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

SPS-2025-00

M.Sc. Thesis

Computationally-Efficient Sparsity-Aware Occupancy Grid Mapping for Automotive Driving

Frank Harraway B.Eng.

Faculty of Electrical Engineering, Mathematics and Computer Science **Delft University of Technology**

Computationally-Efficient Sparsity-Aware Occupancy Grid Mapping for Automotive Driving

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Electrical Engineering

by

Frank Harraway B.Eng.
born in Johannesburg, South Africa

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 **"Computationally-Efficient Sparsity-Aware Occupancy Grid Mapping for Automotive Driving"** by **Frank Harraway B.Eng.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 03/09/25

Chairperson: _____
Dr. G. Joseph

Advisors: _____
Ir. P. Zhai

_____
Dr. A.Pandharipande

Committee Members: _____
Prof.dr.ir. A.J. van der Veen

_____
Dr. H. Caesar

Abstract

Occupancy maps are used in automotive driving applications to understand the scene around the vehicle using data from sensors like LiDAR and/or radar on vehicles. In state-of-the-art work, pattern-coupled sparse Bayesian learning (PCSBL) was used to estimate the occupancy map by leveraging spatial dependencies across grids in the map for both single modalities and the fusion of multiple modalities. The PCSBL method, however, has high computational complexity, making real-time implementation challenging for large-scale grid maps. To address this limitation, we propose several methods to improve the computational efficiency of PCSBL while maintaining mapping accuracy. First, we utilize a precomputed lookup table to accelerate selection matrix construction. Second, we implement adaptive resolution reduction based on sensor measurements. Third, we develop two novel methods that exploit the narrow angular interactions between measurements and the map regions to enhance computational efficiency. The first method partitions measurements into spatially disjoint submaps that enable parallel processing. The second method exploits the angular structure to impose a block structure on the selection matrix, reducing computational overhead. Experiments on the nuScenes and RADIATE public datasets show that the presented methods reduce computational costs compared to the benchmark PCSBL and fusion-based PCSBL methods while preserving detection accuracy.

Acknowledgments

I would like to express my gratitude to those who have contributed to the successful completion of this thesis.

I would like to thank my supervisor Dr. G. Joseph. Throughout the research process of my thesis, I am grateful that Dr. G. Joseph always made time to support me throughout the research and writing process and provided thoughtful feedback that strengthened my reasoning and helped me refine this work. I would also like to express my appreciation to Ir. P. Zhai for his generous availability in answering technical questions and for his invaluable assistance with the writing process of both the conference and journal publications that emerged from this research. I would also extend my gratitude to Dr. A. Pandharipande for his contributions to my conference and journal publications, as well as his guidance regarding my future endeavours. Working with these dedicated faculty members has made the research process both enriching and thoroughly enjoyable.

Last but not least, I would like to thank my parents who wholeheartedly supported me in my decision to pursue this master's degree. I will forever be grateful to my supporting, loving parents.

Frank Harraway B.Eng.
Delft, The Netherlands
03/09/25

Contents

List of Figures

List of Tables

1 Introduction

With the improvement in sensor technology, algorithmic and computational advances, perception tasks have become viable for autonomous vehicles. The deployment of Advanced Driver-Assistance Systems (ADAS). With each step toward higher levels of automation, the responsibility for perception, decision-making, and control is shifting from the human driver to the vehicle itself. In fully autonomous systems at the most stringent requirement level, vehicles must perceive, understand, and react to dynamic and unpredictable environments without human intervention [1, 2].

Giving control to vehicles, which transport humans and have the risk of colliding with other humans or objects, raises the stakes and constraints to which this technology has to perform. In order for autonomous vehicles to perform, they require an accurate perception of their environment, which can be created using sensors such as LiDAR, radar, or cameras [3]. This perception must be built in a timely manner. To illustrate the importance, for a vehicle travelling at a modest speed of 40 km/h or 11.1 m/s. If a pedestrian suddenly enters the vehicle's path 10 m ahead, the system has less than 0.9 seconds to update its environmental perception, plan an avoidance maneuver, and execute it. This time requirement is exaggerated at higher speeds and when vehicles are travelling in opposite directions (potentially towards each other). Therefore, ADAS needs an accurate perception as fast as possible, allowing the ADAS to react to its environment in time to prevent collisions of any kind that happen daily in unknown dynamic scenarios. Any delay or failure in environmental understanding may result in unsafe behaviour.

The occupancy grid framework is a prominent method for representing environments in robotics [4]. It involves dividing the space into a regular grid of cells, where each cell is assigned a probability indicating the likelihood of it being occupied, based on input from sensor data. This mapping technique has been successfully used in a range of robotic tasks, including multi-sensor data integration [5, 6], navigation and motion planning [7], building maps while localizing the robot simultaneously [8], and tracking dynamic objects in the environment [9].

1.1 Occupancy Grid Mapping

The occupancy grid mapping (OGM) framework discretises continuous space into discrete cells within a grid containing N_x rows and N_y columns. Occupancy mapping algorithms assign each cell a discrete occupancy state as either occupied or free. A 2D occupancy grid can be represented as a flattened vector $\boldsymbol{x} \in \{0, 1\}^N$ where $N = N_x \times N_y$ and zero and one refer to free and occupied cells. An example occupancy grid is shown in Fig. 1.1.

The occupancy state of each cell is determined through the processing and inter-

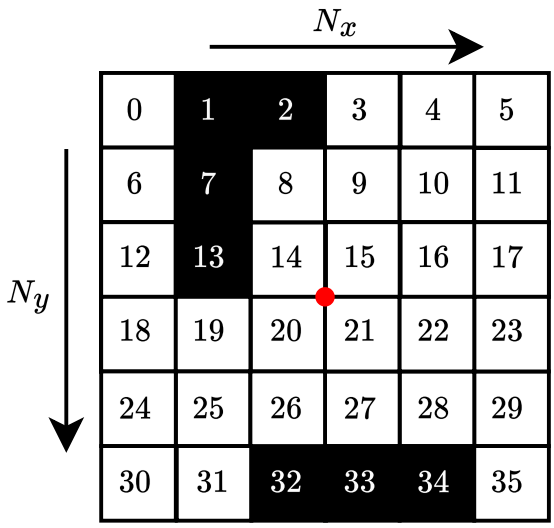


Figure 1.1: Occupancy grid of the environment around the ego vehicle sensor origin (red dot). Occupied cells are shown in black and free cells in white, with index values displayed for the flattened occupancy map.

pretation of sensor measurements. Different sensor modalities provide varying types of information about the environment, each with distinct characteristics in terms of measurement principles, detection capabilities, and associated uncertainties. The choice and configuration of sensors, therefore, directly influence the accuracy and reliability of the resulting occupancy grid representation.

1.2 Sensor Modalities

Using sensors to build a perception of the environment around the ego vehicle is a difficult task. LiDAR, radar and camera modalities each have their own strengths and weaknesses which can contribute a more holistic perception of the environment.

LiDAR is a sensor that can construct an accurate 3D point cloud by emitting laser pulses and measuring the time it takes for the light scatter from surrounding objects to reflect back to calculate the distance. LiDAR is able to emit thousands of pulses per second, allowing for dense and accurate point clouds representing the shape of objects in the line of sight with high detail. However, this reliance on light propagation introduces several limitations. Performance deteriorates under adverse weather conditions—fog, rain, and airborne dust scatter laser pulses before they reach physical objects, reducing measurement accuracy and range. Surface properties also pose challenges where dark, specular, or transparent materials, such as black vehicles or glass panels, reflect insufficient light back to the sensor [3], reducing detection performance. LiDAR technology remains substantially more expensive than alternative sensor modalities.

Radar, instead of light, emits microwave or radio pulses and measures the amount of time it takes for returning pulses to calculate distances. Radar provides exceptional robustness in adverse conditions, operating reliably through fog, rain and dust where

LiDAR fails. It provides velocity measurements via the Doppler effect for accurate motion tracking and offers a longer range than LiDAR. However, radar suffers from limited angular resolution, lower resolution, and suffers from ghost detections (false detections cause by indirect paths of returning electromagnetic pulses).

Stereo vision systems reconstruct depth by analyzing disparities between two spatially separated cameras, offering dense depth maps at relatively low cost. However, they exhibit critical limitations for automated driving. Depth estimation suffers in low-texture regions like asphalt or uniform walls due to poor visual correspondence matching. Performance degrades significantly in challenging lighting conditions, including low-light, high dynamic range, and glare environments. Depth accuracy decreases with distance, limiting long-range perception, while adverse weather conditions like rain or fog further reduce reliability. Despite avoiding LiDAR costs, stereo systems require intensive real-time computation and lack the robustness necessary for safety-critical autonomous driving applications.

In this project, camera modalities are excluded due to the reliability factors mentioned; instead, LiDAR and radar are used separately and fused together to create more robust environmental perception.

1.3 Problem Statement

A widely used model-based approach in OGMs is the inverse sensor model (ISM) [10], which assumes that each cell in the occupancy map is independent and uses ray casting principles to inform occupancy states. This independence assumption enables fast processing, with ray casting being the most computationally intensive component of the algorithm. Various techniques have been developed to further accelerate ray casting. Hierarchical structures like octrees and kd-trees reduce the number of cells traversed during computation [11–13], while approximate table-driven methods from localization applications provide constant-time distance estimates at the expense of quantization error [14]. Although originally developed for localization, these table-driven approaches can be applied to OGM with the same performance benefits. However, the independence assumption that enables ISM's computational efficiency also introduces problems: the method suffers from conflicts caused by inconsistent measurements and fails to exploit spatial structure in OGMs.

An alternative approach employs kernel-based methods, such as Gaussian process OGMs [15], which capture spatial dependencies. Early implementations were computationally intensive. Subsequent work in [16] addressed this by introducing test-data octrees, among other optimizations, to reduce computational load. Test-data octrees differ from traditional octrees (3D) and quadtrees (2D) used in OGMs for storage compression [17–20]—they dynamically decrease resolution as a preprocessing step based on sensor measurements, thereby reducing computational load. Building on these computational improvements, the Bayesian generalized kernel (BGK) [21, 22] method further accelerates Gaussian process OGMs by incorporating optimizations from [16, 23]. Despite these advances, Gaussian process OGMs still do not effectively handle the inherent sparsity in occupancy maps.

A sparsity-aware OGM model was introduced in [24], where pattern-coupled sparse

Bayesian learning (PCSBL) [25] was applied to estimate block-sparse occupancy maps by exploiting spatial structure using a single modality. This work was extended to multi-modal fusion in [6]. While single modality PCSBL for OGM outperforms Gaussian process OGMs and ISM in accuracy, its high computational complexity limits its application to large OGMs, with the multi-modal fusion version being even more computationally expensive. Therefore, this work aims to reduce the computational complexity of the PCSBL methods for OGM.

In addition to model-based algorithms, deep learning-based approaches have also been investigated in the literature. By leveraging large-scale autonomous driving sensor datasets [26,27], deep learning models can be trained to infer occupancy grid maps with classification or semantic information through learnable networks, using either single-modality [28–38] or multi-modal sensor input. The computational complexity is then dependent on the depth, layer sizes, and types of layers. However, deep learning models typically require large annotated datasets, face challenges in generalization, and often necessitate retraining when datasets or sensor types change. Moreover, they frequently struggle with generating occupancy maps of varying sizes and resolutions, estimating confidence in the maps, and handling sensor unreliability. Some studies have partially addressed these issues—for instance, multi-scale maps are generated using coarse-to-fine query structures [39–41], though they still require full-size feature computation. Other approaches estimate uncertainty alongside occupancy mapping [28, 37, 42]. Thus, we work to improve the computational efficiency of the physics-informed PCSBL, which does not require annotated data.

1.4 Project Contributions

In this work, we reduce the computational complexity of PCSBL [24] for OGMs for LiDAR, radar, and the fusion of both modalities [6]. Our main contributions are as follows:

- *Acceleration techniques:* We present three acceleration techniques for PCSBL:
    1. *Ray lookup table:* We reduce the computation required to construct the selection matrix. Using K-d trees, precomputed ray traversals are accessed using a nearest neighbor search, which are then used to efficiently compute the sparse selection matrix.
    2. *Test-Data quadtrees:* We extend the use of test-data quadtrees from [16] to PCSBL. These quadtrees provide a scene-dependent adaptive resolution, reducing the number of unknowns that need to be solved. To handle the resulting inconsistent cell sizes, we modify PCSBL to ensure compatibility.
    3. *Measurement model exploitation:* We leverage the unique structure of the measurement model, where measurements influence only narrow angular regions of the map. We introduce two methods: the partition and overlap (PO) method, which splits measurements into submaps for parallel processing, and the region-based cell permutation (CP) method, which reorganizes the measurement matrix into a block structure using angular patterns for faster computations.

- *Evaluation on real-world datasets:* We evaluate our acceleration techniques on the nuScenes and RADIATE public datasets [27, 43] by comparing them with benchmark PCSBL [6, 24], BGK [21], and ISM [8]. For single-modality PCSBL, run time is reduced from 12.9 s to 0.47 s (27x speedup), while multi-modal fusion was accelerated from 14.6 s to 0.7 s (21x speedup), with no reduction in map accuracy.

1.5 Outline

The report is organised as follows:

Chapter 2 - Introduces the background knowledge on occupancy grid mapping relating to single modal PCSBL [24] and multi-modal fusion PCSBL [6].

Chapter 3 describes the novel computational efficiency improvements applied to PCSBL, as well as methods extended from alternative OGM approaches.

Chapter 4 - 'Evaluations and Results' presents the real-world LiDAR and radar datasets used to access sensor modalities and to construct ground truths. It also explains the metrics used in this project and justifies the parameters used in the accelerated PCSBL methods, and shows the quantitative and qualitative results comparing the accelerated methods to the benchmark algorithms.

Chapter 5 'Conclusion and Future Work' summarises the accelerations and the findings of the accelerated methods, and provides insights on future work relating to computational and occupancy map accuracy improvements.

1.6 Publications from this Thesis

Conference Paper

- **Frank Harraway**, Peiyuan Zhai, Geethu Joseph, and Ashish Pandharipande, "Computationally-Efficient Sparsity-Aware Occupancy Grid Mapping for Automotive Driving," accepted by IEEE Sensors Conference, Oct 2025.

Journal Paper

- **Frank Harraway**, Peiyuan Zhai, Geethu Joseph, and Ashish Pandharipande, "Accelerated Pattern-Coupled Sparse Bayesian Learning for Automotive Occupancy Mapping," IEEE Sensors Journal, Submitted Aug 2025.

Occupancy Grid Mapping Preliminaries

2

To establish a foundation for understanding state-of-the-art PCSBL methods and their application to occupancy grid mapping (OGM), this section first explains the input preprocessing. The per-sensor measurement models that transform sensor measurements into PCSBL-compatible format are described, followed by the algorithmic frameworks underlying single-modal PCSBL for OGM [24]. Finally, multi-modal fusion sensor measurement models and algorithms are presented [6]. Given the computational challenges this project addresses, existing optimization strategies successfully applied to PCSBL implementations are also discussed.

2.1 Preprocessing steps of LiDAR and radar modalities

This section concerns preprocessing the sensor modalities to produce a 2D point cloud containing M (M_{L}, M_{R} for LiDAR, radar) points, which can be used to estimate occupancies for 2D OGM. The resulting point cloud is represented as:

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{p}_0^T \\ \boldsymbol{p}_1^T \\ \vdots \\ \boldsymbol{p}_{M-1}^T \end{bmatrix} = \begin{bmatrix} p_{x,0} & p_{y,0} \\ p_{x,1} & p_{y,1} \\ \vdots & \vdots \\ p_{x,M-1} & p_{y,M-1} \end{bmatrix} \in \mathbb{R}^{M\times 2} \tag{2.1}$$

where the ith point $\boldsymbol{p}_i^\top$ has coordinates $(p_{x,i}, p_{y,i})$ in the x-y plane.

2.1.1 LiDAR Preprocessing

In this project, LiDAR data is accessed in a preprocessed 3D point cloud format. The raw LiDAR data contains numerous reflections from the road surface. These ground-level reflections are undesirable since the ground plane provides no meaningful information about occupancy. Consequently, points below a specified height threshold are filtered out during preprocessing.

Similarly, points above the height of the ego vehicle and points outside the bounds of the occupancy grid are also removed. In this project, for computational efficiency, the occupancy grid estimates are in 2D. Therefore, the point cloud must be projected accordingly. The z-axis (height dimension) is discarded, projecting all remaining points onto a single plane to create a bird's-eye view representation for 2D occupancy grid mapping.

2.1.2 Radar Preprocessing

The radar data is provided as range-azimuth images without Doppler or elevation information. Since these images are susceptible to noise, the constant false alarm

rate (CFAR) technique is applied to detect valid points. The CFAR algorithm adapts its detection threshold based on local noise estimates [44]. A detailed description is provided in Appendix B. Since range-azimuth images already exist in a 2D plane, only minimal preprocessing is required after CFAR detection. Points outside the occupancy grid boundaries are filtered out, while the remaining detections can be directly used for occupancy mapping.

2.1.3 Modality Fusion Preprocessing

When using LiDAR and radar together, each sensor has its own coordinate system because of the distinct mounting points on the ego vehicle. The preprocessed point clouds from subsection 2.1.1 and subsection 2.1.2 need to be rotated and translated to align them to a common coordinate frame. Details on how we handle this registration are given in Appendix C.

2.2 PCSBL for OGM

PCSBL for OGM consists of two fundamental components: encoding sensor information into a usable format through sensor models, and estimating grid cell occupancies using this encoded information. This section explains how sensor information is encoded for the single-modality PCSBL [24] approach, followed by the occupancy estimation process.

2.2.1 Single Modality PCSBL for OGM

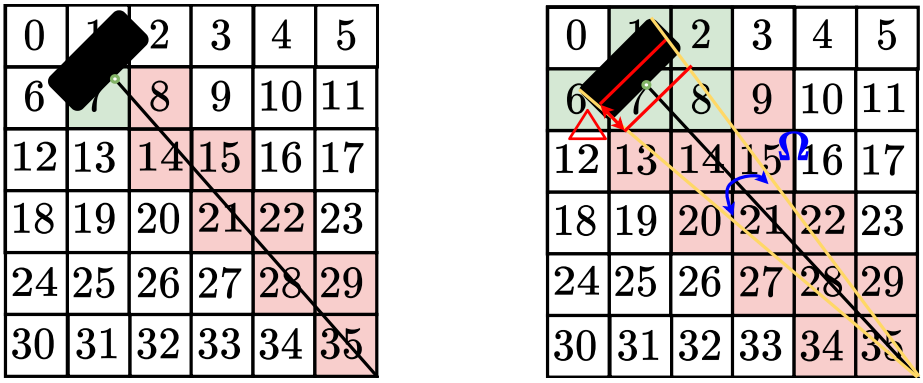
Occupancy grid mapping aims to estimate the occupancy state of a vehicle's surrounding environment from sensor measurements (reflection points). To this end, the environment is discretized into a 2D grid with N cells. The collection of all cell occupancies is represented by the vector $\boldsymbol{x} \in \{0,1\}^N$.

2.2.1.1 LiDAR Measurement Model

Light is emitted by a LiDAR sensor, reflected off an object, and then recorded at the origin. There are no obstructions between the object and the sensor origin. Two pieces of information can be inferred from each LiDAR reflection as seen by the red and green cells in Fig. 2.1a. The cell from which the mth LiDAR reflection originates is occupied, and the set of cells $\mathcal{F}_m$ that the LiDAR beam to the mth reflection intersects are free. The occupied cells are expressed in a linear equation as

$$\sum_{k \in \mathcal{O}_m} \boldsymbol{x}_{\mathrm{L}}[k] = y_{\mathrm{occ}}, \tag{2.2}$$

where $\mathcal{O}_m$ is a set containing the single index of the cell that the LiDAR reflection occupies. For example, in Fig. 2.1a, the occupied index is given by the set $\mathcal{O}_m \in \{7\}$. The selected cell is then equated to y_{occ}. The value of y_{occ} is typically set to one, as it



(a) LiDAR Sensor Model (b) Radar Sensor Model

Figure 2.1: LiDAR measurements showing inferred occupied (green) and free (red) cells for mth reflection; for sparse radar point clouds, cells within a Δ-wide region around each reflection are inferred occupied, and cells within the Ω-shaped beam preceding the reflection are inferred free. The cells show the indices of the flattened occupancy vector.

is expected that $x[k]$ is occupied. Similarly, the set of free cells can be expressed as

$$\sum_{k \in \mathcal{F}_m} \boldsymbol{x}_{\mathrm{L}}[k] = y_{\mathrm{free}}, \tag{2.3}$$

where y_{free} is typically set to zero as the cells are inferred to be free. For example, in Fig. 2.1a, the set of free cells are given by $\mathcal{F}_m \in \{8, 14, 15, 21, 22, 28, 29, 35\}$. Each LiDAR reflection provides two linear equations. Therefore M_{L} LiDAR reflections can represented as $2M_{\mathrm{L}}$ linear equations of $\boldsymbol{x}$ in

$$\boldsymbol{y}_{\mathrm{L}} = \boldsymbol{A}_{\mathrm{L}} \boldsymbol{x}_{\mathrm{L}} + \boldsymbol{n}_{\mathrm{L}}, \tag{2.4}$$

where $\boldsymbol{n}_{\mathrm{L}} \sim \mathcal{N}(\boldsymbol{0}, \sigma_{\mathrm{L}}^2 \boldsymbol{I})$ is distributed noise with unknown variance σ_{L}^2. The selection matrix $\boldsymbol{A}_{\mathrm{L}} \in \{0,1\}^{2M_{\mathrm{L}}}$ is defined by (2.2) and (2.3), where the support of the $(2m-1)$th row of the selection matrix is $\mathcal{O}_m$ and that of the $2m$th row is $\mathcal{F}_m$.

2.2.1.2 Radar Measurement Model

In datasets [27, 45], radar point clouds are considerably sparser than LiDAR, leading to lower-quality occupancy maps. To address this, [24] applied an artificial enhancement inspired by the Inverse Sensor Model [8]. For each radar point, a conical beam of width Ω was constructed terminating in a region of width Δ around the reflection point Fig. 2.1b. The terminal region (green cells) is denoted by $\mathcal{O}_m$ (occupied), and the beam path cells (red cells) by $\mathcal{F}_m$ (free).

Unlike LiDAR, which marks a single cell as occupied, the radar model labels all cells in $\mathcal{O}_m$ to better represent the obstacles and address data sparsity. Therefore, for the mth radar measurement, the set of cells around the measurement is formulated into the linear equation

$$\sum_{k \in \mathcal{O}_m} \boldsymbol{x}_{\mathrm{R}}[k] = y_{\mathrm{occ}} |\mathcal{O}_m|, \tag{2.5}$$

where $|\mathcal{O}_m|$ is the number of elements within the set, which is used to weight the measurement vector value. For example, in Fig. 2.1b the occupied set is given by $\mathcal{O}_m \in \{1, 2, 6, 7, 8\}$ with weight $|\mathcal{O}_m| = 5$ on the measurement vector value. The beam leading up to the measurement has the linear equation

$$\sum_{k \in \mathcal{F}_m} \boldsymbol{x}_{\mathrm{R}}[k] = y_{\mathrm{free}}. \tag{2.6}$$

This is once again expressed as

$$\boldsymbol{y}_{\mathrm{R}} = \boldsymbol{A}_{\mathrm{R}} \boldsymbol{x}_{\mathrm{R}} + \boldsymbol{n}_{\mathrm{R}}, \tag{2.7}$$

where each element of $\boldsymbol{n}_{\mathrm{R}} \sim \mathcal{N}(\boldsymbol{0}, \sigma_{\mathrm{R}}^2 \boldsymbol{I})$ distributed noise with unknown variance σ_{R}^2.

Algorithm 1 Selection matrix construction for LiDAR and radar

Input: Reflection point coordinates $\{\boldsymbol{P}\}$
Output: Selection matrix $\boldsymbol{A}$ and measurement vector $\boldsymbol{y}$
for each measurement $m = 1, \ldots, M$ **do**
  $\mathcal{O}_m \leftarrow$ occupied cell(LiDAR) or occupied cells(radar) at $\boldsymbol{p}_m^\top$
  $\mathcal{F}_m \leftarrow$ free cells along ray (LiDAR) or along beam (radar) preceding $\boldsymbol{p}_m^\top$
  Set $\boldsymbol{A}[2m,\, n] = 1 \quad \forall\, n \in \mathcal{O}_m$
  Set $\boldsymbol{A}[2m+1,\, n] = 1 \quad \forall\, n \in \mathcal{F}_m$
  $y[2m] \leftarrow y_{\mathrm{occ}} |\mathcal{O}_m|$
  $y[2m+1] \leftarrow y_{\mathrm{free}}$
end for

The single modality PCSBL algorithm applies identically to both LiDAR and radar sensing modalities; we omit the subscripts L and R for brevity and use the general notation $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{A}$ and σ^2. PCSBL aims to reconstruct the unknown map $\boldsymbol{x}$ for both LiDAR and radar from $\boldsymbol{y}$ and $\boldsymbol{A}$.

2.2.2 Single-Modality PCSBL

The PCSBL algorithm for a single modality (LiDAR or radar) imposes a two-layer hierarchical prior distribution on the unknown $\boldsymbol{x}$. In the first layer, $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{D}^{-1})$ follows a zero-mean is modeled as a zero-mean Gaussian vector with a diagonal precision matrix $\boldsymbol{D}$. To exploit the block structure, its nth diagonal element is

$$D[n, n] = \alpha[n] + \beta \sum_{m \in \mathcal{L}_n} \alpha[m], \tag{2.8}$$

where $\alpha[n]$ is the nth cell precision hyperparameter, $\beta \in [0, 1]$ is the coupling parameter, and $\mathcal{L}_n$ is the direct neighbor set of the nth cell is. When the precision $\alpha[n]$ of the nth cell is high, the occupancy value $x[n]$ is likely to be close to zero, indicating it is a free cell. Since the precisions are coupled via β, they are spatially correlated, encouraging adjacent cells to agree on the occupancy state. In the second layer, the cell precision hyperparameters $\boldsymbol{\alpha}$ and noise precision σ^{-2} are assigned Gamma hyperpriors

$$\alpha \sim \mathrm{Gamma}(a, b) \quad \text{and} \quad \sigma^{-2} \sim \mathrm{Gamma}(c, d). \tag{2.9}$$

To estimate $\boldsymbol{x}$, the unknown hyperparameters $\boldsymbol{\alpha}$ and σ^2 are estimated iteratively through the expectation-maximization (EM) algorithm [24, 25]. In every iteration, given the current hyperparameters, it first computes (Expectation (E) step) the posterior mean $\hat{\boldsymbol{\mu}}$ and covariance $\hat{\boldsymbol{\Phi}}$ of $\boldsymbol{x}$ as

$$\hat{\boldsymbol{\Phi}} = (\sigma^{-2}\,\boldsymbol{A}^\top \boldsymbol{A} + \boldsymbol{D})^{-1}, \quad \hat{\boldsymbol{\mu}} = \sigma^{-2}\,\hat{\boldsymbol{\Phi}}\,\boldsymbol{A}^\top \boldsymbol{y}, \tag{2.10}$$

and then updates the hyperparameters $\boldsymbol{\alpha}$, σ^2 to maximize the model likelihood given the measurements (Maximisation (M) step),

$$\alpha^{(t+1)}[n] = \frac{a}{0.5\,\hat{v}^{(t)}[n] + \beta \sum_{j\in\mathcal{L}_n} \hat{v}^{(t)}[j] + b}, \tag{2.11}$$

$$\begin{aligned}(\sigma^2)^{(t+1)} = \frac{1}{M+2c}(&\|\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{\mu}}^{(t)}\|_2^2 \\ &+ (\sigma^{-2})^{(t)} \sum_n \hat{\Phi}^{(t)}[n,n]\, D^{(t)}[n,n] + 2d),\end{aligned} \tag{2.12}$$

where $\boldsymbol{v}^{(t)} \in \mathbb{R}^n$ is the vector of second moments of entries of $\boldsymbol{x}$ given $(\sigma^{-2})^{(t)}$, $\boldsymbol{\alpha}^{(t)}$, and $\boldsymbol{y}$. Iterations continue until convergence; the final map estimate is obtained by thresholding $\hat{\boldsymbol{\mu}}$

$$\hat{x}[n] = \begin{cases} 0, & \hat{\mu}[n] < \tau \\ 1, & \hat{\mu}[n] \geq \tau \end{cases}. \tag{2.13}$$

Algorithm 2 Occupancy grid map estimation using single modality PCSBL

Input: Selection matrix $\boldsymbol{A}$, measurement vector $\boldsymbol{y}$
Parameters: coupling β, Gamma parameters $a, b, c, d > 0$
threshold τ, max iterations T
Output: Binary occupancy map $\hat{\boldsymbol{x}}$

$t \leftarrow 0, \quad \boldsymbol{\alpha}^{(0)} \leftarrow \mathbf{1}, \quad (\sigma^2)^{(0)} \leftarrow 0.5$
Precompute $\boldsymbol{A}^\top \boldsymbol{A}$ and $\boldsymbol{A}^\top \boldsymbol{y}$
repeat
  Update $\hat{\boldsymbol{\mu}}^{(t)}$ and $\hat{\boldsymbol{\Phi}}^{(t)}$ via (2.10) ▷ E-step
  Compute $\boldsymbol{D}^{(t)}$ via (2.8)
  Update $\boldsymbol{\alpha}^{(t+1)}, (\sigma^2)^{(t+1)}$ via (2.11),(2.12) ▷ M-step
  $t \leftarrow t + 1$
until convergence or $t > T$
Compute $\hat{\boldsymbol{x}}$ by thresholding using τ

2.2.2.1 Zero Column PCSBL Optimisation

An observation was made in [46], the EM steps could be made more efficient based on the sparsity of the measurements. When cells have no measurements or ray intersections, the selection matrix $\boldsymbol{A}$ contains zero columns at corresponding indices. These Z zero columns can be exploited to reduce computational complexity without approximation.

Zero columns in $\boldsymbol{A}$ produce zero rows and columns in $\boldsymbol{A}^\top\boldsymbol{A}$. This is important because in the E-step (2.10) only diagonal elements from $\boldsymbol{D}$ remain non-zero in the matrix $\boldsymbol{Q} = \boldsymbol{A}^\top\boldsymbol{A} + \boldsymbol{D}$:

$$\boldsymbol{Q} = \begin{bmatrix} * & * & 0 & * & 0 \\ * & * & 0 & * & 0 \\ 0 & 0 & D[2,2] & 0 & 0 \\ * & * & 0 & * & 0 \\ 0 & 0 & 0 & 0 & D[4,4]. \end{bmatrix}$$

This block structure enables us to partition $\boldsymbol{Q}$ and invert it more efficiently. The dense submatrix (marked by $*$) can be inverted independently of the isolated diagonal elements. We achieve this separation using a permutation matrix $\boldsymbol{P}_\mathcal{Z}$. This matrix permutes $\boldsymbol{Q}$ such that the dense and diagonal parts are separated. Here, $\mathcal{Z}$ contains the indices of zero columns in $\boldsymbol{A}$. In the example above, we have $\mathcal{Z} = \{2, 4\}$. The permuted structure is represented as:

$$\boldsymbol{P}_\mathcal{Z}\boldsymbol{Q}\boldsymbol{P}_\mathcal{Z}^\top = \begin{bmatrix} \boldsymbol{Q}_{\mathcal{Z}'} & \cdots \\ \vdots & \boldsymbol{Q}_\mathcal{Z} \end{bmatrix}$$

where $\boldsymbol{Q}_{\mathcal{Z}'}$ represents the dense submatrix corresponding to the complement of $\mathcal{Z}$, and $\boldsymbol{Q}_\mathcal{Z}$ contains the isolated diagonal elements. The isolated matrix $\boldsymbol{Q}_{\mathcal{Z}'}$ can now be inverted separately of the diagonal in $\boldsymbol{Q}_\mathcal{Z}$. The inverse is reconstructed as:

$$\boldsymbol{Q}^{-1} = \boldsymbol{P}_\mathcal{Z}^\top \begin{bmatrix} \boldsymbol{Q}_{\mathcal{Z}'}^{-1} & \cdots \\ \vdots & \boldsymbol{Q}_\mathcal{Z}^{-1} \end{bmatrix} \boldsymbol{P}_\mathcal{Z}$$

The diagonal matrix $\boldsymbol{Q}_\mathcal{Z}$ inverts with $\mathcal{O}(Z)$ complexity, reducing overall complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}((N-Z)^3) + \mathcal{O}(N^2) + \mathcal{O}(Z)$ where $\mathcal{O}(N^2)$ is the computational cost to permute the matrix to the desired form. This optimization's effectiveness is scene-dependent on which cells receive information.

2.2.2.2 Focus Regions

Additionally, another attempt to reduce the complexity is to reduce the number of cells in the map in a structured way. Certain automotive datasets integrate GPS positioning with digital road maps. This combined information enables selective occupancy grid estimation, as demonstrated in [24], where road topology determines which grid cells require estimation versus those that can be excluded from the mapping process.

The use of the focus mask in Fig. 2.2 allowed for the reduction of occupancies that need to be calculated from 6400 to 4259, a 33% decrease in the number of cells.

2.3 Fusion PCSBL for OGM

Building on the single-modality PCSBL framework, this section explains how LiDAR and radar measurement models and algorithms are fused using PCSBL-based common sparse (CS) and common innovative sparse (CIS) approaches [6].

(a) (b)

Figure 2.2: Roadway and walkway shown as a mask in (a) and combined into a single focus mask in (b) to reduce the number of occupancies that need to be computed

2.3.1 Fusion Measurement Model

PCSBL-based fusion, CS, and CIS rely on two separate measurement models relying on separate assumptions about the occupancy map.

2.3.1.1 CS Measurement Model

Common sparse assumes the LiDAR $\boldsymbol{x}_{\mathrm{L}}$ and radar maps $\boldsymbol{x}_{\mathrm{R}}$ correspond to the same map $\boldsymbol{x}$, leading to a model that concatenates the LiDAR and radar measurements

$$\boldsymbol{y}_{\mathrm{CS}} = \begin{bmatrix} \boldsymbol{y}_{\mathrm{L}} \\ \boldsymbol{y}_{\mathrm{R}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{\mathrm{L}} \\ \boldsymbol{A}_{\mathrm{R}} \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} \boldsymbol{n}_{\mathrm{L}} \\ \boldsymbol{n}_{\mathrm{R}} \end{bmatrix} = \boldsymbol{A}_{\mathrm{CS}} \boldsymbol{x} + \boldsymbol{n}_{\mathrm{CS}}, \tag{2.14}$$

to jointly estimate the map $\boldsymbol{x} \in \mathbb{R}^N$ where the linear equations for the LiDAR and radar from (2.4) (2.7) are vertically stacked.

2.3.1.2 CIS Measurement Model

Unreliable sensors or misalignment between the sensors could lead to significant inconsistencies $\boldsymbol{x}_{\mathrm{L}}$ and $\boldsymbol{x}_{\mathrm{R}}$, therefore, CIS introduces error variables

$$\boldsymbol{x}_{\mathrm{L}} = \boldsymbol{x}_{\mathrm{c}} + \boldsymbol{x}_{\Delta \mathrm{L}}, \quad \boldsymbol{x}_{\mathrm{R}} = \boldsymbol{x}_{\mathrm{c}} + \boldsymbol{x}_{\Delta \mathrm{R}}, \tag{2.15}$$

where $\boldsymbol{x}_c$ is the common part of the signal $\boldsymbol{x}_{\Delta \mathrm{L}}$ and $\boldsymbol{x}_{\Delta \mathrm{R}}$ are the error collectors for LiDAR and radar respectively. The CIS measurement model is then formulated as

$$\begin{aligned} \boldsymbol{y}_{\mathrm{CIS}} = \begin{bmatrix} \boldsymbol{y}_L \\ \boldsymbol{y}_R \end{bmatrix} &= \begin{bmatrix} \boldsymbol{A}_{\mathrm{L}} & \boldsymbol{A}_{\mathrm{L}} & 0 \\ \boldsymbol{A}_{\mathrm{R}} & 0 & \boldsymbol{A}_{\mathrm{R}} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_c \\ \boldsymbol{x}_{\Delta \mathrm{L}} \\ \boldsymbol{x}_{\Delta \mathrm{R}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{n}_{\mathrm{L}} \\ \boldsymbol{n}_{\mathrm{R}} \end{bmatrix}, \\ &= \boldsymbol{A}_{\mathrm{CIS}} \boldsymbol{x}_{\mathrm{CIS}} + \boldsymbol{n}_{\mathrm{CIS}}, \end{aligned} \tag{2.16}$$

where $\boldsymbol{A}_{\mathrm{CIS}} \in \mathbb{R}^{2(M_{\mathrm{L}}+M_{\mathrm{R}})3N}$ and $\boldsymbol{x}_{\mathrm{CIS}} \in \mathbb{R}^{3N}$. Only $\boldsymbol{x}_{\mathrm{c}}$ is used to estimate the occupancy grid map, and the error collectors are discarded. Once again using $\boldsymbol{A}_{\mathrm{CIS}}$ and $\boldsymbol{y}_{\mathrm{CIS}}$ we use PCSBL to estimate the occupancy map $\boldsymbol{x}_{\mathrm{c}}$ ($\boldsymbol{A}_{\mathrm{CS}}$, $\boldsymbol{y}_{\mathrm{CS}}$ and $\boldsymbol{x}$ in the case of CS).

2.3.2 Fusion PCSBL

The fusion methods [6] extend on the [24] method using the measurement models in (2.14) and (2.16).

The CS model in (2.16) allows us to directly apply the PCSBL algorithm. From the measurement model, we have

$$p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\alpha}, \sigma_{\mathrm{L}}^2, \sigma_{\mathrm{R}}^2) = \mathcal{N}(\boldsymbol{A}_{\mathrm{CS}}\boldsymbol{x}, \boldsymbol{C}),$$

where the corresponding noise covariance is

$$\boldsymbol{C} = \begin{bmatrix} \sigma_{\mathrm{L}}^2 \boldsymbol{I}_{2M_{\mathrm{L}}} & \boldsymbol{0} \\ \boldsymbol{0} & \sigma_{\mathrm{R}}^2 \boldsymbol{I}_{2M_{\mathrm{R}}} \end{bmatrix}. \tag{2.17}$$

Then, the posterior mean and covariance of the Gaussian $p(\boldsymbol{x}|\boldsymbol{y}; \boldsymbol{\alpha}, (\sigma_{\mathrm{L}}^2), (\sigma_{\mathrm{R}}^2))$ during the E-step is calculated as

$$\begin{aligned} \hat{\boldsymbol{\mu}}_{\mathrm{CS}} &= \hat{\boldsymbol{\Phi}}_{\mathrm{CS}}(\sigma_{\mathrm{L}}^{-2}\boldsymbol{A}_{\mathrm{L}}^{\top} + \sigma_{\mathrm{R}}^{-2}\boldsymbol{A}_{\mathrm{R}}^{\top})\boldsymbol{y}_{\mathrm{CS}}, \\ \hat{\boldsymbol{\Phi}}_{\mathrm{CS}} &= \left(\sigma_{\mathrm{L}}^{-2}\boldsymbol{A}_{\mathrm{L}}^{\top}\boldsymbol{A}_{\mathrm{L}} + \sigma_{R}^{-2}\boldsymbol{A}_{\mathrm{R}}^{\top}\boldsymbol{A}_{\mathrm{R}} + \boldsymbol{D}\right)^{-1}. \end{aligned} \tag{2.18}$$

During the M-step, the noise variances for each modality are updated as

$$\begin{aligned} (\sigma_{\mathrm{L}}^2)^{(t+1)} &= \frac{2d + \|\boldsymbol{y}_{\mathrm{L}} - \boldsymbol{A}_{\mathrm{L}}\boldsymbol{x}\|_2^2 + \mathrm{tr}(\boldsymbol{A}_{\mathrm{L}}^{\top}\boldsymbol{A}_{\mathrm{L}}\boldsymbol{\Phi}_{\mathrm{CS}}^{(t)})}{2c + 2M_{\mathrm{L}}}, \\ (\sigma_{\mathrm{R}}^2)^{(t+1)} &= \frac{2d + \|\boldsymbol{y}_{\mathrm{R}} - \boldsymbol{A}_{\mathrm{R}}\boldsymbol{x}\|_2^2 + \mathrm{tr}(\boldsymbol{A}_{\mathrm{R}}^{\top}\boldsymbol{A}_{\mathrm{R}}\hat{\boldsymbol{\Phi}}_{\mathrm{CS}}^{(t)})}{2c + 2M_{\mathrm{R}}}. \end{aligned} \tag{2.19}$$

and $\boldsymbol{\alpha}_{CS}$ is updated in the same was as single-modality PCSBL in (2.11).

Similarly, CIS also uses the PCSBL framework with the common component $\boldsymbol{x}_{\mathrm{c}}$ following a Gaussian prior with a coupled precision. In contrast, the LiDAR and radar error components $\boldsymbol{x}_{\Delta\mathrm{L}}$ and $\boldsymbol{x}_{\Delta\mathrm{R}}$ are assumed to follow independent zero-mean Gaussian priors. So, $\boldsymbol{x}_{\mathrm{CIS}} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{D}_{\mathrm{CIS}}^{-1})$ where the precision matrix is given by

$$D_{\mathrm{CIS}}[n,n] = \begin{cases} \alpha_{\mathrm{c}}[n'] + \beta \sum\limits_{j \in \mathcal{L}_{n'}} \alpha_{\mathrm{c}}[j], & \text{if } n' \leq N, \\ \alpha_{\Delta\mathrm{L}}[n' - N], & \text{if } N < n' \leq 2N, \\ \alpha_{\Delta\mathrm{R}}[n' - 2N], & \text{if } 2N < n' \leq 3N. \end{cases} \tag{2.20}$$

The entries of the precision vectors $\boldsymbol{\alpha}_{\mathrm{c}}$, $\boldsymbol{\alpha}_{\Delta L}$ and $\boldsymbol{\alpha}_{\Delta R}$ follow the Gamma prior in (2.9), with shape parameter a as $a_{\mathrm{c}}, a_{\mathrm{L}}$, and a_{R}, respectively. Then, the posterior mean and covariance of $\boldsymbol{x}_{\mathrm{CIS}}$ in the E-step are calculated as

$$\begin{aligned} \boldsymbol{\mu}_{\mathrm{CIS}} &= \hat{\boldsymbol{\Phi}}_{\mathrm{CIS}}\boldsymbol{A}_{\mathrm{CIS}}^{\top}\boldsymbol{C}^{-1}\boldsymbol{y}_{\mathrm{CIS}}, \\ \hat{\boldsymbol{\Phi}}_{\mathrm{CIS}} &= \left(\boldsymbol{A}_{\mathrm{CIS}}^{\top}\boldsymbol{C}^{-1}\boldsymbol{A}_{\mathrm{CIS}} + \boldsymbol{D}_{\mathrm{CIS}}\right)^{-1} \end{aligned} \tag{2.21}$$

The M-step update is given by

$$
\begin{aligned}
(\sigma_{\mathrm{L}}^2)^{(t+1)} &= \frac{1}{2c+2M_{\mathrm{L}}}(2d + \|\boldsymbol{y}_{\mathrm{L}} - \boldsymbol{A}_{\mathrm{L}}(\boldsymbol{x}_c + \boldsymbol{x}_{\Delta\mathrm{L}})\|_2^2 \\
&\quad + \mathrm{tr}(\boldsymbol{A}_{\mathrm{CIS}}^T \boldsymbol{A}_{\mathrm{CIS}} \hat{\boldsymbol{\Phi}}_{\mathrm{CIS}}^{(t)})), \\
(\sigma_{\mathrm{R}}^2)^{(t+1)} &= \frac{1}{2c+2M_{\mathrm{R}}}(2d + \|\boldsymbol{y}_{\mathrm{R}} - \boldsymbol{A}_{\mathrm{R}}(\boldsymbol{x}_c + \boldsymbol{x}_{\Delta\mathrm{R}})\|_2^2 \\
&\quad + \mathrm{tr}(\boldsymbol{A}_{\mathrm{CIS}}^T \boldsymbol{A}_{\mathrm{CIS}} \hat{\boldsymbol{\Phi}}_{\mathrm{CIS}}^{(t)})).
\end{aligned}
$$

and $\boldsymbol{\alpha}_{\mathrm{c}}$ is updated using (2.11) as in single-modality PCSBL, while the remaining $\boldsymbol{\alpha}_{\Delta\mathrm{L}}$ and $\boldsymbol{\alpha}_{\Delta\mathrm{R}}$ are updated using (2.11) with $\beta = 0$.

2.3.3 PCSBL Complexity Considerations

Having introduced the PCSBL algorithm, we next present a complexity analysis for both single and multi-modal PCSBL.

2.3.3.1 Single Modality PCSBL

The main computational challenge in single-modality PCSBL arises from the matrix inversion in the E-step (2.10). The precomputation $\boldsymbol{A}^\top \boldsymbol{A}$ requires $\mathcal{O}(MN^2)$ complexity, while the matrix inversion step has $\mathcal{O}(N^3)$ complexity. Since the number of EM iterations repeats the E-step T times, the $\mathcal{O}(N^3)$ matrix inversion complexity is further magnified, making PCSBL computationally expensive for large-scale problems. Additional costs include constructing the selection matrix at $\mathcal{O}(MN)$ and computing ray intersections at $\mathcal{O}(LM)$, where L is the average number of cells intersected per ray.

2.3.3.2 Fusion PCSBL

The CS fusion method retains the same structure and computational characteristics as the single-modality PCSBL formulation in (2.18). However, it adds overhead from the separate computation of $\boldsymbol{A}_{\mathrm{L}}^\top \boldsymbol{A}_{\mathrm{L}}$ and $\boldsymbol{A}_{\mathrm{R}}^\top \boldsymbol{A}_{\mathrm{R}}$, as well as from the additional noise update in (2.19). CIS fusion includes this same overhead and further increases the cost due to a three-times larger selection matrix used for the error collectors. This impacts both the E-step in (2.21) and the noise update in (2.22).

The overall computational complexities, along with those of ISM [10] and BGK [21], are summarized in Table 2.1. As evident from Table 2.1, PCSBL is the most computationally demanding algorithm among model-based mapping methods, despite its superior map accuracy. Therefore, our goal is to optimize the matrix computations in both single- and multi-modal PCSBL, enabling fast and scalable occupancy mapping by leveraging the inherent structure of the measurement model, without compromising accuracy.

Table 2.1: Order of Complexity Comparison

Algorithm	Computational complexity
Single modality LiDAR	$\mathcal{O}(TN^3 + M_{\mathrm{L}}N^2)$
Single modality radar	$\mathcal{O}(TN^3 + M_{\mathrm{R}}N^2)$
CS and CIS fusion	$\mathcal{O}(TN^3 + (M_{\mathrm{L}} + M_{\mathrm{R}})N^2)$
BGK	$\mathcal{O}(N \log(M_{\mathrm{L}}))$
ISM	$\mathcal{O}(LM_{\mathrm{L}})$

2.4 Summary

This chapter has established the foundation for occupancy grid mapping using PCSBL methods, covering both single modality and multi-modal fusion approaches. We began by detailing the essential preprocessing steps required for LiDAR and radar sensor modalities, including ground filtering, CFAR detection, and coordinate frame alignment necessary for effective sensor fusion, where sensor coordinate frames must be equivalent.

For both single and multi-modal PCSBL for OGM the sensor measurements are then used to construct the linear measurement system of equations through appropriate sensor models. We then present the PCSBL algorithm, demonstrating how it leverages this linear system of equations ($\boldsymbol{A}$ and $\boldsymbol{y}$) to estimate occupancy states using a two-layer hierarchical Bayesian model that exploits spatial coupling to recover block-sparse occupancy patterns. The PCSBL framework is then explained how it estimates the occupancy map $\boldsymbol{x}$ from the linear system of equations. The extension of the algorithm to the multi-modal is also outlined. The two distinct fusion approaches: CS fusion assumes identical maps across modalities, and CIS fusion, which introduces error collectors to handle sensor inconsistencies. Finally, we analyze the computational complexity of each method to understand its respective trade-offs.

The concepts presented in this chapter form the technical foundation upon which the subsequent computational improvements will be developed.

Computationally Efficient PCSBL

3

Several techniques are introduced to improve the computational efficiency of PCSBL for OGM for single modalities and the fusion of multiple modalities. PCSBL for OGM involves two primary stages: constructing the selection matrix for the linear system (2.4), and estimating the occupancy vector $\boldsymbol{x}$. An efficient method for constructing the selection matrix using a ray lookup table is introduced, test-data quadtrees are extended to PCSBL to reduce the number of occupancies to estimate, and two strategies manipulating the selection matrix are presented to accelerate the estimations of occupancies for PCSBL. The first, PO, parallelises the processing of sub-maps derived from measurements. The second, CP, improves the efficiency of matrix operations by permuting the columns $\boldsymbol{A}$ to exploit spatial structure.

The implementation proceeds by querying the ray lookup table with preprocessed measurements to extract relevant rays. These rays determine the adaptive quadtree resolution. The varying resolution in combination with PO or CP is then used to construct the exploited measurement matrix $\boldsymbol{A}$. The resulting selection matrix is fed into PCSBL to produce the occupancy map.

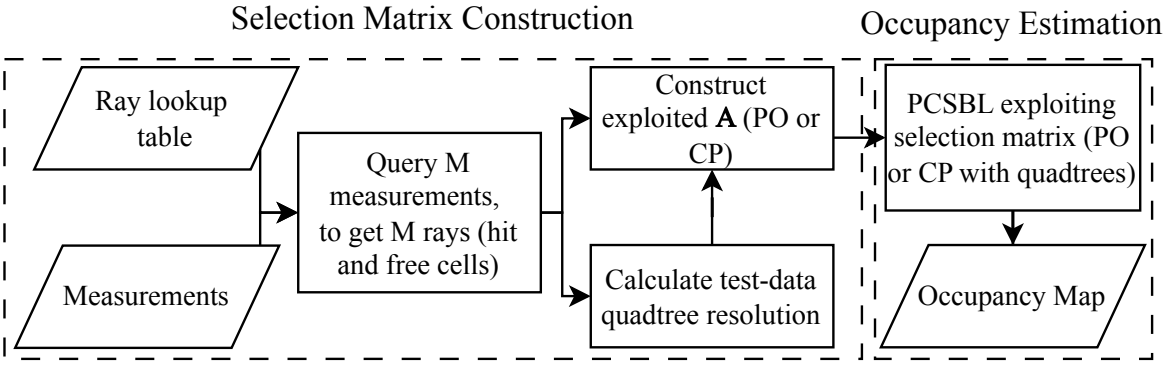
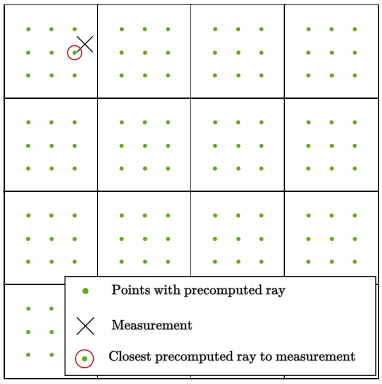


Figure 3.1: Processing pipeline of PCSBL acceleration methods.

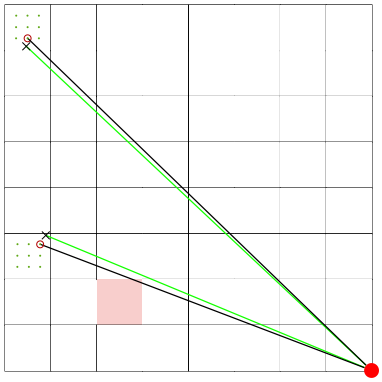
3.1 Ray Lookup Selection Matrix Construction

Building $\boldsymbol{A} \in \{0,1\}^{2M \times N}$, is computationally costly. Firstly, it is expensive to calculate the ray intersections online, and secondly, the construction of the selection matrix itself is computationally expensive. Thus, we generate U points for all of the N cells in the grid. We then create a lookup table for these points using a kd-tree. The cell indices that rays from the origin to each point intersect are precomputed according to the LiDAR model in Fig. 2.1a and the radar model in Fig. 2.1b. Sensor measurements are then used to query the closest point and its associated precomputed ray, as illustrated in

Fig. 3.2a. This is an approximation; Fig. 3.2b shows that it is possible for quantization error where free cells can be identified incorrectly, but the hit cells will always be identified correctly. This method reduces the computational complexity required to obtain the rays from $\mathcal{O}(LM)$ to $\mathcal{O}(M \log(UN))$. The memory complexity is $\mathcal{O}(UNL)$ where L is the average number of cells intersected by the rays to the precomputed points. Practically, an occupancy grid with $N = 16384$ cells and $U = 25$ points per cell requires only 150 MB of memory to store all the necessary intersections. Further,



(a) Uniformly spaced points with precomputed rays.



(b) Comparison of precomputed ray intersections to true ray intersections.

Figure 3.2: (a) Precomputed rays sampled at uniformly spaced endpoints, with a spatial tree used to find the closest precomputed point for each measurement. (b) Comparison between the precomputed ray (black line) and the true ray (green line) from the ego centre (red dot): red cells highlight incorrectly identified free cells.

using measurements to access the precomputed rays, the selection matrix for LiDAR and radar can then be constructed using the compressed sparse row format, reducing the complexity from $\mathcal{O}(MN)$ to $\mathcal{O}(\|\boldsymbol{A}\|_0)$, where $\|\boldsymbol{A}\|_0$ refers to the number of nonzero elements in $\boldsymbol{A}$.

In practice, the precomputed free and occupied cell indices $\{\mathcal{O}_i\}_{i=0}^{UN-1}$ and $\{\mathcal{F}_i\}_{i=0}^{UN-1}$ are given as input, along with the UN precomputed points organized within a spatial kd-tree structure. To construct the selection matrix, the kd-tree is queried to identify the closest precomputed point to each measurement, then uses the corresponding precomputed indices to populate the nonzero entries in the selection matrix, following the procedure outlined in Algorithm 3. Using the precomputed rays to construct the selection matrix can extended to radar by pre-computing $\{\mathcal{O}_i\}_{i=0}^{UN-1}$ and $\{\mathcal{F}_i\}_{i=0}^{UN-1}$ using the model described in Fig. 2.1b.

3.2 Test-Data Quadtrees

Quadtrees in the 2D case or Octrees in the 3D case are common in OGM applications. The technique was initially used to reduce the memory usage of occupancy maps by varying the resolution of the stored maps [12, 47]. However, updating occupancy values on trees suffers from an access complexity when compared to typical arrays [16].

Algorithm 3 Fast selection matrix construction

Input: Reflection point coordinates *boldsymbolP*, spatial tree,
$\{\mathcal{O}_i\}_{i=0}^{UN-1}$, $\{\mathcal{F}_i\}_{i=0}^{UN-1}$ ▷ occupied/free index-sets for each grid point
Output: Selection matrix $\boldsymbol{A}$ and measurement vector $\boldsymbol{y}$

Query spatial tree to find closest points indexing precomputed occupied/free index-sets.
for each query point $m = 0, \ldots, M-1$ **do**
    Set $\boldsymbol{A}[2m,\, n] = 1 \quad \forall\, n \in \mathcal{O}_m$
    Set $\boldsymbol{A}[2m+1,\, n] = 1 \quad \forall\, n \in \mathcal{F}_m$
    $y[2m] \leftarrow y_{\text{occ}}|\mathcal{O}_m|$
    $y[2m+1] \leftarrow y_{\text{free}}$
end for

Therefore, in the case of computational speed test-data, Quadtrees were first applied in [16] to reduce the number of test points $\boldsymbol{x}^*$ (occupancies) to infer to increase time performance instead of reducing memory consumption.

Quadtrees are hierarchical structures where each node contains four leaf nodes. In occupancy grid mapping, each leaf in the quadtree is used to represent a spatial partition within some spatial region. By pruning a node, the resulting leaf's spatial region is four times larger, and it reduces the total spatial partitions that need to be represented. In Fig. 3.3, this hierarchical tree structure is illustrated, as well as the decision-making process in deciding whether a node can be pruned. Sensor measurements are typically sparsely grouped, as they appear where objects are sensed; therefore, in the space between the sensor and the sensed objects, the ray cast model samples many free space points in these regions where no measurements are recorded. For this reason, it is not necessary to represent the grid at a high resolution, as it is very likely that those cells will be estimated to be free and not occupied. Thus, using quadtrees, the resolution can be lowered, reducing the computational load.

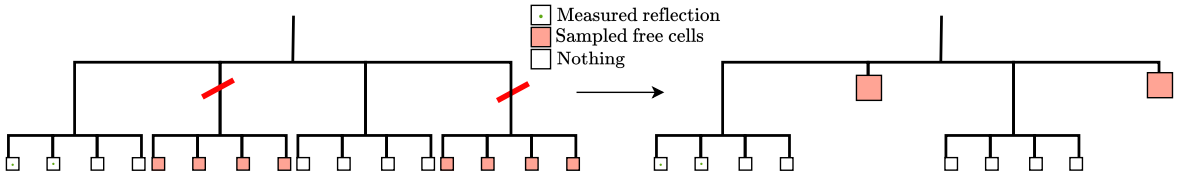


Figure 3.3: A depth two quadtree before and after pruning. Branches containing only sampled free cells are pruned as shown by the diagonal red line.

The quadtree is populated with the measurements and the sampled free space points along the rays cast to the measured points, as illustrated in Fig. 3.4. If across all the leaves within a node, there are only sampled free cells, then the node is pruned.

3.2.1 Test-Data Quadtrees with PCSBL and Ray Lookup Tables

First, we query the precomputed ray lookup table to retrieve occupied and free cells along each ray path. Second, we populate the quadtree with occupancy states as shown in the first grid in Fig. 3.4. Finally, we prune any node whose four child leaves (cells)

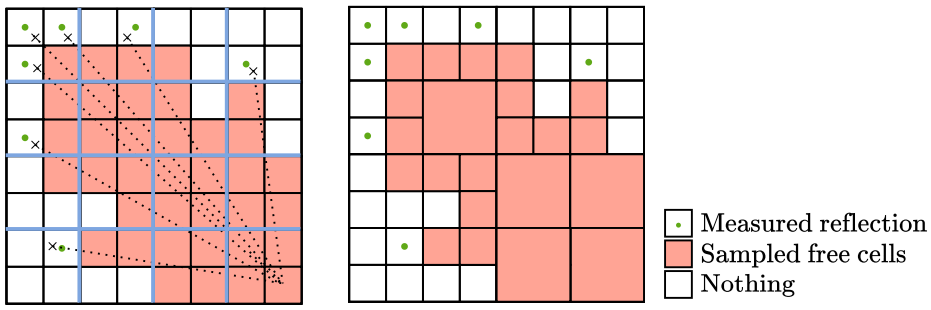
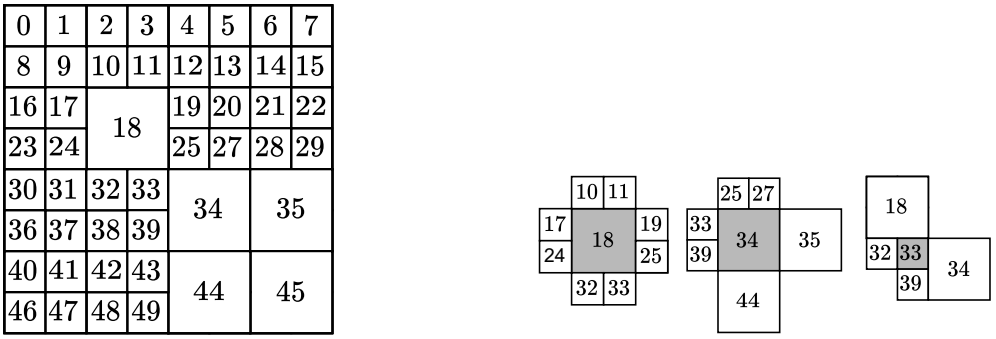


Figure 3.4: Adaptive grid resolution using quadtrees refines the grid based on spatial data. In the grid on the left, the blue grid shows quadtree nodes at one level higher. Measurements (crosses) populate the quadtree, marking cells with measurements as occupied (green dots) and preceding cells along rays (dotted lines) as free (red cells). Homogeneous nodes are pruned to larger cells.

contain only free states, as demonstrated in the second grid in Fig. 3.4. After pruning, cells vary in size, which reduces the effective number of unknowns N in the map vector, improving the algorithm complexity.

With varying resolution of the grid cells, standard row indexing is no longer applicable; however, row-major ordering is maintained for vectorizing the 2D map to a map vector. Additionally, because of the varying resolution of the grid cells, the number of neighbours of each cell also varies. Thus, the coupling pattern based on the direct neighbours $\mathcal{L}_n$ for the nth cell needs to be modified as illustrated in Fig. 3.5b. In the single- and multi-modal PCSBL algorithms, this has implications for (2.8), (2.11), and (2.20). Furthermore, it is observed that pruned cells are more likely to be free cells.



(a) Indexing of quadtree structure

(b) Neighbours with indices of 18th, 34th, and 33rd cells

Figure 3.5: Indexing of the occupancy grid after pruning the quadtree.

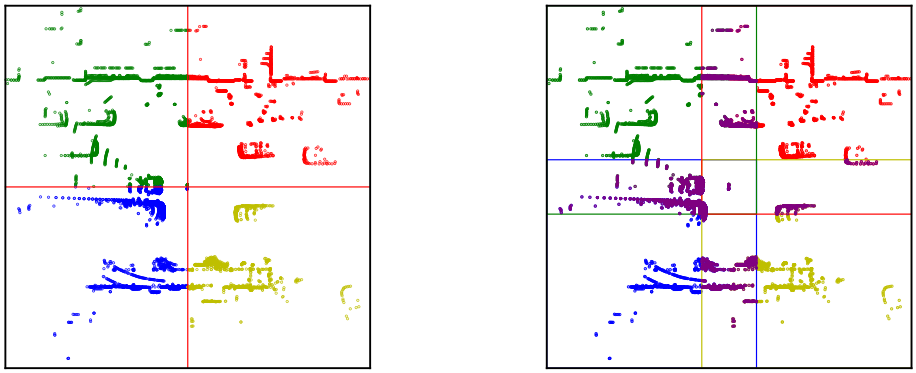
Larger cells at higher pruning levels have more neighbours, which can disproportionately influence free space information. Thus, only one level of pruning is recommended.

3.3 Exploiting the Measurement Model

We present methods to reduce mapping computation by exploiting LiDAR's angular structure. We propose two approaches—PO, parallel processing of sub-maps from measurements, and CP, optimising matrix operations by permuting $\mathbf{A}$'s columns by angular region.

3.3.1 Partition and Overlap

One method is partitioning the measurements into K discrete angular regions based on angular LiDAR measurement regions, as shown in Fig. 3.6a for $K = 4$. This reduces the number of columns in $\mathbf{A}$ by a factor of K as fewer grid cells are considered. Then the PCSBL algorithm can be applied to each region with lower complexity. This reduces the total complexity to approximately $\mathcal{O}(TN^3/K^2 + MN^2/K)$.



(a) No overlap (b) Overlap

Figure 3.6: Partitioning LiDAR measurements into $K = 4$ regions with (a) no overlap in the regions defined by four different colours and (b) overlap in the defined regions coloured in purple

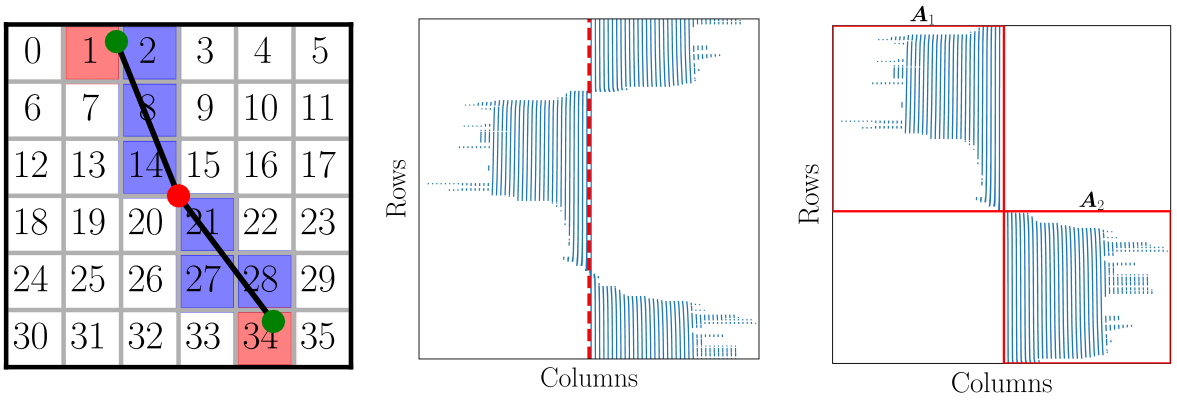
For K independent maps, the pattern coupling across the region boundaries is lost. This can be alleviated by partitioning and overlapping measurements as illustrated in Fig. 3.6b. After estimating the mean $\hat{\boldsymbol{\mu}}_k$ and variance $\boldsymbol{\sigma}_k$ ($\text{diag}(\hat{\boldsymbol{\Phi}}_k)$) for each of the k regions, overlapping regions are fused using Bayesian fusion for the nth cell

$$\hat{\mu}_{\text{merged}}[n] = \frac{\sum_{r \in \mathcal{R}_n} \frac{\hat{\mu}_r[n]}{\sigma_r^2[n]}}{\sum_{r \in \mathcal{R}_n} \frac{1}{\sigma_r^2[n]}}, \tag{3.1}$$

where $\mathcal{R}_n$ denotes the set of regions that overlap at the nth cell. The resulting merged mean map $\hat{\boldsymbol{\mu}}_{\text{merged}}$ is then thresholded to yield the final binary occupancy estimate $\hat{\boldsymbol{x}}$. Overlapping regions require the estimation of occupancy in cells belonging to overlapping regions to be processed repeatedly, increasing the cell count N and computational complexity compared to measurement partitioning without overlapping regions.

3.3.2 Region-Based Cell Permutation

In the benchmark PCSBL [24], the measurement model operates on row-indexed grids (Fig. 3.7a). As shown in Fig. 3.7a, LiDAR rays from distinct angular sectors (e.g., top vs. bottom) intersect mutually disjoint sets of cells. Consequently, the selection matrix is structurally split in the middle (dotted line in Fig. 3.7b); map cell indices are equivalent to selection matrix column indices. If the measurements are sorted angularly, a two block structured matrix can be realized as shown in Fig. 3.7c.



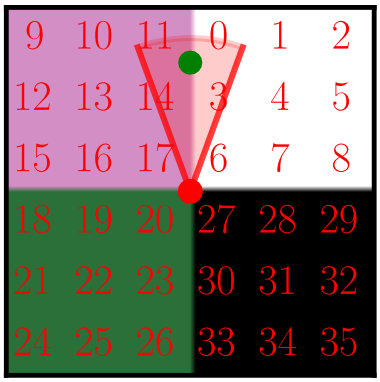
(a) 1-Region row indexing (b) 1-Region selection matrix (c) Angularly sorted measurement selection matrix

Figure 3.7: (a) LiDAR occupancy-mapping indexing scheme for the 1-region case, and (b–c) the corresponding selection matrices with nonzero entries in blue. In (a), free cells are marked in blue along the ray originating at the ego center (red dot) and the reflection (green point) cell is marked in red.

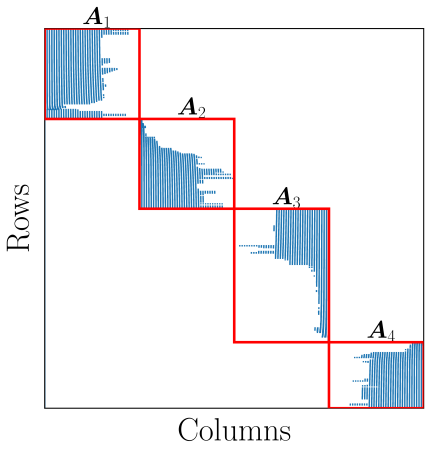
Exploiting this property further, the row-based indexing can be permuted to a Kregion-based indexing, where each region is defined by an angular sector around the ego center. For $K = 4$, the reindexing shown in Fig. 3.8a produces a four-block structure in $\boldsymbol{A}$ (Fig. 3.8b), with nonzero entries of the selection matrix perfectly within each block. Consequently, $\boldsymbol{A}^\top \boldsymbol{A}$ is block-diagonal. The computational complexity is reduced through block matrix operations in several areas: precomputation of $\boldsymbol{A}^\top \boldsymbol{y}$ and $\boldsymbol{A}^\top \boldsymbol{A}$, efficient residual computation for the noise variance update in (2.12), and most significantly, optimized matrix inversion for posterior covariance and multiplication operations in the mean expectation steps in (2.10). This achieves the same complexity reduction described in subsection 3.3.1.

It can be shown that permuting the columns of the selection matrix does not affect the final result of (2.10) because the inversion of the columns of a matrix transposed and multiplied with itself is permutation invariant. Applying the permutation matrix $\boldsymbol{P}$ to the selection matrix Since inversion is permutation-invariant, permuting the columns of the selection matrix does not affect the final result of (2.10). In particular, if we apply the permutation $\boldsymbol{P}$ to the selection matrix,

$$\boldsymbol{A}_P \;=\; \boldsymbol{A}\,\boldsymbol{P}, \tag{3.2}$$



(a) 4-Region row-major indexing



(b) 4-Region selection matrix

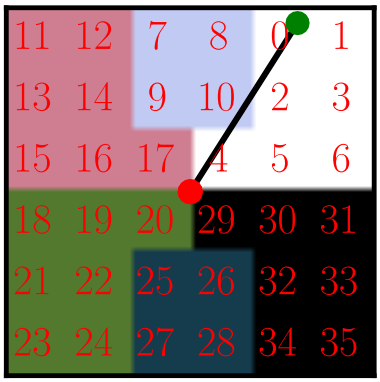
Figure 3.8: (a) LiDAR occupancy-mapping indexing scheme for the 4-region case, and (b) the corresponding selection matrix. In (a), region row-major indexing is shown for regions defined by distinct colours. In (b) the

then inverting the Gram matrix of $\boldsymbol{A}_P$ yields

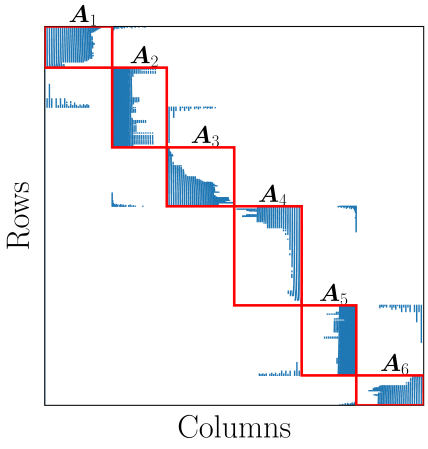
$$\left(\boldsymbol{A}_P^\top \boldsymbol{A}_P\right)^{-1} = \left(\boldsymbol{P}^\top \boldsymbol{A}^\top \boldsymbol{A}\, \boldsymbol{P}\right)^{-1} = \boldsymbol{P}^{-1} \left(\boldsymbol{A}^\top \boldsymbol{A}\right)^{-1} \left(\boldsymbol{P}^\top\right)^{-1}. \tag{3.3}$$

By left- and right-multiplying by $\boldsymbol{P}$ and $\boldsymbol{P}^\top$ the original inverse is recovered:

$$\boldsymbol{P} \left(\boldsymbol{A}_P^\top \boldsymbol{A}_P\right)^{-1} \boldsymbol{P}^\top = \boldsymbol{P}\, \boldsymbol{P}^{-1} \left(\boldsymbol{A}^\top \boldsymbol{A}\right)^{-1} \left(\boldsymbol{P}^\top\right)^{-1} \boldsymbol{P}^\top = \left(\boldsymbol{A}^\top \boldsymbol{A}\right)^{-1} \tag{3.4}$$



(a) 6-Regions row-major indexed



(b) 6-Regions selection matrix

Figure 3.9: (a) LiDAR occupancy-mapping indexing scheme for the 6-region case, and (b) the corresponding selection matrix. In (a), six distinct index regions are shown, and rays may traverse multiple regions before reaching a reflection point.

Furthermore, for $K > 4$ in the LiDAR measurement model, the region boundaries can pass through grids, and the LiDAR ray may intersect cells belonging to multiple

regions. For example, the LiDAR ray in Fig. 3.9a results in a $\boldsymbol{A}$ matrix like in Fig. 3.9b where certain rows have nonzero entries spanning multiple regions. In the case of the radar measurement model, due to the beam width, no matter the number of regions, the corresponding beam/ray can intersect multiple regions, as shown in Fig. 3.8a. This breaks the block structure and hinders the use of block matrix operations. This can be dealt with by discarding the elements outside the blocks to retain the block structure. Alternatively, we split rows spanning multiple regions to enforce the block structure so that no information is discarded. Fig. 3.10 shows an example where a row with support falling in two regions is split into two rows.

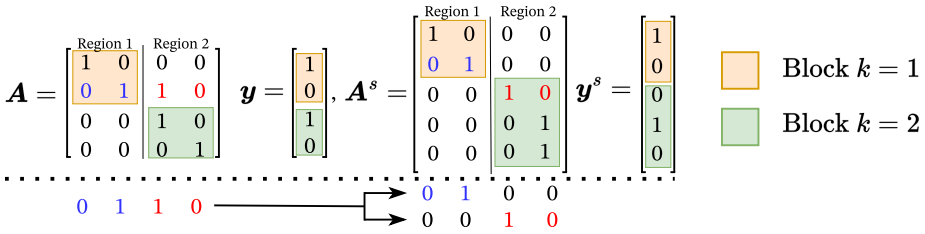


Figure 3.10: Splitting LiDAR measurements that span multiple regions: the original matrix $\mathbf{A}$ (left) has a row with overlapping elements (in red). This row is split at the boundary and shifted into the appropriate blocks to form the block-structured matrix $\mathbf{A}^s$ (right), introducing additional measurements (see highlighted rows below the dotted line).

Comparing PO and CP, CP improves the computational performance of PCSBL while preserving full coupling across the K regions, eliminating an extra fusion step.

A brief overview is now provided on how PCSBL exploits the resulting blocks through modifications to the posterior covariance and posterior mean when using an exploited CP selection matrix. Following the original formulation in (2.12), the residual for the kth block becomes:

$$\|\boldsymbol{y}_k - \boldsymbol{A}_k \hat{\boldsymbol{\mu}}_{\boldsymbol{k}}{}^{(t)}\|_2^2, \tag{3.5}$$

where $\boldsymbol{y}_k$ and $\hat{\boldsymbol{\mu}}_k$ is the subset of the measurements corresonding to the kth block. The posterior mean and covariance from the original formulation (2.10) are correspondingly modified to

$$\hat{\boldsymbol{\Phi}}_k^{(t)} = ((\sigma^{-2})^{(t-1)} \boldsymbol{A}_k^\top \boldsymbol{A}_k + \boldsymbol{D}_k^{(t)})^{-1} \tag{3.6}$$

and

$$\hat{\boldsymbol{\mu}}_k^{(t)} = (\sigma^{-2})^{(t-1)} \hat{\boldsymbol{\Phi}}_k^{(t)} \boldsymbol{A}_k^\top \boldsymbol{y}_k. \tag{3.7}$$

The exact implementation of these equations is shown in Algorithm 4.

3.3.2.1 Extension of Selection Matrix Exploitation for Fusion

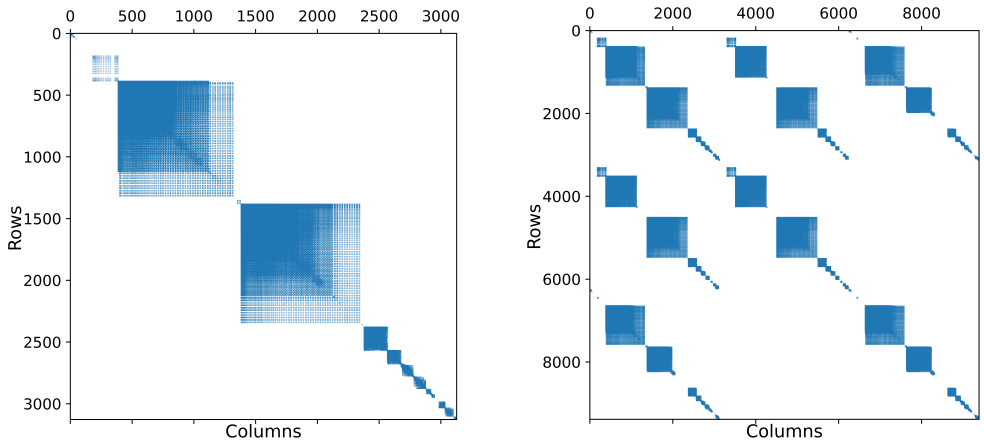
The PO method extends easily to multi-modal fusion methods, while CP requires more consideration. For fusion-based PCSBL variants (CIS and CS), the CP method can be directly applied. CS maintains the same selection matrix structure in (2.14), where columns correspond to cells of a single occupancy map, enabling the same region-based cell permutation. The permuted CS selection matrix creates a block diagonal $\boldsymbol{A}_{\text{CS}}^\top \boldsymbol{A}_{\text{CS}}$,

Algorithm 4 Occupancy grid map estimation using PCSBL-CP

Input: Permuted selection matrix $\boldsymbol{A}$, measurement vector $\boldsymbol{y}$, number of regions K
Parameters: coupling β, Gamma parameters $a, b, c, d > 0$
Output: Binary occupancy map $\hat{\boldsymbol{x}}$

$t \leftarrow 0, \quad \boldsymbol{\alpha}^{(t)} \leftarrow \mathbf{1} \quad (\sigma^2)^{(0)} \leftarrow 0.5$
Pre-compute K blocks $(\boldsymbol{A}_k^\top \boldsymbol{A}_k)$ and $\boldsymbol{A}_k^\top \boldsymbol{y}_k$
repeat
    Compute $\boldsymbol{D}^{(t)}$ via (2.8)
    Update $\boldsymbol{\alpha}^{(t)}, (\sigma^2)^{(t)}$ via (2.11), (2.12), compute the residual block-wise via (3.5)
    Update $\hat{\boldsymbol{\Phi}}^{(t)}$ block-wise via (3.6)
    Update $\hat{\boldsymbol{\mu}}^{(t)}$ block-wise via (3.7)
    $t \leftarrow t + 1$
until convergence
Compute $\hat{\boldsymbol{x}}$ via (2.13)

and the residual and trace calculations in (2.19) exploit identical structures as single-modality PCSBL-CP, this is illustrated in Fig. 3.11a. Structurally, CP application to CS fusion mirrors the single-modality case, utilizing the same block structures in (2.18) for block inversions, matrix operations, and residual computations in (2.19).



(a) $\boldsymbol{A}_{\mathrm{CS}}^\top \boldsymbol{A}_{\mathrm{CS}}$ (b) $\boldsymbol{A}_{\mathrm{CIS}}^\top \boldsymbol{A}_{\mathrm{CIS}}$

Figure 3.11: Structure of (a) $\boldsymbol{A}_{\mathrm{CS}}^\top \boldsymbol{A}_{\mathrm{CS}}$ and (b) $\boldsymbol{A}_{\mathrm{CIS}}^\top \boldsymbol{A}_{\mathrm{CIS}}$ after region-based cell permutation ($K = 16$) of selection matrices $\boldsymbol{A}_{\mathrm{L}}$ and $\boldsymbol{A}_{\mathrm{R}}$.

This is not the case for CIS due to the altered measurement model described in (2.16) and illustrated in Fig. 3.11. With the addition of the residual variables, the columns no longer correspond to the cells of a single occupancy map. To restore a block–diagonal structure in $\mathbf{A}_{\mathrm{CIS}}^\top \mathbf{A}_{\mathrm{CIS}}$, we permute

$$\left[x_{\mathrm{c}}[1] \dots x_{\mathrm{c}}[N] \; x_{\Delta\mathrm{L}}[1] \dots x_{\Delta\mathrm{L}}[N] \; x_{\Delta\mathrm{R}}[1] \dots x_{\Delta\mathrm{R}}[N]\right], \tag{3.8}$$

so that the triplets $\big(x_{\mathrm{c}}[i],\, x_{\Delta\mathrm{L}}[i],\, x_{\Delta\mathrm{R}}[i]\big)$ appear consecutively for $i = 1, \dots, N$. The

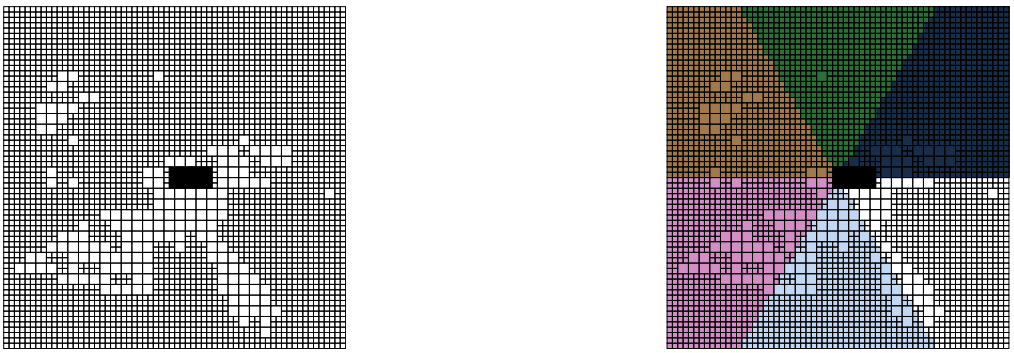
permuted vector is given by

$$\begin{bmatrix} x_{\mathrm{c}}[1] & x_{\Delta\mathrm{L}}[1] & x_{\Delta\mathrm{R}}[1] & \dots & x_{\mathrm{c}}[N] & x_{\Delta\mathrm{L}}[N] & x_{\Delta\mathrm{R}}[N] \end{bmatrix}, \tag{3.9}$$

which enforces block–diagonality in $\mathbf{A}_{\mathrm{CIS}}^{\top}\mathbf{A}_{\mathrm{CIS}}$ by eliminating cross–terms between different columns belonging to separate regions.

3.3.2.2 Quadtrees and region-based cell permutation

The adaptive resolution quadtree method is applied on top of region-based cell permutation. The method could be applied directly without any modifications; however, if a node were pruned directly on a region border, when calculating the free cells and inserting them into the selection matrix, even more rays would intersect multiple regions due to the less smooth regions caused by pruning. This would ultimately lead to more measurements in our selection matrix as a result of the splitting of the measurements in Fig. 3.10. The solution to this is to add another step in the pruning logic. If a node



(a) One region (b) Six regions

Figure 3.12: For a single region, the quadtree is pruned as illustrated in (a). However, when dealing with multiple regions, nodes are only pruned if they are entirely contained within a single region.

falls on the boundary of a region, then this node is prevented from pruning. Thus, when comparing Fig. 3.12a and Fig. 3.12b, it is visible that the lower half of the map has a higher resolution along the boundaries because of the prevention of node pruning.

3.4 Summary

This chapter presents computational acceleration techniques for PCSBL in occupancy grid mapping, addressing the algorithm's computational burden. A ray lookup table construction method using spatial k-d trees reduces computational complexity, followed by the use of the CSR sparse matrix construction technique. The integration of quadtree adaptive resolution maintains efficiency with variable-resolution grids, requiring modified indexing and coupling structures to work with PCSBL. Two measurement exploitation strategies, PO and CP, leverage the measurements' angular interactions with the map to reduce the precomputation complexity of the matrices required for

EM in PCSBL, as well as the EM steps themselves. When combining quadtree methods with region-based approaches, pruning constraints prevent nodes spanning region boundaries from being pruned to avoid measurement splitting, resulting in higher resolution along boundaries. The complexity contributions are outlined in Table 3.1.

Table 3.1: Order of Complexity Comparison

Algorithm	Initial Complexity	Accelerated Complexity
PCSBL	$\mathcal{O}(TN^3 + MN^2)$	$\mathcal{O}(T\frac{N^3}{K^2} + \frac{MN^2}{K})$
Selection Matrix	$\mathcal{O}(MN + ML)$	$\mathcal{O}(\|\|(\boldsymbol{A}\|\|_0) + M\log(UN))$
PCSBL Fusion	$\mathcal{O}(TN^3 + ((M_\mathrm{L} + M_\mathrm{R})N^2)$	$\mathcal{O}(T\frac{N^3}{K^2} + \frac{(M_\mathrm{L}+M_\mathrm{R})N^2}{K})$

4 Evaluations and Results

Using the real-world automotive datasets nuScenes [27] and RADIATE [43], this work uses LiDAR and radar modalities to compare the accelerations discussed in Chapter 3 with the benchmark algorithms, PCSBL [6, 24], fusion PCSBL [6], BGK [21], and ISM [8].

This chapter first explains the evaluation details for occupancy grid maps. The datasets are presented along with their provided modalities and ground truths, followed by a discussion of how the metrics combine with ground truths to quantify algorithm performance on occupancy maps. Second, the chapter presents experiments that justify the chosen parameters for the accelerated methods (PO, CP, and ray lookup selection matrix construction). Finally, the chapter presents comprehensive qualitative and quantitative results, including occupancy grid map quality and time complexity results on both datasets using the justified acceleration parameters.

4.1 Evaluation

This section describes the evaluation framework used to assess the proposed methods. First, the datasets are presented, including the ground truth data and sensor modalities employed. This is followed by an explanation of the quantitative metrics that utilize the ground truth data for performance assessment. Finally, the experimental settings and configurations used to evaluate the methods are detailed.

4.1.1 Datasets

4.1.1.1 nuScenes

The nuScenes dataset scenes [27] are recorded in either Boston or Singapore. Both of these locations are home to dense traffic. The goal of the dataset is to introduce rich, multi-modal sensor data to support the development of robust perception algorithms, particularly those capable of performing reliably under adverse weather and challenging real-world driving conditions. Additionally, to support the development of machine learning methods, it consists of a rich set of 3D ground truth bounding boxes for the classes specified in Table 4.1.

It consists of 6 cameras, 5 MIMO radars, and 1 LiDAR, which together provide a full 360-degree field of view around the ego vehicle, as illustrated in Fig. 4.1. The platform is also equipped with GPS and an IMU for precise localization and motion estimation. With the map expansion for nuScenes, the map can be classified into its semantic layers, making it possible to specify the smaller focus regions around the ego vehicle where occupancies must be estimated, an optimisation discussed in subsubsection 2.2.2.2.

Table 4.1: Object Classes in the nuScenes Dataset

Class Name	**Sub Class Names**
Animal	Animal
Movable Object	Barrier, Debris, Push/Pull-able, Traffic cone
Static Object	Bicycle rack
Vehicle	Bicycle, Bendy bus, Rigid bus, Car, Construction vehicle, Ambulance, Police vehicle, Motorcycle, Trailer, Truck
Human	Adult, Construction worker, Personal mobility, Police officer, Stroller, Wheelchair

We use the roadway and walkway layers to include humans as well as vehicles in the occupancy map.

The radar and lidar are provided as a point cloud as illustrated in Fig. 4.2. Un-

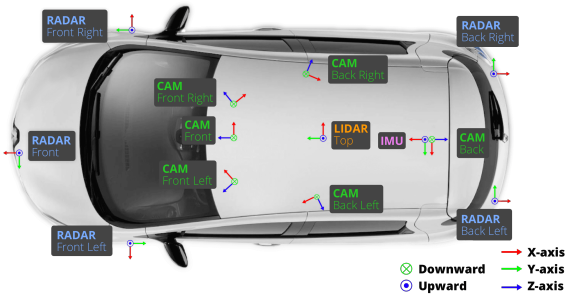
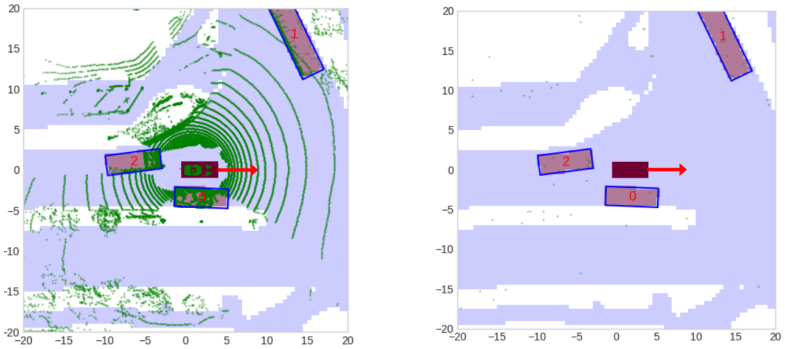


Figure 4.1: nuScenes ego vehicle showing all the sensors and their locations on the ego vehicle

fortunately, the preprocessed radar point cloud in the nuScenes dataset is excessively sparse. This sparsity often results in surrounding objects containing no radar points at all, as reported in previous studies [24, 48]. Therefore, we omit radar-based evaluation on nuScenes due to this known limitation.



(a) LiDAR point cloud (b) Radar point cloud

Figure 4.2: Illustration of the nuScenes birds eye view LiDAR (a) and radar (b) modalities with ground truth boxes (pink boxes outlined in blue) within the focus area (pale blue) with the ego vehicle (dark red) facing the right depicted by the red arrow.

4.1.1.2 Radar Dataset In Adverse Weather

The RAdar Dataset In Adverse weaThEr (RADIATE) recorded in Edinburgh, Scotland [43] is a dataset providing LiDAR, radar, and GPS in varying weather conditions (rain, snow, fog, sun, and night). The goal of the dataset is to use radar in situations where vision or LiDAR systems may fail. The RADIATE dataset provides annotated scenes with Pedestrian and Vehicle classes Table 4.2. The annotations are provided as 2D bounding boxes. Additionally, the dataset is not as accurately annotated as nuScenes, with many scenes omitting bounding boxes for vehicles and pedestrians; therefore, the scenes used to evaluate the accelerated methods are selected carefully. Thus, there are fewer accurately annotated scenes than nuScenes.

Table 4.2: Object Classes in the RADIATE Dataset

Class Name	Sub Class Names
Vehicle	Bicycle, Car, Van, Truck, Bus, Motorbike
Pedestrian	Single pedestrian, Group of pedestrians

The dataset consists of a single 360° scanning radar, a 360° LiDAR (same model as the nuScenes LiDAR), and a single front-facing stereo camera. Automotive radar

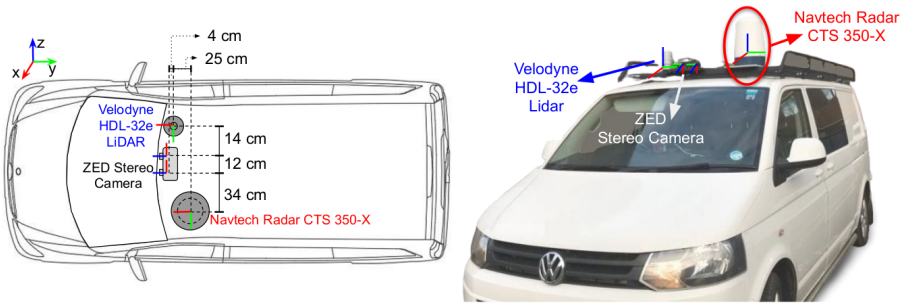
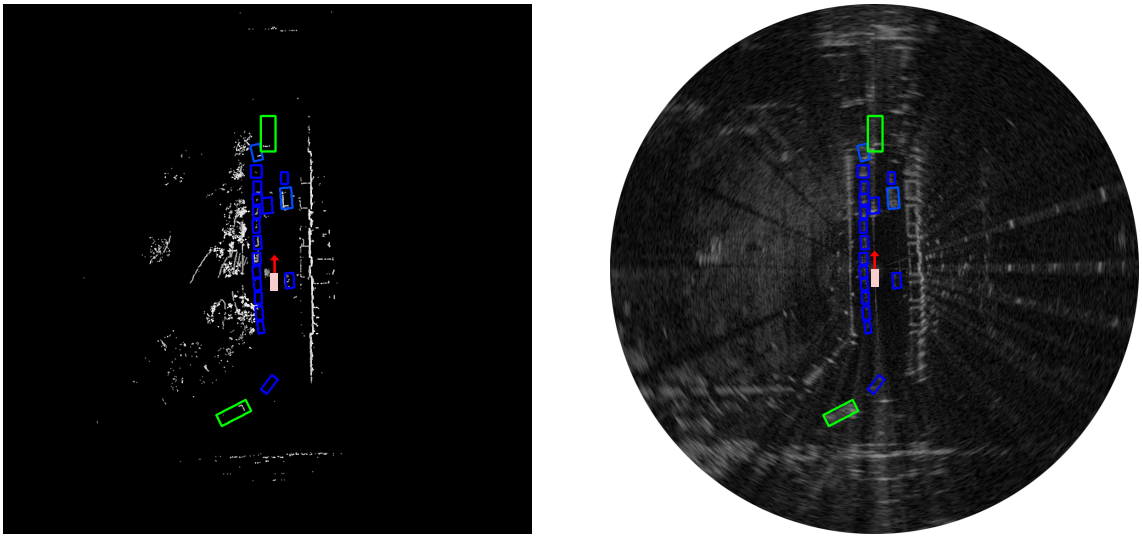


Figure 4.3: RADIATE ego vehicle showing all the sensors and their locations on the ego vehicle

systems commonly use MIMO technology with multiple transmitters and receivers to detect the direction of arrival. While cost-effective, these systems suffer from poor azimuth resolution, resulting in blurry cross-range images that may lack sufficient detail for object recognition or precise scene mapping. Scanning radar offers improved azimuth resolution by mechanically rotating the antenna to measure each direction sequentially, but this approach suffers from slower update rates of 4Hz. As a result, slight synchronization offsets between the LiDAR and radar modalities can occur, leading to some misalignments in the sensor data. The bounding box annotations are more accurately aligned with the radar modality.

The 3D LiDAR information is provided in point cloud form as shown in Fig. 4.4a, whereas the radar range-azimuth image Fig. 4.4b is not. Due to how the LiDAR is placed on the ego vehicle roof in Fig. 4.3, the scan radar obstructs the LiDAR line of sight towards the back left, leaving a dead zone. Due to the radar being a scan radar,

only 2D radar information is obtained. To extract a 2D point cloud, CFAR is used as explained in section 2.1.



(a) LiDAR point cloud

(b) Radar Cartesian range-azimuth map

Figure 4.4: Illustration of the RADIATE birds-eye view LiDAR and radar modalities with ground truth boxes. The ego vehicle is facing upwards as depicted by the red arrow.

4.1.2 Evaluation Metrics

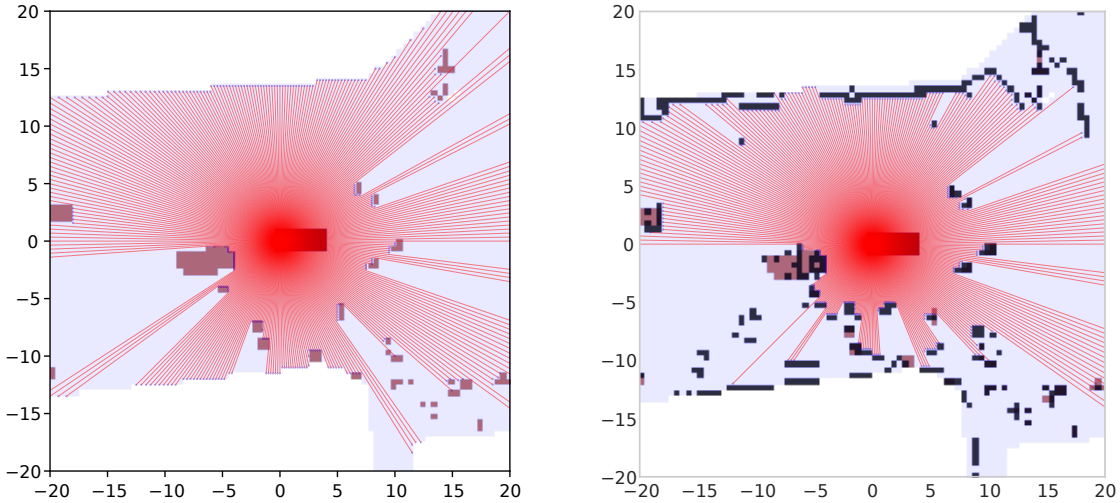
To evaluate the accuracy of the maps after applying the acceleration techniques discussed in Chapter 3 three metrics are used. Intersection over bounding box (IoBB), angular scan normalised mean squared error (AS-NMSE), and the free space error.

4.1.2.1 Angular Normalised Mean Squared Error

The AS-NMSE from [24] is a metric that evaluates how accurately driveable areas and surrounding obstacle boundaries are identified. This is accomplished by casting virtual rays outward in all directions, with one ray every degree for a complete 360-degree view around the vehicle. For each ray, the distance is measured from the vehicle to where the ray first hits an occupied cell in the estimated occupancy map. These distances are then compared to the corresponding distances the same rays would travel in the ground truth map, yielding

$$\text{AS-NMSE} = \frac{||\hat{\boldsymbol{d}} - \boldsymbol{d}||^2}{||\boldsymbol{d}||^2}, \tag{4.1}$$

where the collection of all 360 distance measurements forms the vectors $\hat{\boldsymbol{d}}$ (estimated distances) and $\boldsymbol{d}$ (ground truth distances). The angular scans with virtual rays with both ground truth distances and the estimated distances are illustrated in Fig. 4.5a and Fig. 4.5b, respectively.



(a) Angular scan on the ground truth

(b) Angular scan on the estimated occupancy map

Figure 4.5: Figure illustrating the (a) ground truth (pink cells)angular scan and the (b) angular scan on the estimated occupancy mask (black cells) around the ego vehicle (red cells) within the focus area (pale blue)

4.1.2.2 Intersection Over Bounding Box

IoBB is a metric used to quantify the level of spatial coverage of objects that the estimated occupancy map has for each and every ground truth object. In a scene, the IoBB can be represented as a vector $\boldsymbol{v} \in \mathbb{R}^{V_{Gt-boxes}}$, with a value for each ground truth box. The ith object is defined as

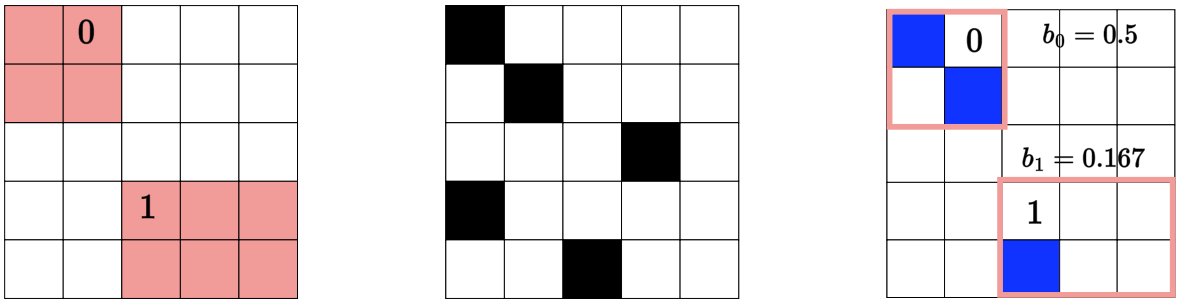
$$h_i = \frac{V_{Intersection-i}}{V_{Gt-i}} \tag{4.2}$$

where $V_{Intersection-i}$ is the number of occupied cells in the estimated occupancy map Fig. 4.6b that intersect with the ith object ground truth box shown in Fig. 4.6a and V_{GT-i} is the number of cells in the ith ground truth box. Therefore, the ith object has a IoBB value $h_i \in [0, 1]$. An object is considered detected if the IoBB value is greater than zero.

4.1.2.3 Detection Rate

Thus, using the IoBB for each object in a given scene, the detection rate is defined as the number of objects with non-zero IoBB divided by the total number of ground truth boxes in the scene

$$\text{Detection Rate} = \frac{\|\boldsymbol{h}\|_0}{V_{Gt-boxes}}. \tag{4.3}$$



(a) Ground truth with single object

(b) Estimated occupancy map

(c) Intersection

Figure 4.6: IoBB metric illustrated for a map with two objects. IoBB quantifies the intersection between the ground truth box for each object with IDs 0 and 1 and the estimated occupancy map as a fraction of the number of intersecting cells divided by the total number of cells in the bounding box.

4.1.2.4 Free Space Error

The free space error is the number of false positives—points incorrectly estimated as occupied outside the ground truth boxes—divided by the total number of cells outside the ground truth boxes:

$$\text{Free Space Error} = \frac{\#\text{False Positives}}{\#\text{Cells outside ground truth boxes}}. \tag{4.4}$$

For example in Fig. 4.6, the free space error is $\frac{2}{25} = 0.08$. This metric evaluates the accuracy of free space representation in regions beyond the angular scan coverage. Such accuracy is crucial for path planning algorithms that rely on out-of-line-of-sight occupancy estimates when computing future trajectories. It also allows us to identify when an occupancy algorithm introduces additional error into its estimations.

4.1.3 Experimental Setting

This section discusses the parameters relevant to the evaluation of accelerated PCSBL variants. Different parameters are used for each dataset, while the overlapping parameters between the accelerated PCSBL variants and the benchmark PCSBL variants, for both single- and multi-modal PCSBL, are outlined.

For the nuScenes dataset, the map parameters are defined in Table 4.3a. Since digital map information is provided, a square map centered around the ego vehicle is selected, while the narrower digital road geometry determines the actual area of interest. Consequently, the map can contain up to 6400 cells, although the actual number is typically much lower.

In contrast, the RADIATE dataset does not provide digital map information. Therefore, a fixed rectangular region around the ego vehicle is defined to emulate a straight road. Furthermore, because the stereo camera visuals are front-facing and crucial for evaluating scene results, the region is positioned to include only minimal space behind the ego vehicle.

Table 4.3: Map parameters for the nuScenes and RADIATE datasets.

(a) nuScenes parameters

Parameter	Value
N (scene road geometry dependent)	6400
N_x, N_y	80, 80
grid size	0.5m

(b) RADIATE parameters

Parameter	Value
N	3200
N_x, N_y	40, 80
grid size	0.5m

The parameters for benchmark single and multi-modal PCSBL as defined in [6, 24] are highlighted in Table 4.4. The alternate methods, BGK [21] and ISM [8], have their parameters along with their implementation details outlined in Appendix A.

Table 4.4: List of parameters used in single and multi-modal PCSBL and the measurement model.

Parameter	Description	Value
$y_{\text{free}}, y_{\text{occ}}$	Labels for the Proposed Measurement Models	0, 1
β	PCSBL coupling parameter	1
a, b, c, d	Gamma Prior Parameters	0.5, 10^{-6}, 10^{-6}, 10^{-6}
Ω	Beam Width for and Radar Measurement Model	2°
Δ	Obstacle Thickness for and Radar Measurement Model	2 grid cells
τ	Threshold	0.3
a_c, a_{L}, a_{R},	CIS gamma Parameters	0.5, 1.3, 1.3

The parameters regarding the accelerations are not yet defined. The PCSBL variants employ approximate ray lookup for selection matrix construction using the selected U, whereas benchmark methods from [6, 8, 21, 24] use exact ray casting. All methods are implemented in Python running on a Ryzen AI 9 365 CPU.

4.2 Acceleration Method Parameter Evaluation

This section presents experiments that inform the selection of parameters for the acceleration methods listed in Table 4.5. To evaluate these parameters, 40 scenes from the nuScenes dataset are used, chosen specifically for their high presence of pedestrians and vehicles captured by the LiDAR sensor.

Table 4.5: PCSBL Acceleration Parameters

Parameter	Description
U	Ray lookup number of points per cell
K	Measurement model exploited the number of regions
Overlap	Number of overlapping cells between each of the k regions
Measurement Splitting	split measurements with rays overlapping in multiple regions
Quadtree Pruning	The adaptive resolution is capped at one level of pruning

4.2.1 Ray Look-Up Table

The ray lookup table is used to optimize selection matrix construction speed and requires the number of points per cell U to be selected. However, excessively large values of U lead to prohibitive memory requirements due to the increased storage needed for precomputed rays. Therefore, U is chosen to use memory efficiently while preserving performance relative to online computation of the rays.

Box plots for different U values are illustrated in Fig. 4.7 and Fig. 4.8. From the

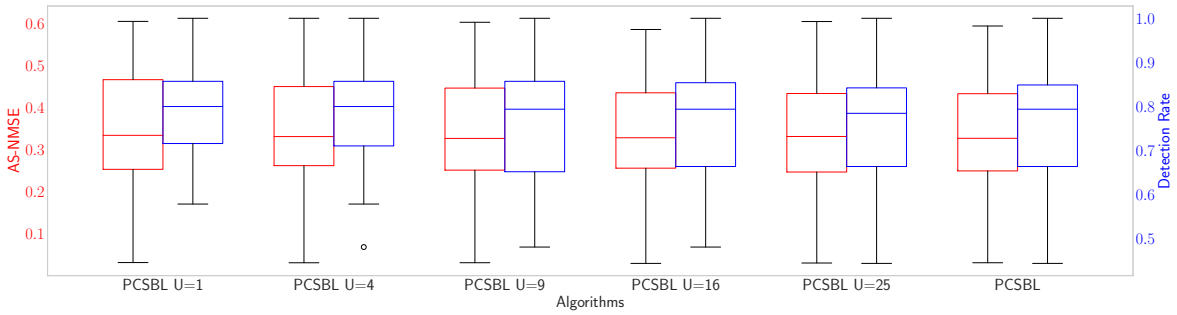


Figure 4.7: Distribution of AS-NMSE and the proportion of detected objects in ground-truth maps over 40 densely populated nuScenes samples, shown for varying numbers of uniformly sampled precomputed points U per cell.

AS-NMSE and proportion of detection in Fig. 4.7 alone, it may be difficult to select U. With four points per cell, the performance appears to have approximately the same AS-NMSE as the benchmark PCSBL while outperforming it in the proportion of detections.

For this reason, the freespace error metric in Fig. 4.8 is more informative. It can be observed that 4 points per cell introduces the most error, whereas 16 or 25 cells per grid perform almost the same as the benchmark PCSBL across all the metrics. Thus,

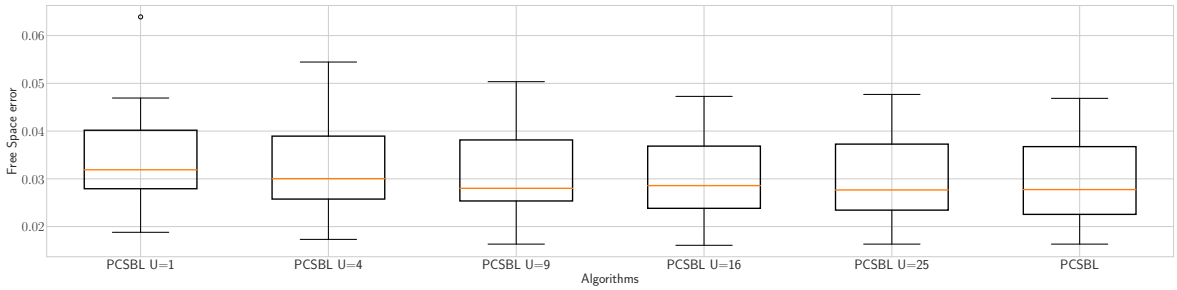
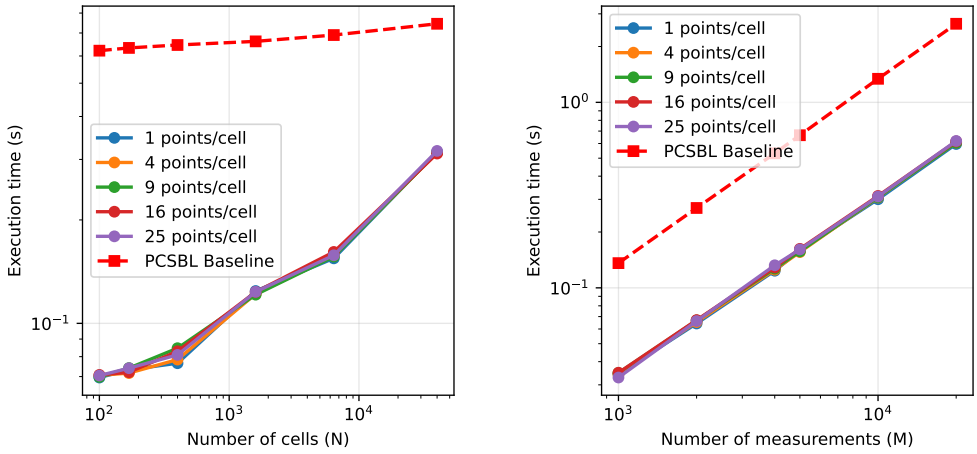


Figure 4.8: Distribution of AS-NMSE and the proportion of detected objects in ground-truth maps over 40 densely populated nuScenes samples, shown for varying numbers of uniformly sampled precomputed points U per cell.

either 16 or 25 points is a desirable choice.

We observe that the runtime remains largely unaffected by increasing the number of points per cell. The computational boost comes from both the ray lookup and the common sparse row technique used to construct the selection matrix. This behaviour



(a) Number of cells vs execution time (b) Measurements vs execution time

Figure 4.9: Time complexity for construction of selection matrix. (a)Time complexity as the number of cells changes for $M = 5000$ and (b) as the number of measurements changes for $N = 6400$

is illustrated in Fig. 4.9a, which shows experimental results for the time complexity of measurement matrix construction. To achieve an accurate and much faster approximation of online ray computation, the ray lookup method uses $U = 25$ points per cell.

4.2.2 Exploiting the Measurement Model

The PO and CP methods both make use of regions to reduce the complexity of PCSBL. Furthermore, PO has an overlap between cells to be considered, and CP has splitting or no splitting of measurements. The CP method is evaluated for $K = 4, 8, 16, 25$ and 30 regions for the grid resolution specified in Table 4.3a. The regions are visualised in Fig. 4.10, where, notably, after 16 regions, for the selected grid parameters, the borders become significantly more irregular than for $K = 4, 8$ and 16.

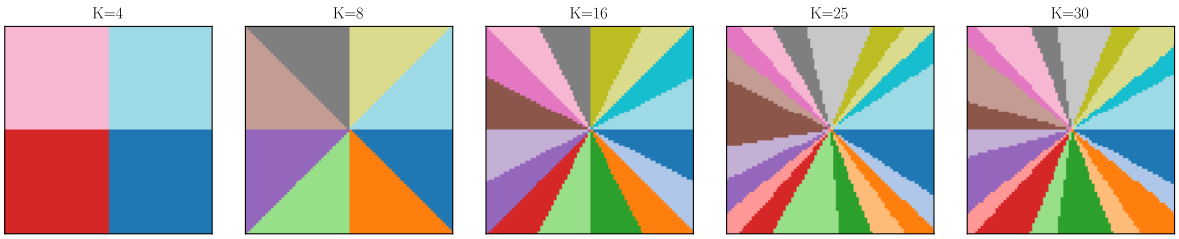


Figure 4.10: Map partitioned regions for increasing numbers of regions.

To aid in parameter selection, mean maps (before thresholding to $\hat{\boldsymbol{x}}$ with equation (2.13)) for 40 separate samples are generated for each method with different parameters. The mean maps are then evaluated at various threshold values τ, producing the occupancy estimates. At each threshold, the metrics discussed in subsection 4.1.2 are

computed and averaged across the 40 samples. The methods are only evaluated on LiDAR, and the time results shown for the methods in this section do not include the time required for constructing the selection matrix.

4.2.2.1 Region-Based Cell Permutation

The CP method is evaluated across multiple regions with and without the splitting approach in Fig. 4.11b . For 4 regions, both variants achieve identical performance to benchmark PCSBL since no LiDAR rays intersect multiple regions. This provides computational acceleration while maintaining mathematical equivalence to the benchmark method.

As the number of regions increases, the computational complexity of the CP method decreases. However, both CP variants exhibit higher free space error compared to benchmark PCSBL, while simultaneously achieving improved detection rates. This trade-off indicates that selecting the optimal number of regions requires careful consideration rather than simply maximizing the region count.

The splitting method preserves free space information that would otherwise be lost in the non-splitting approach, leading to consistently superior AS-NMSE performance compared to the non-splitting variant. This can be observed by the fact that the splitting method typically has lower AS-NMSE than the non-splitting variant. By splitting measurements into separate artificial measurements, the splitting approach artificially inflates the measurement count (M_{L} or M_{R}) in the denominator of the noise variance update equation (2.12), while the residual term in the numerator remains unchanged. This manipulation results in underestimated noise variance, which in turn produces slightly elevated occupancy estimates. This consistent bias toward higher occupancy values is observed in Fig. 4.11, both when comparing the splitting method against its non-splitting counterpart and against the benchmark PCSBL method when using the same τ.

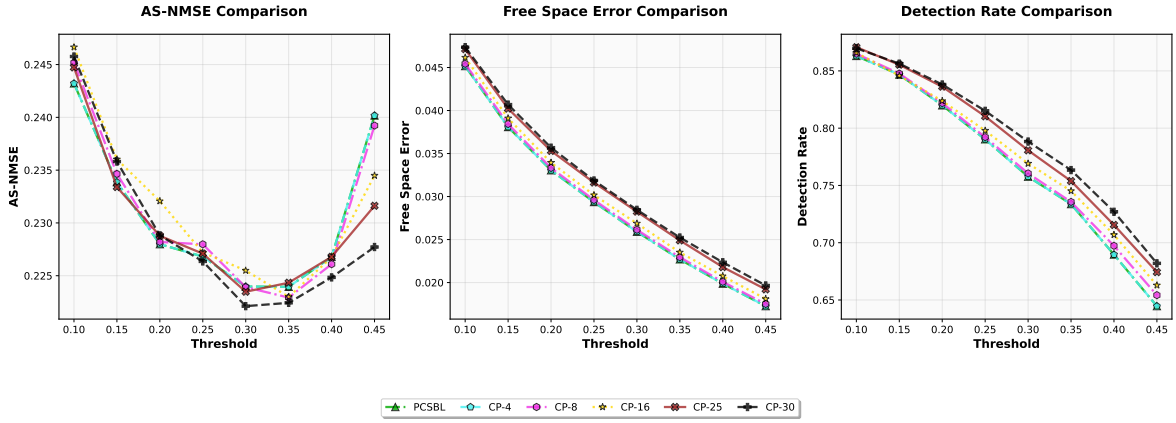
The number of regions is decided as $K = 16$, as it is observed in Fig. 4.11 that for more regions than 16, the AS-NMSE can increase relative to benchmark PCSBL and the free space error increases disproportionately.

To preserve the AS-NMSE performance when using this acceleration method, the splitting method is used with 16 regions; however, to achieve comparable free space error, the threshold is adjusted from the original τ =0.3 to τ_{CP} =0.35. For brevity, CP is used to refer to PCSBL with $K = 16$ regions with the splitting method.

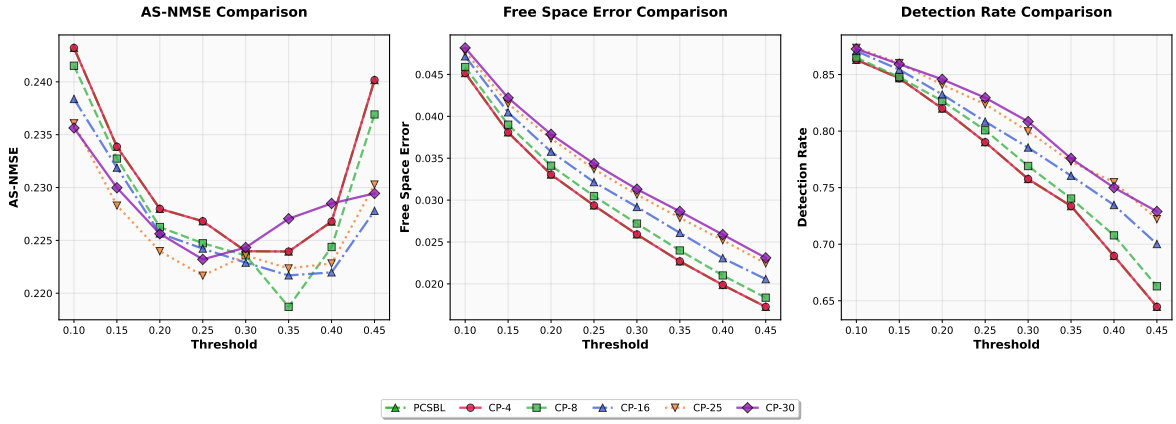
As shown in Fig. 4.12, computational benefits extend beyond 16 regions, though runtime improvements exhibit diminishing returns as the number of regions increases. At higher grid resolutions (smaller grid sizes), additional regions become viable due to the smoother region boundaries that emerge from increased spatial discretization.

4.2.2.2 Partition and Overlap

Now, the overlap and number of regions for the PO method are decided, where the results are formatted (*number of regions*)-PO-(*Overlap cells*), in Fig. 4.13 and Fig. 4.14. Notably, when using PO, each region is computed separately, with smaller regions than in the benchmark PCSBL; the sparsity level in each regional map may vary significantly



(a) Without splitting measurements



(b) With splitting measurements

Figure 4.11: Performance evaluation across varying numbers of regions K for CP using 40 nuScenes samples, with metrics re-evaluated at each threshold value. (a) Standard approach without splitting measurements overlapping multiple regions. (b) Modified approach where measurements overlapping multiple regions are split.

based on the scene. This may be the reason for the increased freespace error. Since both PO and CP employ similar region-based principles, evaluation focuses on two representative values: $K = 4$ and $K = 16$.

Beyond 4 regions, this approach discards the same information as the non-splitting variant of CP, resulting in poorly informed free space estimates along the borders of each region, and there is no information sharing across the regions. The fusing of regions based on less informed estimates fails to improve the AS-NMSE, as demonstrated by the 16-region case in Fig. 4.14. This effect is evident when comparing the minimal impact of overlap for 4 regions against the notably larger AS-NMSE increase observed for 16 regions with overlap. In the case of 16 regions, increasing the overlap marginally increases the detection rate as well as the free space error.

The introduction of overlapping regions creates a computational trade-off, as increasing the number of regions K simultaneously increases the number of occupancies that must be estimated. This is evident from Fig. 4.15, where the 16-region, 4-cell

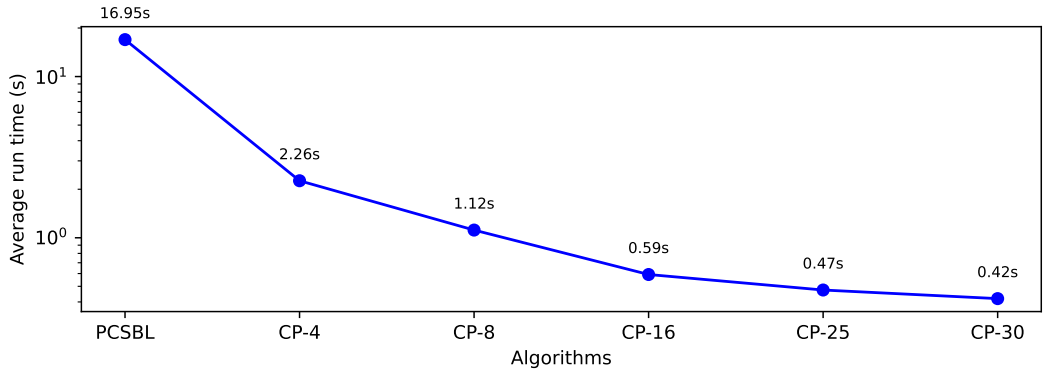


Figure 4.12: Runtimes across 40 samples for CP-based methods, for varying numbers of regions K

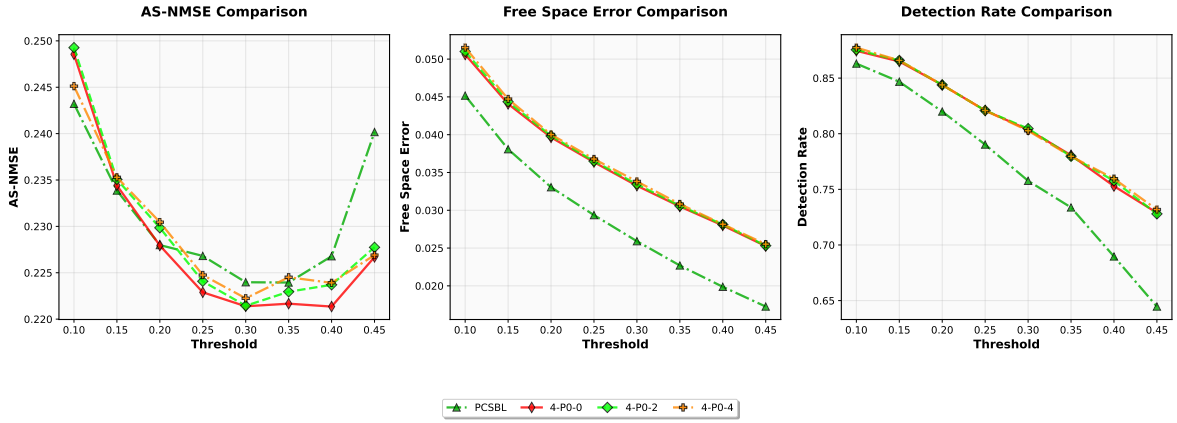


Figure 4.13: Performance evaluation for $K = 4$ regions using 40 nuScenes samples, with varying overlap and metrics re-evaluated at each threshold.

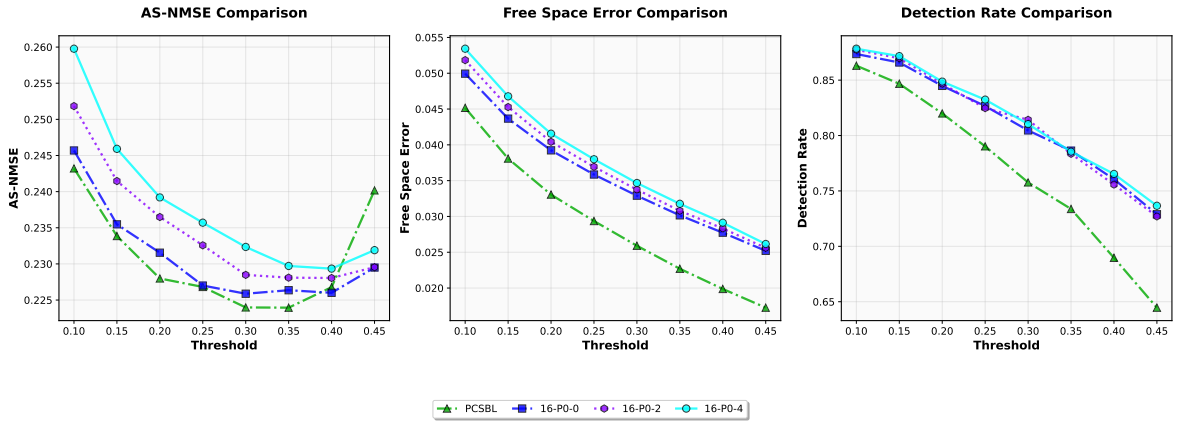


Figure 4.14: Performance evaluation for $K = 16$ regions using 40 nuScenes samples, with varying overlap and metrics re-evaluated at each threshold

overlapped variant takes nearly as long as 4 regions with 2 cells overlapped.

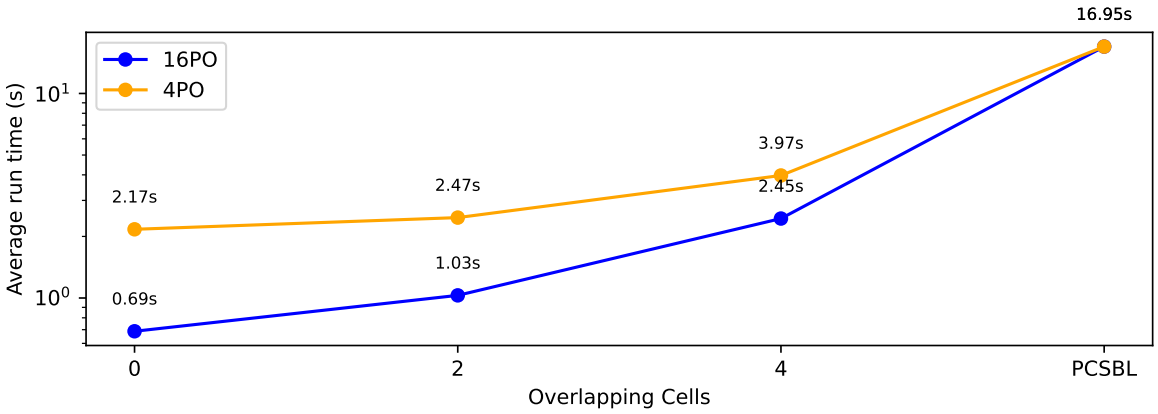


Figure 4.15: Runtimes across 40 samples for 4 and 16 regions PO methods, for varying numbers of overlapping cells. Compared with the benchmark PCSBL.

At this stage, it is already evident that the CP, which has coupling along the region boundaries and therefore no need for these overlapping cells, is computationally more efficient and also does not have the issue of varying sparsity per region, resulting in a lower free space error. While CP may be faster than PO, PO is easier to implement. The partition overlap method is tested using $K = 16$ and an overlap of two cells is chosen to highlight the added computational complexity and the threshold is once again shifted $\tau_{PO} = 0.35$. For brevity, PO is used to refer to PCSBL with $K = 16$ regions and a two cell overlap.

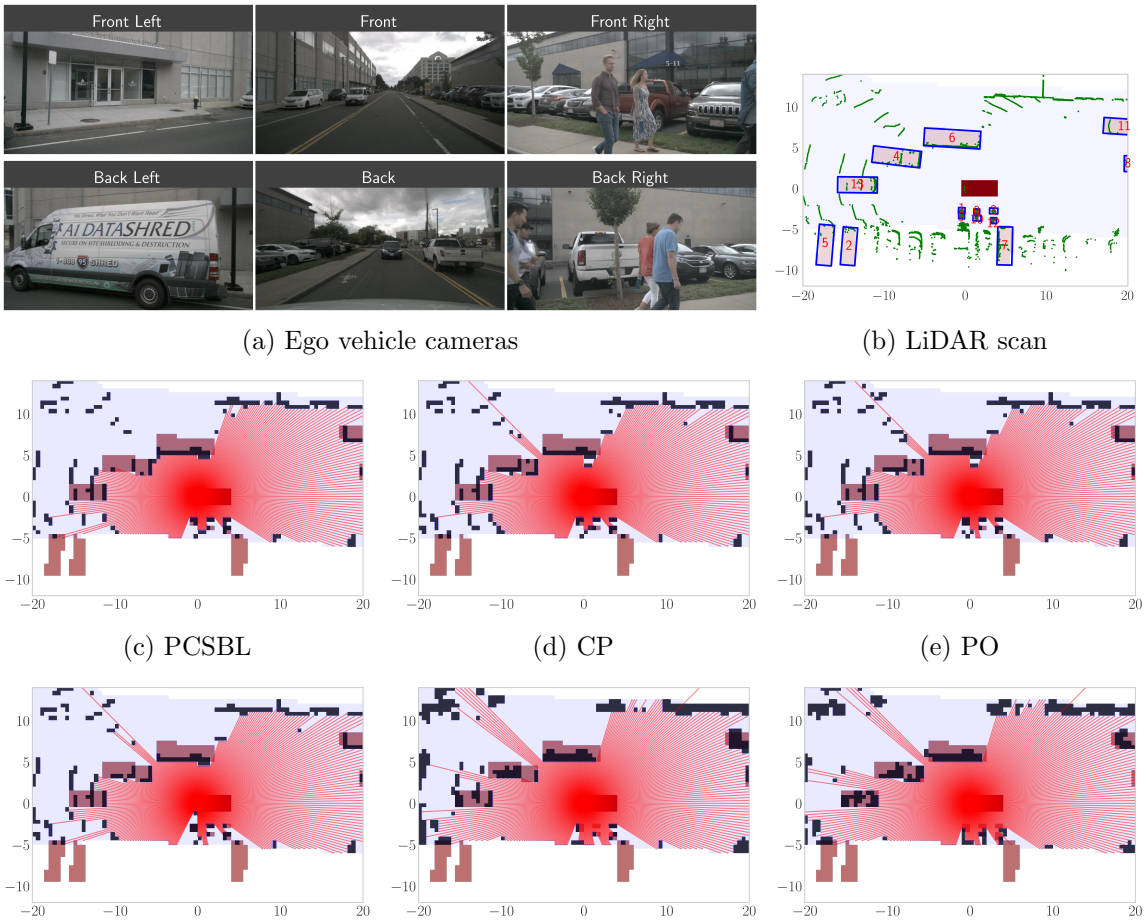
4.3 Results

Using the identified parameters, we now quantitatively and qualitatively evaluate the accelerated PCSBL variants against the benchmark methods. All PCSBL variants use $K = 16$ regions for PO and CP accelerations. We evaluate our methods across two datasets with different modality focuses. For nuScenes, we evaluate single-modality methods on LiDAR data, including the CP method, the PO method with a two-cell overlap, and the quadtree-accelerated CP method (QCP). The benchmark methods are BGK [21], ISM [8], and baseline PCSBL [24]. The RADIATE dataset is used to evaluate the fusion methods as well as the single modality radar PCSBL.

4.3.1 Results on nuScenes

We compare single modality LiDAR on a single scene and provide statistical results over 200 scenes. Fig. 4.16 and Table 4.6 show evaluations on scene 204 from nuScenes, with $M_{\mathrm{L}} = 6231$ LiDAR measurements and a map of $N = 2926$ cells. Quantitative results from Table 4.6 show that the PCSBL variants run up to 20 times faster than the benchmark PCSBL and achieve the same detection rate while incurring only a marginal increase in free space error and AS-NMSE.

4.3.1.1 nuScenes Scene 204 Frame 0



(a) Ego vehicle cameras

(b) LiDAR scan

(c) PCSBL

(d) CP

(e) PO

(f) QCP

(g) BGK

(h) ISM

Figure 4.16: Occupancy mapping comparison on nuScenes scene 204, frame 0. Inputs include ego vehicle camera views (a) and the LiDAR scan with object IDs (b). Occupancy maps for benchmark and accelerated variants (c–h) show the ego vehicle (dark red), ground truth boxes (pink), estimated occupancy (black), and road mask (pale blue). The vehicle front faces right.

Table 4.6: Results of Model Comparison on nuScenes LiDAR scene 204 frame 0

Metric	ID	Methods					
		PCSBL	**CP**	**PO**	**QCP**	**BGK**	**ISM**
IoBB	0	1.000	1.000	1.000	1.000	0.000	0.000
	1	1.000	1.000	1.000	1.000	0.500	0.000
	2	0.000	0.000	0.000	0.000	0.000	0.000
	3	1.000	1.000	1.000	0.500	1.000	0.000
	4	0.188	0.313	0.292	0.271	0.396	0.479
	5	0.027	0.054	0.027	0.054	0.108	0.081
	6	0.279	0.279	0.279	0.279	0.492	0.459
	7	0.028	0.028	0.028	0.028	0.028	0.000
	8	0.500	0.500	0.500	0.500	0.750	0.000
	9	0.500	0.500	1.000	1.000	0.500	0.500
	10	0.500	0.500	0.500	0.500	0.500	1.000
	11	0.348	0.348	0.348	0.348	0.696	0.565
	12	0.750	0.750	0.750	0.750	0.500	0.250
	13	0.333	0.333	0.333	0.333	0.472	0.667
Detected Targets		**13**/14	**13**/14	**13**/14	**13**/14	12/14	8/14
AS-NMSE		0.244	0.274	0.271	0.280	0.445	0.394
Freespace Error		**0.045**	0.051	0.053	0.048	0.079	0.067
Time (s)		8.254	0.428	1.420	**0.411**	0.165	0.178

Although BGK and ISM have good IoBB and runtimes, they suffer from poor free space error and lower AS-NMSE because of inaccurate edge detection. PCSBL outperforms BGK and ISM on all metrics, despite lower IoBB for large objects (e.g., targets 4, 6, 11, and 13). The higher IoBB values of BGK and ISM for large objects come at the expense of increased false alarms and failure to detect many pedestrians. PCSBL and its variants are significantly better at pedestrian detection. Meanwhile, BGK misses one pedestrian (target 0), and ISM misses two (target 0 and 3). Finally, the CP method is significantly closer to BGK and ISM in terms of runtime than the benchmark PCSBL.

4.3.1.2 Statistical Results

Further, Fig. 4.17 and Table 4.7 show the statistical performance of the methods averaged over 200 random scenes from nuScenes. Fig. 4.17 shows that the proposed PCSBL variants achieve similar performance compared to the benchmark PCSBL. It also shows that the BGK and ISM have higher AS-NMSE and lower detection rate compared to the PCSBL and its variants, which is similar to the observations in scene 204. The results in Table 4.7 show that across the three PCSBL algorithms, the proposed methods perform similarly to benchmark PCSBL, while the fastest PCSBL variant runs on average 27 times faster than benchmark PCSBL.

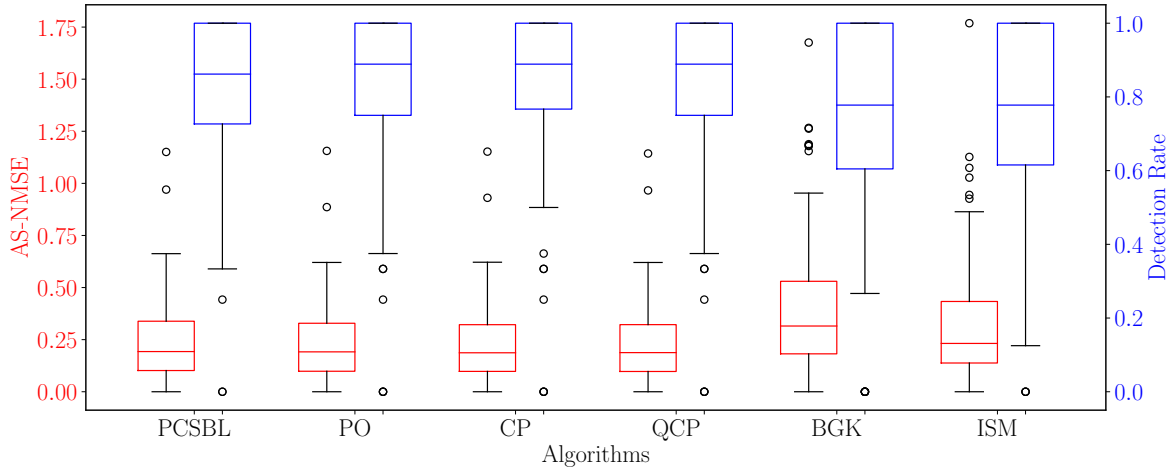


Figure 4.17: Distribution of the proportion of detected objects and AS-NMSE in the ground truth maps over 200 random LiDAR nuScenes samples.

Table 4.7: Mean runtime over 200 random LiDAR nuScenes samples.

Method	**PCSBL**	**CP**	**PO**	**QCP**	**BGK**	**ISM**
Time (s)	12.990	0.672	1.405	**0.474**	0.170	0.167

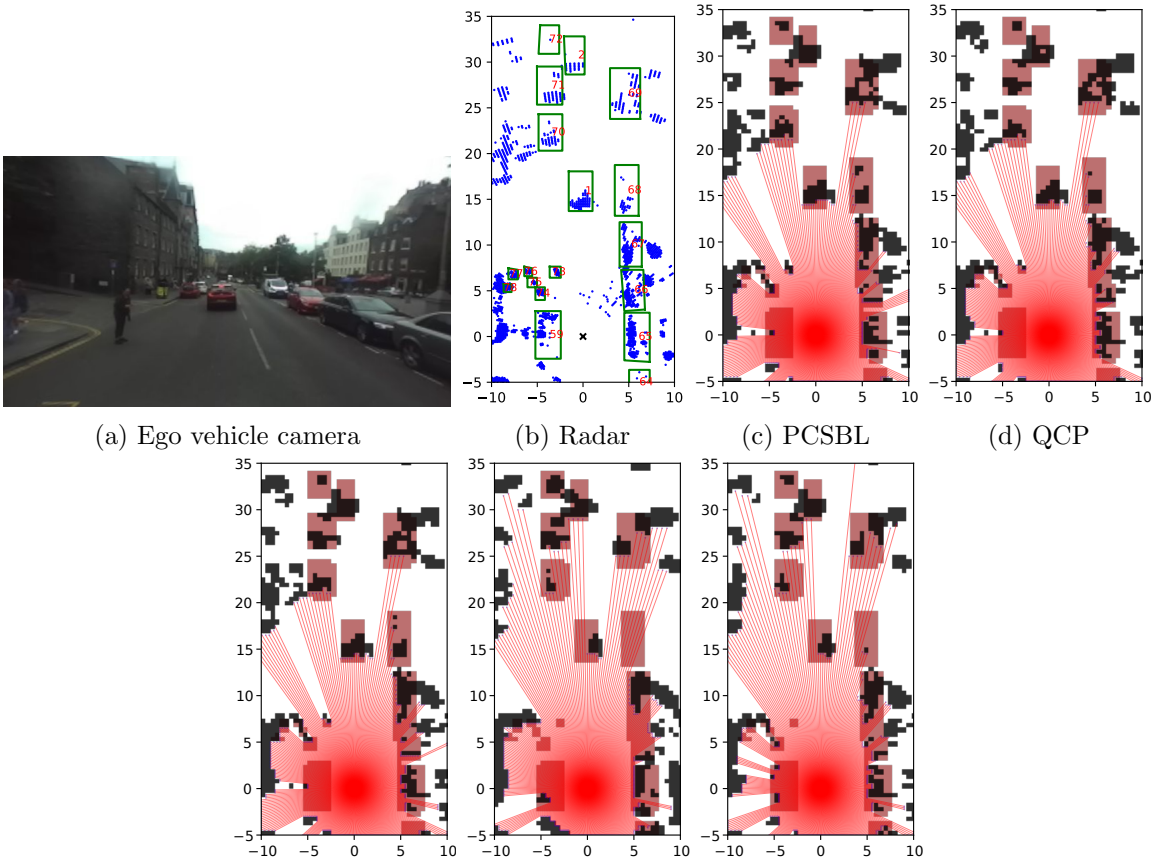
4.3.2 Results on RADIATE

For RADIATE, we evaluate both radar-only methods (PCSBL, CP, QCP) and fusion-based methods combining LiDAR and radar. The fusion-based methods include CS and CIS [6] along with their CP and QCP accelerated variants (CP-CS, QCP-CS, QCP-CIS, CP-CIS). PO is omitted as it has been shown that CP performs the same or better in most cases while being faster.

We evaluate single-modal and multi-modal PCSBL, presenting single scene results, statistical results over 40 samples, and time complexity as a function of the number of cells. Radar modalities from city scene 3-7 are used to compare the accelerated methods. Additionally, LiDAR and radar modalities from city scene 3-0 are used to compare the accelerated CS and CIS to the benchmark versions in Fig. 4.19, where minimal differences are observed, the same as with the single modality PCSBL.

4.3.2.1 RADIATE Scene 3-7 Frame 120

This scene is used to evaluate the radar modality on benchmark algorithms PCSBL [24], ISM [8], and BGK, as well as the QCP and CP accelerated variants. The scene within the grid parameters specified in Table 4.3b contains $M_{\mathrm{R}} = 1720$ radar measurements.



(a) Ego vehicle camera (b) Radar (c) PCSBL (d) QCP

(e) CP (f) BGK (g) ISM

Figure 4.18: Radar-only occupancy mapping comparison on RADIATE City Scene 3-7, frame 120. Shows ego vehicle camera view (a), radar points (b), and occupancy maps for various accelerated and benchmark methods with ground truth boxes (pink/green outlines).

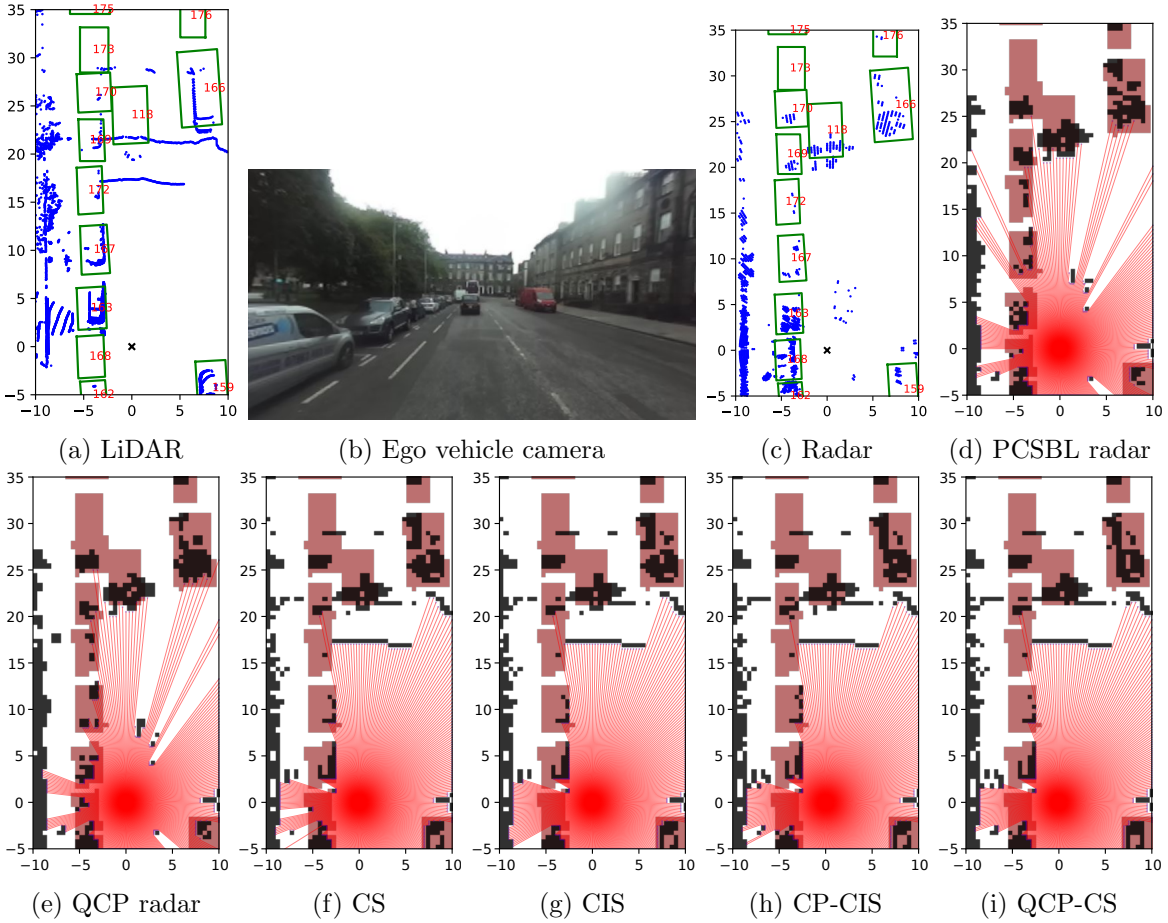
All methods successfully eliminate false CFAR detections directly ahead of the vehicle. However, their performance varies significantly in other areas. BGK and ISM demonstrate excellent computational efficiency on radar data but struggle with pedestrian detection in occupancy mapping. BGK fails to detect pedestrians (targets 72, 73, 74) and misses vehicle target 68, while ISM only fails to detect pedestrian target 73. The PCSBL variants consistently detect all pedestrians and vehicles, with closely matched IoBB values across variants. However, target 59 proves challenging for most methods. While faster methods like BGK and ISM offer computational advantages, the accelerated PCSBL variants provide the most comprehensive and accurate detection results

Table 4.8: Results of RADIATE Scene City-3-7 Frame 120 for Radar only

Metric	ID	Methods				
		PCSBL	**QCP**	**CP**	**BGK**	**ISM**
IoBB	1	0.422	0.400	0.422	0.156	0.244
	2	0.500	0.472	0.500	0.500	0.500
	59	0.015	0.015	0.015	0.062	0.062
	64	0.025	0.025	0.025	0.225	0.075
	65	0.317	0.381	0.317	0.365	0.333
	66	0.191	0.213	0.149	0.106	0.149
	67	0.520	0.560	0.540	0.480	0.580
	68	0.217	0.167	0.200	0.000	0.017
	69	0.451	0.463	0.488	0.268	0.317
	70	0.314	0.255	0.275	0.118	0.137
	71	0.354	0.417	0.438	0.500	0.417
	72	0.167	0.133	0.133	0.467	0.167
	73	0.750	0.250	0.250	0.000	0.000
	74	0.250	0.125	0.125	0.000	0.125
	75	0.500	0.500	0.500	0.000	0.250
	76	0.500	0.750	0.750	0.500	0.750
	77	0.667	0.667	0.667	0.667	0.667
	78	0.750	0.750	0.750	0.000	0.250
Detected Targets		**18**/18	**18**/18	**18**/18	13/18	17/18
AS-NMSE		**0.229**	0.234	0.239	0.411	0.364
Free-Space Error		0.085	0.084	0.086	0.090	**0.071**
Time (s)		6.115	0.623	0.745	**0.093**	**0.087**

4.3.2.2 RADIATE Scene 3-0 Frame 275

LiDAR and radar modalities from city scene 3-0 are used to compare the accelerated CS and CIS to the benchmark versions in Fig. 4.19, where minimal differences are observed, the same as with the single modality PCSBL. The similarity in performance between the methods is reinforced by the closeness of the AS-NMSE free-space error in Table 4.9, and most of the IoBB values are mostly the same or very close for different objects across the PCSBL fusion variants. The scene consists and $M_{\mathrm{L}} = 5702$ LiDAR and $M_{\mathrm{R}} = 1245$ radar measurements. The acceleration methods on CS fusion decreased the runtime by more than 10 times with CP and 22 times with QCP. Benchmark CIS runs approximately 17 times faster with QCP-CIS, but is still very slow at 12.2 seconds.



(a) LiDAR (b) Ego vehicle camera (c) Radar (d) PCSBL radar

(e) QCP radar (f) CS (g) CIS (h) CP-CIS (i) QCP-CS

Figure 4.19: Occupancy mapping comparison on RADIATE City Scene 3-0, frame 275. Modalities include ego vehicle camera view (a), radar points (b), and LiDAR points (c) with object IDs in red. Occupancy maps for benchmark methods PCSBL (d), CS (f), CIS (g) and their accelerated variants (e,h,i) show estimated occupancy with ground truth boxes (pink/green outlines).

Table 4.9: Results of RADIATE Scene City-3-0 Frame 275 for Radar and Fusion of LiDAR and Radar

Metric	ID	Methods							
		PCSBL radar	**QCP radar**	**CS**	**CP-CS**	**QCP-CS**	**CIS**	**CP-CIS**	**QCP-CIS**
IoBB	118	0.432	0.409	0.375	0.364	0.386	0.375	0.352	0.330
	159	0.105	0.118	0.224	0.237	0.237	0.224	0.224	0.211
	162	0.260	0.260	0.260	0.260	0.260	0.260	0.260	0.260
	163	0.268	0.232	0.286	0.304	0.304	0.286	0.286	0.286
	166	0.402	0.423	0.445	0.423	0.394	0.445	0.409	0.372
	167	0.180	0.197	0.115	0.164	0.098	0.115	0.147	0.049
	168	0.160	0.180	0.100	0.080	0.100	0.060	0.060	0.080
	169	0.298	0.213	0.149	0.213	0.170	0.149	0.213	0.149
	170	0.271	0.288	0.237	0.237	0.237	0.237	0.237	0.237
	172	0.113	0.094	0.019	0.019	0.019	0.019	0.019	0.019
	173	0.000	0.000	0.056	0.056	0.056	0.056	0.056	0.056
	175	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	176	0.200	0.200	0.200	0.200	0.200	0.200	0.200	0.200
Detected		11/13	11/13	**12**/13	**12**/13	**12**/13	**12**/13	**12**/13	**12**/13
AS-NMSE		0.243	0.200	**0.091**	0.092	0.093	0.103	0.094	0.101
Free-Space		**0.055**	0.058	0.068	0.069	0.070	0.068	0.069	0.063
Runtime(s)		7.405	**0.610**	14.989	1.412	**0.669**	210.241	22.621	12.221

4.3.2.3 Statistical Results

Further, Fig. 4.20 and Table 4.10 show the statistical performance of the multi-modal and radar PCSBL methods averaged over 40 scenes from RADIATE. It is clear that the single- and multi-modal accelerations perform closely to the PCSBL benchmarks [6,24] methods while outperforming the BGK [21] and ISM [8] benchmarks in Fig. 4.20. Notably, the QCP acceleration method on CS more significantly altered the AS-NMSE than the CP method alone.

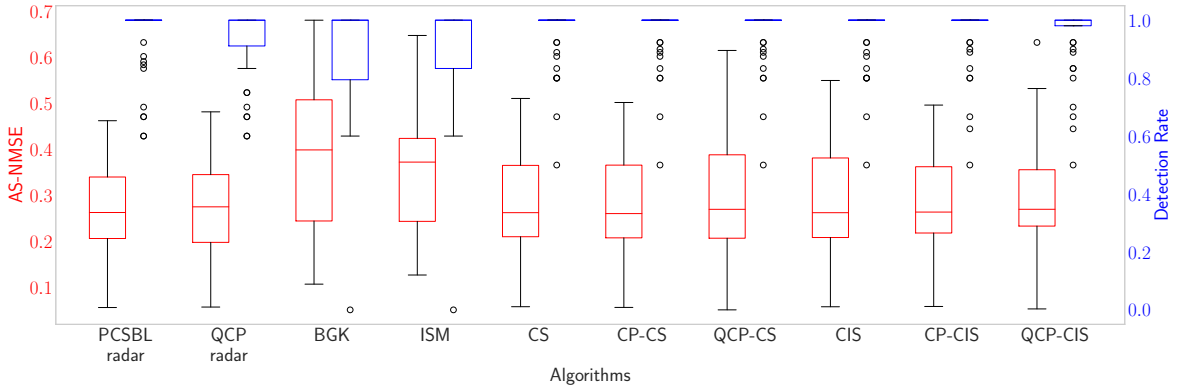


Figure 4.20: Distribution of the proportion of detected objects and AS-NMSE in the ground truth maps over 40 RADIATE samples.

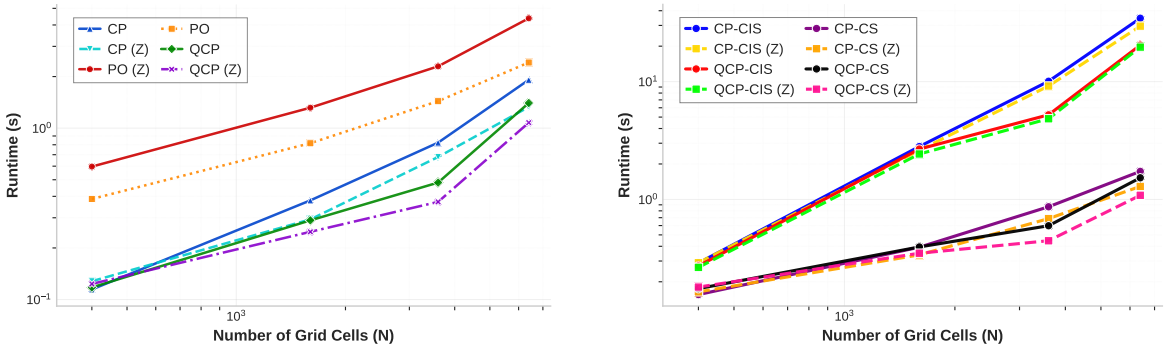
Table 4.10: Mean runtime over 40 RADIATE samples.

Method	**PCSBL radar**	**QCP radar**	**BGK**	**ISM**	**CS**	**CP-CS**	**QCP-CS**	**CIS**	**CP-CIS**	**QCP-CIS**
Time(s)	6.871	**0.623**	0.093	0.088	14.593	1.481	**0.709**	208.483	20.884	8.771

4.3.2.4 Time Complexity

The runtime performance is analyzed in Fig. 4.22 and Fig. 4.21 across varying cell counts N. Performance was evaluated on frame 275 (city-3-0, RADIATE dataset) using LiDAR within a 20×20m ego vehicle grid, with map resolution varied by adjusting cell size.

PCSBL and its variants have more computational cost associated with the number of cells; therefore, only the number of cells is varied because it is more informative than varying the number of measurements. First, the zero column and quadtree accelerations are compared on PO and CP in Fig. 4.21, and then all the accelerations are compared to the benchmark methods from [6, 8, 21, 24] in Fig. 4.22.



(a) Single modality

(b) Fusion of modalities

Figure 4.21: Time complexity analysis comparing accelerated methods for (a)single modality (b)fusion PCSBL variants with and without zero column optimisations (Z) across varying grid resolutions. Single-modality methods use LiDAR measurements, while fusion methods use LiDAR and radar measurements from RADIATE scene city 3-0 frame 275.

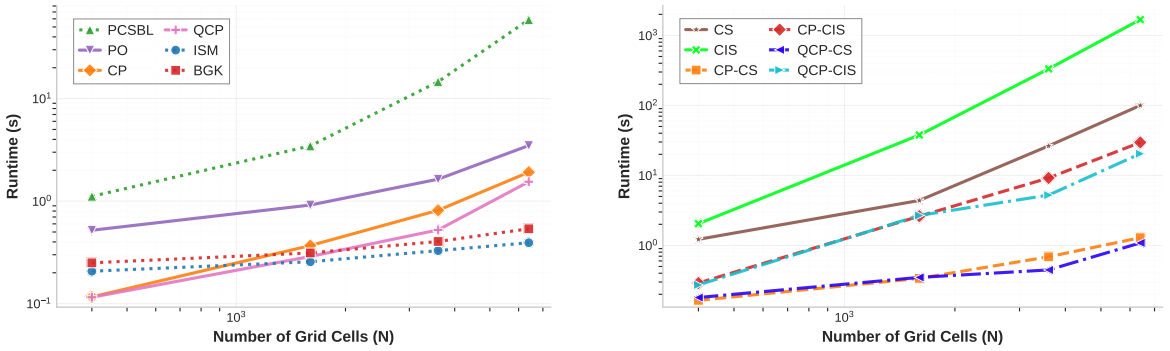
At very low resolutions, Fig. 4.21a reveals that applying zero column methods to the already efficient CP and QCP algorithms introduces computational overhead, resulting in longer execution times compared to their baseline implementations. However, this overhead becomes negligible as the problem size grows beyond approximately 1000 cells, at which point the benefits of zero column elimination become pronounced. The effectiveness of the zero column optimization is particularly evident when comparing CP(Z) to QCP, where the runtime performance becomes nearly equivalent, demonstrating that zero column elimination provides acceleration benefits comparable to the test-data quadtree for the CP method. The PO method also appears to benefit more significantly from the zero column method than CP, likely because the difference between CP and PO runtime stems from the increased number of occupancies PO needs to estimate.

The fusion methods in Fig. 4.21b demonstrate different acceleration benefits depending on the fusion approach. For CS fusion, both zero-column optimization and quadtrees (CS-CP, CS-QCP, CS-CP(Z), CS-QCP(Z)) provided notable runtime improvements comparable to their single-modality performance. However, for CIS fusion, the zero column optimisation yielded only marginal, almost invisible gains in total

computational runtime. This reduced effectiveness in CIS fusion may stem from the computational overhead of the permutation of the three times larger selection matrix. Additionally, quadtree accelerations show significant performance gains only at higher resolutions when applied to multi-modal fusion, unlike in single-modal cases. Using multiple modalities introduces more occupied states into quadtree nodes, reducing the chance that nodes contain only free space and making cell pruning less effective. Despite this, significant performance gains are still realised.

Now, comparing all the accelerated PCSBL variants to the benchmark methods applied in Fig. 4.22. The accelerated methods demonstrate superior performance compared to BGK and ISM when estimating occupancy for relatively small numbers of cells. However, as the cell count increases, the QCP and CP methods exhibit less favourable scaling characteristics than BGK and ISM. PO, which requires calculation of a larger number of occupancies, operates significantly slower than CP while maintaining a notable performance advantage over PCSBL. The accelerated CS fusion methods achieve runtime performance levels that are close to the accelerated single-modality counterparts. In contrast, CP-CIS and its quadtree variant have runtime performance characteristics that align more closely with the benchmark CS PCSBL than with the other accelerated methods.

The accelerated methods demonstrate superior performance compared to BGK and ISM when estimating occupancy for relatively small numbers of cells. However, as the cell count increases, the QCP and CP methods exhibit less favourable scaling characteristics than BGK and ISM. The CP and PO methods exhibit better scaling behaviour with increasing N due to their $\mathcal{O}(N^3/K^2)$ complexity versus $\mathcal{O}(N^3)$ for benchmark PCSBL. While both accelerated methods significantly outperform PCSBL, PO operates more slowly than CP due to additional occupancy calculations and separate selection matrix overhead.



(a) Single-modality methods

(b) Multi-modal fusion methods

Figure 4.22: Time complexity analysis comparing benchmark and accelerated methods across varying grid resolutions. Single-modality methods use LiDAR measurements, while fusion methods combine LiDAR and radar measurements from RADIATE scene city 3-0 frame 275.

4.4 Summary

This chapter began by presenting the evaluation techniques, the selection of parameters for the acceleration methods, and the resulting performance in terms of runtime and the evaluation metrics described earlier.

The nuScenes and RADIATE datasets were introduced, detailing the provided ground truth data, map information, and sensor modalities. nuScenes offers LiDAR and radar data, but only the LiDAR modality is evaluated due to the overly sparse preprocessed radar point cloud. In contrast, RADIATE is used to evaluate both radar alone and LiDAR–radar fusion. Its radar data is captured from a mechanically scanning radar in range–azimuth format, enabling improved azimuth resolution compared to typical automotive MIMO radars, and allowing for custom preprocessing to generate the desired input.

The evaluation metrics—AS-NMSE, IoBB (with the related detection rate), and free space error—were explained, along with the occupancy grid map (OGM) parameters for map size and resolution. Parameters for the benchmark single-modality and multi-modal fusion methods were also outlined.

Using the specified map parameters, the acceleration methods were tuned to select optimal configurations. This included choosing the number of points $U = 25$ per cell for the ray lookup table, justifying the use of the splitting method for CP, and selecting $K = 16$ regions for both PO and CP, with the CP threshold adjusted upward to maintain comparable free space error. At this stage, it was already observed that CP is faster than PO with no need for the additional overlap across regions that PO introduces.

The selected parameters were then applied for a full comparison against benchmark methods on both datasets. On nuScenes, LiDAR-only evaluation over 200 samples demonstrated that the accelerated PCSBL variants produce occupancy maps of very similar quality to the benchmark PCSBL while achieving an average 27× speed-up. The PO method with overlap was noted to be significantly more computationally expensive due to the larger number of occupancies that must be calculated.

On RADIATE, both single-modality radar and LiDAR–radar fusion were evaluated. For CS fusion, runtime was improved by 20× to sub-second execution times, while CIS fusion with QCP achieved a 17× speed-up but remained slower overall ($> 20\,\mathrm{s}$). Across both datasets, the accelerations maintained detection rates and AS-NMSE close to the benchmarks, with only minor increases in free space error.

Finally, a time complexity analysis was performed for varying grid cell counts. The accelerated single-modality and CS fusion PCSBL variants outperformed all benchmark methods at lower map resolutions. CIS was still relatively slow even after applying accelerations. At higher resolutions, BGK and ISM scaled more favourably.

5 Conclusion and Future Work

5.1 Conclusion

PCSBL for OGM offers state-of-the-art mapping performance but suffers from cubic scaling in the number of cells, leading to high computational cost. This thesis addressed these bottlenecks by introducing practical accelerations that maintain mapping quality while substantially reducing runtime.

We introduced an approximate raycasting with a sparse matrix construction method to speed up the selection matrix construction, extended test-data quadtrees to PCSBL to accommodate variable map resolution, and proposed two computationally efficient PCSBL techniques (PO and CP) that exploit the selection matrix for OGM reconstruction. The fast selection matrix construction reduces the computational complexity of the selection matrix

Evaluations on the nuScenes and RADIATE datasets demonstrated that these methods maintain detection accuracy and AS-NMSE comparable to benchmark single-modality and fusion PCSBL approaches, while significantly reducing runtime and only marginally increasing free space errors. Among the acceleration techniques, the CP method is more computationally efficient than the PO method, since PO calculates more cell occupancies due to overlapping regions. Unlike PO, CP preserves the sparsity level by not computing independent sub-maps. Furthermore, CP is further accelerated using quad trees; for single-modality methods, this acceleration has minimal impact on the quality of the maps for single-modality PCSBL. However, for fusion methods and single modality radar, QCP acceleration affects AS-NMSE or detection rate more than the CP acceleration technique. These effects are slight, and the worst accelerated methods still outperform BGK and ISM in quality but do not scale as well with increasing map resolution in time complexity. Accelerated single-modality PCSBL and CS fusion methods achieve sub-second runtimes at practical grid resolutions. Although CIS methods also benefit from acceleration with notable runtime improvements, they remain too slow for practical use.

The CP and PO methods achieve a quadratic reduction in cubic complexity based on the number of regions. Test-data quadtrees reduce computational overhead by reducing the number of cells requiring occupancy estimation. The CSR selection matrix construction makes the complexity dependent on the sparsity of the matrix, while the ray lookup table improves the complexity from linear scaling in both measurements and cells to linear scaling in measurements with logarithmic scaling in cells.

To conclude, the accelerations introduced in this work substantially reduce runtime for both single- and multi-modal fusion PCSBL, with minimal impact on metrics. Although PCSBL has been significantly sped up, further research outlined in section 5.2 could yield even greater efficiency, which may be important for edge devices.

5.2 Future work

- *Polar grids* instead of rectangular grid: Using polar grids would completely separate measurement rays (in the case of LiDAR, not radar due to the beam sensor model), meaning that measurements within the selection matrix cannot intersect multiple regions, where the region would perfectly align with a subset of angular sectors. Therefore, the number of regions K can be increased while remaining mathematically the same as the benchmark PCSBL algorithms. Using polar grids may make it infeasible to use quad trees in their current form. Trees could still be used to spatially define polar grid cells; however, this is a direction that would require investigation. It is worth noting that while polar grids do not adapt its resolution dynamically based on the scene like quadtrees, they do have variable resolution due to the increasing size in cells radially. Cells closer to the vehicle are smaller, and cells further away are larger.

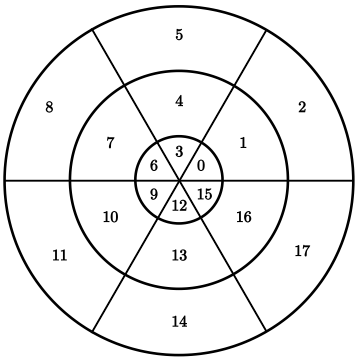


Figure 5.1: Radially indexed polar grid

- *Iterative temporal PCSBL:* Currently this work has only taken into account static occupancy grid maps, and therefore calculates T iterations per map, however passing prior information from the previous frame may help PCSBL converge faster at the following time step allowing the number of iterations to reduce from the constant static map number of iterations.

- *Selection matrix values experimentation:* Instead of having only binary $\{0, 1\}$ values within the selection matrix, the values corresponding to measurement rays can be weighted continuously within $[0, 1]$ based on their intersection geometry with grid cells. The weighting could be determined by the fractional length a ray intersects a cell, providing a more accurate representation of the measurement's influence on each grid cell.

- *Inversion-free methods:* With matrix inversion being the most computationally expensive part of PCSBL, inversion-free methods offer significant potential for computational acceleration. Generalised Approximate Message Passing [49] (GAMP) was briefly investigated as an alternative that approximates the posterior distribution rather than computing the MAP estimate directly, thereby eliminating matrix

inversion requirements. It is well known that GAMP requires a well-conditioned $\boldsymbol{A}$ matrix, something which our selection matrix is not. However, using the fractional values previously mentioned could improve the condition of the selection matrix.

Bibliography

[1] S. O.-R. A. V. S. Committee *et al.*, "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems," *SAE J.*, vol. 3016, p. 1, 2014.

[2] A. Yoganandhan, S. Subhash, J. H. Jothi, and V. Mohanavel, "Fundamentals and development of self-driving cars," *Mater. Today Proc.*, vol. 33, pp. 3303–3310, 2020.

[3] 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, 2019.

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

[5] H. P. Moravec, "High resolution maps from wide angle sonar," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1985, pp. 116–121.

[6] P. Zhai, G. Joseph, N. J. Myers, C. Önen, and A. Pandharipande, "Sparsity-aware occupancy grid mapping for automotive driving using radar-lidar fusion," in *2024 IEEE SENSORS*, 2024, pp. 1–4.

[7] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, 2016.

[8] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, ser. Intell. Robot. Auton. Agents Ser. Cambridge, MA, USA: MIT Press, 2005.

[9] M. Baum and U. D. Hanebeck, "Tracking extended objects using a random hypersurface model," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 44, no. 3, pp. 1541–1550, 2008.

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

[11] J. Revelles, C. Ureña, and M. Lastra, "An efficient parametric algorithm for octree traversal," *Journal of WSCG*, vol. 8, no. 1, pp. 212–219, 2000.

[12] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[13] J. Zhou and D. Wen, "Research on ray tracing algorithm and acceleration techniques using kd-tree," in *Proc. Int. Conf. Intell. Comput. Signal Process.*, 2021, pp. 107–110.

[14] C. H. Walsh and S. Karaman, "Cddt: Fast approximate 2D ray casting for accelerated localization," in *Proc. IEEE Int. Conf. Robot. Autom.* IEEE, 2018, pp. 3677–3684.

[15] S. O'Callaghan, Fabio. T. Ramos, and H. Durrant-Whyte, "Contextual occupancy maps using Gaussian processes," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 1054–1060.

[16] J. Wang and B. Englot, "Fast, accurate Gaussian process occupancy maps via test-data octrees and nested Bayesian fusion," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1003–1010.

[17] L. Piotrowski, *Robotic Systems: Advanced Techniques and Applications.* Dordrecht: Springer Netherlands, 1992.

[18] G. Kraetzschmar, "Probabilistic quadtrees for variable-resolution mapping of large environments," *IFAC Proc. Vol.*, vol. 37, no. 8, pp. 695–700, Jul. 2004.

[19] Y. Chen, W. Shuai, and X. Chen, "A probabilistic, variable-resolution and effective quadtree representation for mapping of large environments," in *Proc. Int. Conf. Adv. Robot.*, 2015, pp. 605–610.

[20] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 4093–4098.

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

[22] K. Doherty, T. Shan, J. Wang, and B. Englot, "Learning-aided 3-D occupancy mapping with Bayesian generalized kernel inference," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 953–966, 2019.

[23] S. Kim and J. Kim, "Recursive Bayesian updates for occupancy mapping and surface reconstruction," *Proc. Australas. Conf. Robot. Autom.*, 2014.

[24] C. Önen, A. Pandharipande, G. Joseph, and N. J. Myers, "Occupancy grid mapping for automotive driving exploiting clustered sparsity," *IEEE Sens. J.*, vol. 24, no. 7, pp. 9240–9250, 2024.

[25] 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, 2015.

[26] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012.

[27] 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 Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11 621–11 631.

[28] D. Bauer, L. Kuhnert, and L. Eckstein, "Deep, spatially coherent inverse sensor models with uncertainty incorporation using the evidential framework," in *Proc. IEEE Intell. Veh. Symp.* IEEE, 2019, pp. 2490–2495.

[29] R. Cheng, C. Agia, Y. Ren, X. Li, and B. Liu, "S3cnet: A sparse semantic scene completion network for lidar point clouds," in *Proc. Conf. Robot. Learn.* PMLR, 2021, pp. 2148–2161.

[30] L. Sless, B. E. Shlomo, G. Cohen, and S. Oron, "Road scene understanding by occupancy grid learning from sparse radar clusters using semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop*, 2019, pp. 867–875.

[31] R. Weston, S. Cen, P. Newman, and I. Posner, "Probably unknown: Deep inverse sensor modelling radar," in *Proc. IEEE Int. Conf. Robot. Autom.* IEEE, 2019, pp. 5446–5452.

[32] J. Wilson, J. Song, Y. Fu, A. Zhang, A. Capodieci, P. Jayakumar, K. Barton, and M. Ghaffari, "Motionsc: Data set and network for real-time semantic mapping in dynamic environments," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 8439–8446, 2022.

[33] X. Yan, J. Gao, J. Li, R. Zhang, Z. Li, R. Huang, and S. Cui, "Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 4, 2021, pp. 3101–3109.

[34] C. B. Rist, D. Emmerichs, M. Enzweiler, and D. M. Gavrila, "Semantic scene completion using local deep implicit functions on lidar data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 7205–7218, 2021.

[35] M. Schreiber, V. Belagiannis, C. Gläser, and K. Dietmayer, "A multi-task recurrent neural network for end-to-end dynamic occupancy grid mapping," in *Proc. IEEE Int. Veh. Symp.* IEEE, 2022, pp. 315–322.

[36] A. Popov, P. Gebhardt, K. Chen, R. Oldja, H. Lee, S. Murray, R. Bhargava, and N. Smolyanskiy, "NVRadarNet: Real-time radar obstacle and free space detection for autonomous driving," 2023.

[37] A.-Q. Cao, A. Dai, and R. De Charette, "Pasco: Urban 3D panoptic scene completion with uncertainty awareness," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 14 554–14 564.

[38] F. Ding, X. Wen, Y. Zhu, Y. Li, and C. X. Lu, "Radarocc: Robust 3D occupancy prediction with 4D imaging radar," *Adv. Neural Inf. Process. Syst.*, vol. 37, pp. 101 589–101 617, 2024.

[39] X. Wang, Z. Zhu, W. Xu, Y. Zhang, Y. Wei, X. Chi, Y. Ye, D. Du, J. Lu, and X. Wang, "Openoccupancy: A large scale benchmark for surrounding semantic occupancy perception," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 17 850–17 859.

[40] X. Tian, T. Jiang, L. Yun, Y. Mao, H. Yang, Y. Wang, Y. Wang, and H. Zhao, "Occ3D: A large-scale 3D occupancy prediction benchmark for autonomous driving," *Adv. Neural Inf. Process. Syst.*, vol. 36, pp. 64 318–64 330, 2023.

[41] J. Zhang, Y. Ding, and Z. Liu, "Occfusion: Depth estimation free multi-sensor fusion for 3D occupancy prediction," in *Proc. Asian Conf. Comput. Vis.*, 2024, pp. 3587–3604.

[42] R. Van Kempen, B. Lampe, T. Woopen, and L. Eckstein, "A simulation-based end-to-end learning framework for evidential occupancy grid mapping," in *Proc. IEEE Intell. Veh. Symp.*, 2021, pp. 934–939.

[43] 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.

[44] H. Rohling, "Radar cfar thresholding in clutter and multiple target situations," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-19, no. 4, pp. 608–621, Jul. 1983.

[45] A. Palffy, E. Pool, S. Baratam, J. F. P. Kooij, and D. M. Gavrila, "Multi-class road user detection with 3+1d radar in the view-of-delft dataset," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4961–4968, 2022.

[46] N. Fotopoulos, "Deep unrolled sparse Bayesian learning for occupancy grid mapping," Master's thesis, Delft University of Technology, Delft, The Netherlands, 2024, master's thesis, Faculty of EEMCS.

[47] G. K. Kraetzschmar, G. P. Gassull, and K. Uhl, "Probabilistic quadtrees for variable-resolution mapping of large environments," *IFAC Proc. Vol.*, vol. 37, no. 8, pp. 675–680, 2004.

[48] S. Muckenhuber, E. Museljic, and G. Stettinger, "Performance evaluation of a state-of-the-art automotive radar and corresponding modeling approaches based on a large labeled dataset," *J. of Intell. Transp. Syst.*, vol. 26, no. 6, pp. 655–674, 2022.

[49] 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, 2016.

[50] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, UK: Cambridge University Press, 2003.

Occupancy Grid Map Methods

A

A.1 Inverse Sensor Model

This appendix outlines the key assumptions underlying ISM, describes its occupancy state update mechanism, presents the complete algorithm, and specifies the parameters used in its implementation.

The ISM algorithm, implemented from [8] a extended version of vanilla ISM [4], makes use of the same sensor model outlined in Fig. 2.1b. However, ISM makes two problematic assumptions that simplify computation at the expense of accuracy. First, it assumes each cell in the occupancy grid is independent of every other cell:

$$p(\boldsymbol{x}) = \prod_{n=0}^{N-1} p(x[n])) \tag{A.1}$$

Second, it extends this assumption to state that given a set of measurements, each cell remains independent of every other cell in the map. These assumptions are not necessarily valid since occupancy grid cells often form part of larger physical structures. Nevertheless, these independence assumptions significantly simplify the model, enabling fast computation of occupancy grid maps.

A.1.1 Log-Odds Update Rule

ISM employs a log-odds representation to iteratively update occupancy probabilities in a numerically stable and computationally efficient manner. Using the same sensor model as radar for PCSBL (Fig. 2.1b) with beam width Ω and object thickness Δ, the algorithm processes each measurement as follows:

For cells within the conical beam leading up to the terminal points (free space region):

$$\ell_m[n] \quad = \ell_{m-1}[n] + \ell_{\text{free}}, \qquad \ell_{\text{free}} \quad = \log\left(\tfrac{p_{\text{free}}}{1-p_{\text{free}}}\right) \tag{A.2}$$

For cells in the terminal region where measurements are detected (occupied space):

$$\ell_m[n] \quad = \ell_{m-1}[n] + \ell_{\text{occ}}, \qquad \ell_{\text{occ}} \quad = \log\left(\tfrac{p_{\text{occ}}}{1-p_{\text{occ}}}\right) \tag{A.3}$$

Here, $\ell_m[n]$ denotes the log-odds of occupancy for cell n after processing the m-th measurement. The terms p_{occ} and p_{free} represent the inverse sensor model likelihoods: $p_{\text{occ}} = p(x[n] = 1|\boldsymbol{p}_m)$ (likelihood of occupation given measurement) and $p_{\text{free}} = p(x[n] = 0|\boldsymbol{p}_m)$ (likelihood of being unoccupied given measurement).

Note that vanilla ISM would compute only the current measurement's contribution without maintaining the running belief represented by the $\ell_{m-1}[n]$ term.

A.1.2 Probability Conversion and Thresholding

The final occupancy probability estimate for each cell is computed by converting from log-odds:

$$\hat{p}(x[n] = 1) = 1 - \frac{1}{1 + \exp(\ell[n])}. \tag{A.4}$$

The binary occupancy grid $\hat{\mathbf{x}}$ is then obtained by thresholding each probability estimate to either 0 (unoccupied) or 1 (occupied) using the threshold τ_{ISM}.

Algorithm 5 ISM

Input: Reflection point coordinates $\{\mathbf{p}_m\}_{m=0}^{M-1}$
Parameters: Beam width Ω, object thickness Δ, ISM parameters p_{free}, p_{occ}, and ℓ_0
Output: Binary occupancy grid map $\hat{\mathbf{x}}$

for each measurement m **do**
  for each grid cell n **do**
    if n is inside the conical beam of $\mathbf{p}_m$ **then**
      $\ell_m[n] \leftarrow \ell_{m-1}[n] + \ell_{\text{free}}$
    end if
    if n is one of the conical beam's terminal points **then**
      $\ell_m[n] \leftarrow \ell_{m-1}[n] + \ell_{\text{occ}}$
    end if
  end for
end for
for each grid cell n **do**
  Compute $\hat{p}(x[n] = 1)$ using Equation A.4
end for
Compute binary occupancy grid $\hat{\mathbf{x}}$ via threshold τ_{ISM}

From the algorithm, it follows that the complexity of ISM is given by $\mathcal{O}(ML)$, where L denotes the average number of cells within the beam per update, for M measurements.

Table A.1: ISM Parameters

Parameter	Description	Value
Ω	Beam Width	2°
Δ	Obstacle Thickness	2 grid cells
$p_{\text{occ}}, p_{\text{free}}$	Occupancy and Non-occupancy Probabilities	0.8, 0.2
τ_{ISM}	Threshold	0.5

A.2 Bayesian Gaussian Kernels

This appendix section explains how occupancies are represented in BGK [21], how the training data used to inform these occupancies is constructed, and how the occupancies are updated using spatial correlation methods. The complete BGK process is summarized in Algorithm 6, which utilizes the parameters outlined in Table A.2.

BGK [21], unlike ISM, expresses correlation between neighbouring cells, enabling more sophisticated spatial reasoning. While the BGK algorithm allows estimation of occupancy at any continuous point, the OGM framework is adopted for computational efficiency, considering discrete points where each cell is defined at the cell centre $\boldsymbol{p}_n^* = (p_{x,n}^*, p_{y,n}^*)$.

A.2.1 Probabilistic Cell Representation

Each grid cell is modelled using a Beta probability distribution $\text{Beta}(\alpha, \beta)$, providing uncertainty quantification through the variance term. The mean and variance at each grid cell are given by:

$$\mu[n] = \frac{\alpha[n]}{\alpha[n] + \beta[n]}, \tag{A.5}$$

$$\sigma^2[n] = \frac{\alpha[n]\,\beta[n]}{(\alpha[n] + \beta[n])^2(\alpha[n] + \beta[n] + 1)}. \tag{A.6}$$

A.2.2 Training Data Construction

To update the Beta parameters that inform occupancy values, BGK constructs training data using ray casting principles similar to PCSBL and ISM. The process involves sampling multiple points along rays from the sensor origin to each measured sensor point $\boldsymbol{p}_m$. Free space points are sampled every r_{free} meters along each ray and receive free space labels (zero), while the measurement endpoint $\boldsymbol{p}_m$ itself receives an occupied label (one). This generates a training dataset $\boldsymbol{P}_T$ containing M_T training points $\boldsymbol{p}_i$ (where i indexes all training points, including both free space samples and measurements) with corresponding binary labels $\boldsymbol{y}_T$.

A.2.3 Spatial Correlation Through Kernel Functions

BGK incorporates spatial correlation between cells through a kernel function. This kernel considers only training points $\boldsymbol{p}_i$ within distance l of the query point $\boldsymbol{p}_n^*$, defined as:

$$k(\boldsymbol{p}_i, \boldsymbol{p}_n^*) = \begin{cases} \sigma_0 \left[\frac{2+\cos(2\pi\frac{d}{l})}{3} \left(1 - \frac{d}{l}\right) + \frac{1}{2\pi}\sin(2\pi\frac{d}{l}) \right] & d < l, \\ 0 & d \geq l, \end{cases} \tag{A.7}$$

where the Euclidean distance between any training point $\boldsymbol{p}_i$ and cell centre is:

$$d = \|\boldsymbol{p}_i - \boldsymbol{p}_n^*\|_2. \tag{A.8}$$

The kernel length parameter l determines the extent of spatial correlation—larger values increase the influence region, allowing distant observations to affect a query cell's occupancy estimate.

A.2.4 Parameter Updates

Using the kernel weights, the Beta parameters for each cell are updated by incorporating both the training data and prior beliefs through parameters α_0 and β_0:

$$\alpha[n] = \alpha_0 + \sum_{i=1}^{M_{nn}} k(\boldsymbol{p}_i, \boldsymbol{p}_n^*)\, y[i], \tag{A.9}$$

$$\beta[n] = \beta_0 + \sum_{i=1}^{M_{nn}} k(\boldsymbol{p}_i, \boldsymbol{p}_n^*)\, (1 - y[i]). \tag{A.10}$$

For each query cell n, M_{nn} denotes the number of training points within kernel distance l of the current query point $\boldsymbol{p}_n^*$. The updated Beta parameters are then used to calculate the expected mean and variances for each cell using equations (A.5) and (A.6). Finally, the binary occupancy grid $\hat{\mathbf{x}}$ is obtained by thresholding each mean $\mu[n]$ using the threshold τ_{BGK}.

Algorithm 6 Bayesian Gaussian Kernel (BGK)

Input: Measurements $\{\boldsymbol{p}_m\}_{m=0}^{M-1}$, query centres $\{\boldsymbol{p}_n^*\}_{n=0}^{N-1}$
Parameters: kernel length l, scale σ_0, priors α_0, β_0, free-space sampling resolution r_{free}
Output: Occupancy map $\hat{\boldsymbol{x}}$

```
Initialize α[n] = α_0, β[n] = β_0 for all n
for each measurement p_m do
    Sample free space points with y = 0 every r_free meters along the ray to p_m
    Assign y = 1 to the measurement point p_m itself
    Insert all training points (p_i, y_i) into a kd-tree spatial index
end for
for each query cell center p*_n do
    Query points and labels within distance l from kd-tree (size M_nn)
    Compute weights k(p_i, p*_n) via Eq. (A.7)
    Update α[n] and β[n] using (A.9),(A.10)
    Compute μ(n), σ(n) via Eqs. (A.5), (A.6)
end for
Compute binary occupancy grid x̂ via the threshold τ_BGK
```

From the algorithm it can be seen that the complexity of BGK is given by $\mathcal{O}(N \log(M_T))$ where N is the total number of cells and $\log(M_T)$ is cost of querying the points for each cell.

Table A.2: BGK parameters

Parameter	**Description**	**Value**
σ_0	Kernel scale	0.1
α_0, β_0	Beta prior parameters	0.001
l	Kernel length	1m
r_{free}	Free space ray sampling resolution	1m
τ_{BGK}	Threshold	0.5

Constant False Alarm Rate

B

CFAR is first explained for use with a range-azimuth image, after which the the parameters are provided in a table Table B.1 for the algorithm explained in Algorithm 7.

Radar signals have noise levels which vary significantly, thus it is not a simple task to separate the true signal from the noise. Within a range-azimuth image which is represented in polar form $\boldsymbol{E} \in \mathbb{R}^{N_{\text{Range}} \times N_{\text{azimuth}}}$, where $N_{\text{Range}} \times N_{\text{azimuth}}$ is the radar image resolution. CFAR then iterates through each angular portion of the range-azimuth image (takes each azimuth direction $\boldsymbol{e}_i$ where $i = 1, 2, \ldots, N_{\text{azimuth}}$) and then calculates a variable threshold in order to classify elements within the current azimuth direction as signal or noise. The threshold is determined by calculating the noise power for each cell under test, which is the specific range-azimuth cell currently being evaluated for the presence of a target. Each cell in the radar image is sequentially treated as the cell under test, and the local noise characteristics around that cell are analyzed to determine an appropriate detection threshold. The threshold for the j-th range cell in the i-th azimuth direction is given by:

$$\tau_i[j] = \epsilon \hat{\sigma}^2_{i,j}, \tag{B.1}$$

where $\hat{\sigma}^2_{i,j}$ is the estimated noise power for the cell under test at position (i, j) and ϵ is the threshold factor given by

$$\epsilon = N_{\text{train}}(P_{\text{fa}}^{-1/N_{\text{train}}} - 1). \tag{B.2}$$

The noise power is estimated for each cell under test by averaging the neighbouring training cells, where the training cells are not immediately adjacent but have a set of guard cells between the cell under test and the training cells as illustrated in Fig. B.1. Thus, for each azimuth direction $\boldsymbol{e}_i$, the local noise power is estimated by convolving

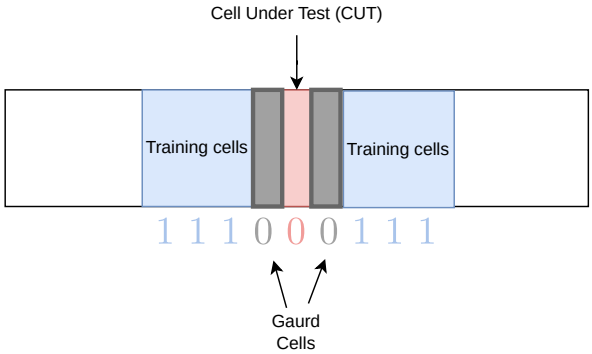


Figure B.1: CFAR training mask $\boldsymbol{m}_{\text{train}}$ for cell under test noise estimate

the training mask with the signal power corresponding to the current azimuth direction

$$\hat{\boldsymbol{\sigma}}_i^2 = \frac{1}{N_{\text{train}}} \text{convolve}(\boldsymbol{m}_{\text{train}}, |\boldsymbol{e}_i|^2), \tag{B.3}$$

where $\hat{\boldsymbol{\sigma}}_i^2 \in \mathbb{R}^{N_{\text{Range}}}$ represents the vector of local noise power estimates for each range cell in the i-th azimuth direction.

Thereafter, for each azimuth direction i, the threshold vector is calculated by element-wise multiplication:

$$\boldsymbol{\tau}_i = \epsilon \hat{\boldsymbol{\sigma}}_i^2, \tag{B.4}$$

and the detections are classified by performing element-wise comparison between the signal power and threshold vectors:

$$b_i[j] = \begin{cases} 0, & |e_i[j]|^2 < \tau_i[j] \\ 1, & |e_i[j]|^2 \geq \tau_i[j] \end{cases}, \tag{B.5}$$

where $\boldsymbol{w}_i \in \{0,1\}^{N_{\text{Range}}}$ represents the binary detection vector for the i-th azimuth direction, and j indexes the range cells. The complete detection matrix is then $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{N_{\text{azimuth}}}] \in \{0,1\}^{N_{\text{Range}} \times N_{\text{azimuth}}}$.

Algorithm 7 CFAR Detection Algorithm

Input: Range-azimuth image $\boldsymbol{E} \in \mathbb{R}^{N_{\text{Range}} \times N_{\text{azimuth}}}$
Parameters: Number of training cells N_{train}, Number of guard cells N_{guard}, false alarm rate P_{fa}
Output: Detection matrix $\boldsymbol{W} \in \{0,1\}^{N_{\text{Range}} \times N_{\text{azimuth}}}$

Compute threshold factor ϵ using (B.2)
for $i = 1$ to N_{azimuth} **do**
    Extract azimuth direction: $\boldsymbol{e}_i \leftarrow \boldsymbol{E}[:, i]$
    Estimate local noise power $\hat{\boldsymbol{\sigma}}_i^2$ using (B.3)
    Compute threshold vector $\boldsymbol{\tau}_i$ using (B.4)
    Classify detections $\boldsymbol{w}_i$ using (B.5)
    $\boldsymbol{W}[:, i] \leftarrow \boldsymbol{w}_i$
end for
return Detection matrix $\boldsymbol{W}$

Table B.1: List of parameters used in CFAR algorithm for range-azimuth radar image

Parameter	**Description**	**Value**
N_{train}	Number of train cells	150
N_{guard}	Number of guard cells	90
P_{fa}	false alarm rate	0.2

Multi-Sensor Coordinate Registration

C

This appendix describes the coordinate transformation process to register LiDAR point clouds to the radar coordinate frame for multi-sensor fusion.

In multi-sensor systems, different sensors operate in separate coordinate frames because of their physically separated locations. To fuse LiDAR and radar data effectively, all measurements must be expressed in a common coordinate system using a rigid body transformation. Techniques used can be found in [50].

In this project, radar is used as the origin, and the LiDAR coordinate frame is therefore registered with the radar coordinate frame.

C.1 Transformation Parameters

The spatial relationship from the LiDAR to the radar is characterized by calibrated parameters, rotation and translation vectors:

$$\mathbf{r}_{\text{L2R}} = [r_x, r_y, r_z]^T \quad \mathbf{t}_{\text{L2R}} = [t_x, t_y, t_z]^T. \tag{C.1}$$

C.2 Rodrigues Formula

The rotation vector $\mathbf{r}_{\text{L2R}}$ is a compact representation where the vector direction indicates the rotation axis and the magnitude indicates the rotation angle. To apply this rotation to points, we convert it to a 3×3 rotation matrix using Rodrigues formula.

First, extract the rotation angle and axis:

$$\|\mathbf{r}_{\text{L2R}}\| = \sqrt{r_x^2 + r_y^2 + r_z^2} \quad \text{(rotation angle in radians)} \tag{C.2}$$

$$\mathbf{u} = \mathbf{r}_{\text{L2R}} / \|\mathbf{r}_{\text{L2R}}\| = [u_x, u_y, u_z]^T \quad \text{(unit rotation axis)} \tag{C.3}$$

Create the skew-symmetric matrix $\mathbf{K}$ from the unit axis:

$$\mathbf{K} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}$$

The rotation matrix is then computed as:

$$\mathbf{R} = \mathbf{I} + \sin(\|\mathbf{r}_{\text{L2R}}\|)\,\mathbf{K} + (1 - \cos(\|\mathbf{r}_{\text{L2R}}\|))\,\mathbf{K}^2$$

C.3 Homogeneous Transformation

Rotation requires matrix multiplication while translation requires vector addition. To combine both operations into a single matrix multiplication, we use homogeneous coordinates—a mathematical technique that represents 3D points as 4D vectors by adding a fourth coordinate with value 1.

This allows us to encode both rotation and translation in a unified 4×4 transformation matrix:

$$\mathbf{T}_{\text{L2R}} = \begin{pmatrix} \mathbf{R} & \mathbf{t}_{\text{L2R}} \\ \mathbf{0}^T & 1 \end{pmatrix} \in \mathbb{R}^{4\times4}$$

The upper-left 3×3 block contains the rotation matrix, the upper-right column contains the translation vector, and the bottom row ensures proper homogeneous coordinate handling.

C.4 Point Cloud Transformation

Transform LiDAR points $\mathbf{P}_{\text{lidar}} \in \mathbb{R}^{M\times3}$ to radar coordinates using homogeneous coordinates:

Step 1: Convert to homogeneous coordinates (denoted by superscript ^{h}):

$$\mathbf{P}^h_{\text{lidar}} = [\mathbf{P}_{\text{lidar}}, \mathbf{1}_M] \in \mathbb{R}^{M\times4} \tag{C.4}$$

where $\mathbf{1}_M$ is a column vector of ones with length M, augmenting each 3D point to 4D.

Step 2: Apply transformation using matrix multiplication:

$$\mathbf{P}^h_{\text{lidar-in-radar}} = \left(\mathbf{T}_{\text{L2R}} \cdot (\mathbf{P}^h_{\text{lidar}})^T\right)^T \tag{C.5}$$

Step 3: Extract the transformed 3D coordinates:

$$\mathbf{P}_{\text{lidar-in-radar}} = \mathbf{P}^h_{\text{lidar-in-radar}}[:, :3] \in \mathbb{R}^{M\times3} \tag{C.6}$$

where $[:, :3]$ notation extracts the first three columns, converting back from homogeneous to Cartesian coordinates.

The result $\mathbf{P}_{\text{lidar-in-radar}}$ represents the LiDAR point cloud expressed in the radar coordinate frame, enabling consistent multi-sensor fusion algorithms to use the synchronised point clouds.