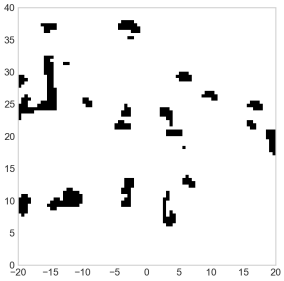
(c) Occupancy values [15]



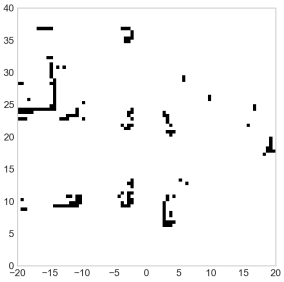
(d) Occupancy values [Proposed]



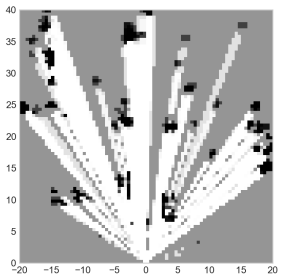
(e) Occupancy map [13]



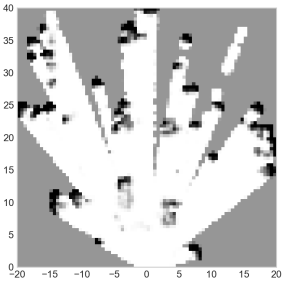
(f) Occupancy map [15]



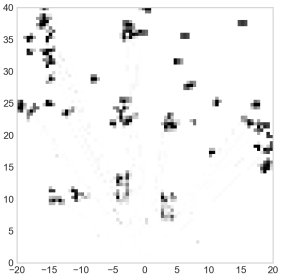
(g) Occupancy map [Proposed]



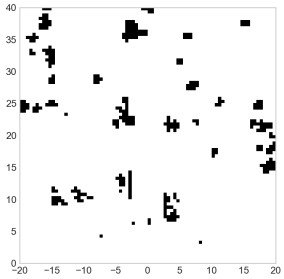
(h) Occupancy values [13]



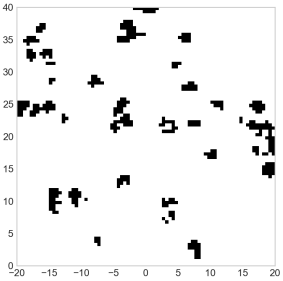
(i) Occupancy values [15]



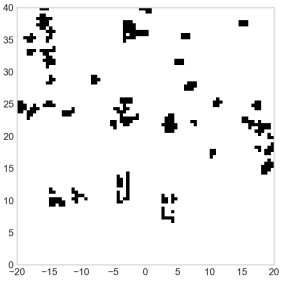
(j) Occupancy values [Proposed]



(k) Occupancy map [13]



(l) Occupancy map [15]



(m) Occupancy map [Proposed]

Figure 3.14: LiDAR (b-g) and radar (h-m) occupancy grid mapping results for scene-639 from RADIal dataset.

To conclude, based on the results obtained from the selected scenes from RadIaL dataset, we see that the proposed approach also provides accurate reconstructions using radar point cloud when the point cloud density is high enough. However, we see that in most of the scenarios maps generated using LiDAR point are more informative and include more details. Firstly, the higher density of LiDAR point clouds let the representation of the narrow space between the obstacles correctly enabling the detection of nearby obstacles separately. Furthermore, the boundaries of the vehicles are reconstructed more smoothly. It has been observed that radar data is more helpful than LiDAR when detecting obstacles in long range. Keeping these observations in mind, the conclusions drawn from this study are discussed in Chapter 4.

4 Conclusion and Future Work

4.1 Conclusion

Occupancy map estimation problem using point cloud data originating from the range sensors radar and LiDAR was considered in this study. To address the gap in the literature for an occupancy grid mapping method that addresses spatial dependencies and sparsity jointly, we applied the PC-SBL algorithm to the 2D occupancy grid map estimation problem. To make the method applicable to occupancy grid map estimation, linear sensor measurement models were designed for the sensor modalities LiDAR and radar. A common linear model was proposed from which the occupancy values can be estimated using PC-SBL and the performance of the proposed approach was demonstrated using real-life data from two automotive datasets nuScenes and RadIal.

When applying the proposed approach to LiDAR point cloud data, better performance was observed in two aspects than the chosen benchmark algorithms. First, the occupancy values due to road reflections were suppressed and the possible false alarms were avoided. This is due to the sparse prior and exploiting spatial correlation. The proposed approach tends to reconstruct the objects in the form of blocks while suppressing the noisy measurements in between the blocks. However, the algorithm's success in suppressing false alarms does not make the algorithm miss the detection of smaller objects like pedestrians. It was observed through the sample scenes chosen from the datasets that the proposed method is able to detect smaller obstacles like pedestrians better. In detecting small objects, there are two aspects that determine the success of the detection: detecting the cells corresponding to the objects as occupied and detecting the free space between the obstacles correctly if there is an observable distance separating them. Our algorithm excels in both cases due to the promoted sparsity. The algorithm gradually enhances sparsity by augmenting the count of sparse coefficients during each iteration of the EM algorithm. This, in turn, leads to an expansion in the number of available free cells within the map. Although this phenomenon decreases the thickness of the boundaries of the detected obstacles and leads to lower IoBB values, it makes the space between the obstacles represented accurately.

When the proposed approach was applied to radar point clouds, different results were obtained from the two datasets. Occupancy map estimation with LiDAR data was better than with radar data on the nuScenes dataset, which can be attributed to LiDAR point cloud data being one to two orders denser compared to radar point cloud data [34]. The resultant radar occupancy grid maps are too sparse and far less informative than LiDAR occupancy grid maps when the nuScenes dataset is used. This is not a problem for our method only, also the benchmark algorithms calculated non-informative maps. Therefore, we can conclude that the radar data provided in the nuScenes dataset is not suitable for the occupancy grid map estimation problem due

to its poor density.

More definitive conclusions were derived from the examination of radar occupancy maps utilizing the RadIaL dataset, which encompasses a higher density of radar point cloud data. Notably, when focusing on the RadIaL dataset, the estimates for radar-based occupancy maps exhibit parity with those generated through LiDAR-based means. Delving into the radar outcomes obtained from this dataset, it becomes apparent that radar excels in detecting objects positioned at greater distances. Nevertheless, its efficacy in accurately discerning small objects and reconstructing the boundaries of proximate vehicles lag behind that of LiDAR.

To conclude, the PC-SBL algorithm has demonstrated its efficacy in addressing the challenge of occupancy grid map estimation, as evidenced by the outcomes of real-world data experimentation. The potential for further advancement lies in its expansion to incorporate various sensor modalities, engaging in sensor fusion techniques, and devising dynamic models to enhance the temporal evolution of occupancy grid maps through iterative processes. A compilation of prospective directions for subsequent research is elaborated upon in Section 4.2.

4.2 Future Work

- Radar point cloud is known to provide better occupancy grid estimates under adverse weather conditions and it performs better in detecting objects in long range. As an extension of this work, the fusion of sensor point clouds from LiDAR and radar using Bayesian learning approaches to improve occupancy map estimation can be explored.

- In our study, we do not utilize the Doppler and velocity information of radar data. They can be used to provide a distinction between static and dynamic objects and to design a dynamic motion model to track the detected objects.

- Finally, we have put forth a proposition advocating the utilization of digital map data for the purpose of conducting elementary segmentation, thereby facilitating the extraction of road boundaries. Alternatively, an alternative approach involves carrying out segmentation on images captured by cameras, subsequently enabling the extraction of road boundaries in a congruent manner.

Bibliography

[1] O.-R. A. D. O. Committee, "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems J3016," Tech. Rep., 2014.

[2] M. Galvani, "History and future of driver assistance," *IEEE Instrum. Meas. Mag.*, vol. 22, no. 1, pp. 11–16, 2019.

[3] Y. Anbanandham, S. s.d, H. Jothi, and M. Vinayagam, "Fundamentals and development of self-driving cars," *Materials today: proceedings*, 05 2020.

[4] E. Marti, M. A. De Miguel, F. Garcia, and J. Perez, "A review of sensor technologies for perception in automated driving," *IEEE Intell. Transp. Syst. Mag.*, vol. 11, no. 4, pp. 94–108, Sep. 2019.

[5] A. Pandharipande, C.-H. Cheng, J. Dauwels, S. Z. Gurbuz, J. Ibanez-Guzman, G. Li, A. Piazzoni, P. Wang, and A. Santra, "Sensing and Machine Learning for Automotive Perception: A Review," *IEEE Sens. J.*, vol. 23, no. 11, pp. 11 097–11 115, 2023.

[6] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, Jun. 1989.

[7] D. Nuss, M. Stuebler, and K. Dietmayer, "Consistent environmental modeling by use of occupancy grid maps, digital road maps, and multi-object tracking," in *Proc. IEEE Intell. Veh. Symp. (IV)*, Jun. 2014, pp. 1371–1377.

[8] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[9] M. Zaffar, S. Ehsan, R. Stolkin, and K. M. Maier, "Proc. sensors, slam and long-term autonomy: A review," in *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2018, pp. 285–290.

[10] J. Ryde and N. Hillier, "Performance of laser and radar ranging devices in adverse environmental conditions," *J. Field Robotics*, vol. 26, pp. 712–727, 09 2009.

[11] J. Degerman, T. Pernstål, and K. Alenljung, "3D occupancy grid mapping using statistical radar models," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2016, pp. 902–908.

[12] C. Yu, V. Cherfaoui, and P. Bonnifait, "Evidential occupancy grid mapping with stereo-vision," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2015, pp. 712–717.

[13] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. The MIT Press, 2005.

[14] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *Int. J. Rob. Res.*, vol. 31, no. 1, pp. 42–62, Jan. 2012.

[15] K. Doherty, J. Wang, and B. Englot, "Bayesian generalized kernel inference for occupancy map prediction," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2017, pp. 3118–3124.

[16] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous robots*, vol. 15, pp. 111–127, 2003.

[17] E. Kaufman, T. Lee, Z. Ai, and I. S. Moskowitz, "Bayesian occupancy grid mapping via an exact inverse sensor model," in *Proc. ACC*, Jul. 2016, pp. 5709–5715.

[18] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006.

[19] W. Vega-Brown, M. Doniec, and N. Roy, "Nonparametric Bayesian Inference on Multivariate Exponential Families," in *Proc. 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, p. 2546–2554.

[20] A. Melkumyan and F. Ramos, "A sparse covariance function for exact gaussian process inference in large datasets." in *Proc. Twenty-first international joint conference on artificial intelligence*, 01 2009, pp. 1936–1942.

[21] W. J. Fleming, "New automotive sensors—a review," *IEEE Sens. J.*, vol. 8, no. 11, pp. 1900–1921, Nov. 2008.

[22] K. Irie and M. Tomono, "Localization and road boundary recognition in urban environments using digital street maps," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 4493–4499.

[23] M. Kurdej, J. Moras, V. Cherfaoui, and P. Bonnifait, "Map-aided evidential grids for driving scene understanding," *IEEE Intell. Transp. Syst. Mag.*, vol. 7, no. 1, pp. 30–41, Jan. 2015.

[24] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, Jun. 2001.

[25] Y. C. Eldar, P. Kuppinger, and H. Bolcskei, "Block-sparse signals: Uncertainty relations and efficient recovery," *IEEE Trans. Signal Process.*, vol. 58, no. 6, pp. 3042–3054, Mar. 2010.

[26] E. Van Den Berg and M. P. Friedlander, "Probing the Pareto frontier for basis pursuit solutions," *SIAM J. Sci. Comput.*, vol. 31, no. 2, pp. 890–912, Nov. 2009.

[27] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde, "Model-based compressive sensing," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1982–2001, Mar. 2010.

[28] Z. Zhang and B. D. Rao, "Extension of SBL algorithms for the recovery of block sparse signals with intra-block correlation," *IEEE Trans. Signal Process.*, vol. 61, no. 8, pp. 2009–2015, Jan. 2013.

[29] J. Fang, Y. Shen, H. Li, and P. Wang, "Pattern-coupled sparse Bayesian learning for recovery of block-sparse signals," *IEEE Trans. Signal Process.*, vol. 63, no. 2, pp. 360–372, Nov. 2014.

[30] R. J. Peter and C. R. Murthy, "Learned-SBL: A deep learning architecture for sparse signal recovery," *arXiv preprint arXiv:1909.08185*, Sep. 2019.

[31] L. Wang, L. Zhao, S. Rahardja, and G. Bi, "Alternative to extended block sparse Bayesian learning and its relation to pattern-coupled sparse Bayesian learning," *IEEE Trans. Signal Process.*, vol. 66, no. 10, pp. 2759–2771, Mar. 2018.

[32] H. Duan, L. Zhang, J. Fang, L. Huang, and H. Li, "Pattern-coupled sparse Bayesian learning for inverse synthetic aperture radar imaging," *IEEE Signal Process. Lett.*, vol. 22, no. 11, pp. 1995–1999, Jul. 2015.

[33] J. Fang, L. Zhang, and H. Li, "Two-dimensional pattern-coupled sparse bayesian learning via generalized approximate message passing," *IEEE Trans. Image Process.*, vol. 25, no. 6, pp. 2920–2930, Apr. 2016.

[34] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11 621–11 631.

[35] J. Rebut, A. Ouaknine, W. Malik, and P. Pérez, "Raw high-definition radar for multi-task learning," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. (CVPR)*, Jun. 2022, pp. 17 021–17 030.

[36] V. Jiménez, J. Godoy, A. Artuñedo, and J. Villagra, "Object-wise comparison of lidar occupancy grid scan rendering methods," *Robotics and Autonomous Systems*, vol. 161, p. 104363, Mar. 2023.

[37] "RADIal dataset," https://github.com/valeoai/RADIal, 2021.

[38] M. Sheeny, E. De Pellegrin, S. Mukherjee, A. Ahrabian, S. Wang, and A. Wallace, "Radiate: A radar dataset for automotive perception in bad weather," in *Proc. IEEE Int. Conf. Robot. Autom.* IEEE, 2021, pp. 1–7.