Titre: Title:	A Segmentation and Data Association Scheme for 2D Mapping with Geometric Primitives
Auteur: Author:	Ahmad Zaydan
Date:	2024
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Zaydan, A. (2024). A Segmentation and Data Association Scheme for 2D Mapping with Geometric Primitives [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/59216/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

A Segmentation and Data Association Scheme for 2D Mapping with Geometric Primitives

AHMAD ZAYDAN
Département de génie électrique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie électrique

Août 2024

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

A Segmentation and Data Association Scheme for 2D Mapping with Geometric Primitives

présenté par **Ahmad ZAYDAN**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Roland MALHAMÉ, président
Jérôme LE NY, membre et directeur de recherche
Richard GOURDEAU, membre et codirecteur de recherche
Bowen YI, membre

DEDICATION

I [prefer] a short life with width,
to a narrow one with length
- Ibn Sina

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisors, Professor Jérôme Le Ny and Professor Richard Gourdeau, for their invaluable guidance throughout this project.

I am grateful to Professor Jérôme Le Ny for his meticulous approach and the lessons that have shaped my understanding of the field. I am equally thankful to Professor Richard Gourdeau for his inspired passion and unwavering encouragement. I also thank Professor Roland Malhamé and Professor Bowen Yi for their time and feedback as my jury.

I wish to thank my parents for their endless patience and understanding as I spent countless hours away from home. Your love and encouragement have been my foundation. To my family at large, thank you for your constant support and belief in me.

To my partner for life, meeting you has been the best thing to come out of this entire journey. I am grateful to have you by my side.

To my old friends Tian Ci, Mehdi, and Pranavaa, thank you for enduring my constant rants and complaints.

Last but not least, I am also grateful to the friends I made in the lab—Narimane, Cyrille, and Andréa—who helped me navigate the ups and downs of this project.

Without all of you, this work would have been far less enjoyable. Thank you all for your support.

RÉSUMÉ

Nous introduisons un nouvel algorithme de cartographie LiDAR 2D tirant parti des régularités géométriques dans les environnements structurés. En utilisant des superellipses pour une représentation compacte et polyvalente des caractéristiques structurelles, nous développons un cadre d'extraction de caractéristiques géométriques multi-modèles conscient du modèle. Ce cadre adapte l'architecture EKF duale pour la segmentation et utilise des conditions basées sur la distance de Mahalanobis pour le changement de modèle. Le cadre est construit en explorant et en améliorant les méthodes de segmentation d'un seul scan, l'estimation des paramètres des caractéristiques et l'estimation des caractéristiques multi-scans. Les simulations et les expériences en conditions réelles démontrent des améliorations de la performance de cartographie par rapport à un algorithme LiDAR 2D de pointe, notamment dans des conditions difficiles telles que des vitesses de robot plus élevées, des scans moins fréquents et une exploration limitée.

ABSTRACT

We introduce a novel 2D LiDAR mapping algorithm leveraging geometric regularities in structured environments. Using superellipses for compact and versatile representation of structural features, we develop a model-aware multi-model geometric feature extraction framework. This framework adapts the dual EKF architecture for segmentation and employs Mahalanobis distance-based conditions for model switching. The framework is built by exploring and improving methods for single-scan segmentation, feature parameter estimation, and multi-scan feature estimation. Simulations and real-world experiments demonstrate improvements in mapping performance over a state-of-the-art 2D LiDAR algorithm, particularly under challenging conditions such as higher robot speeds, less frequent scans, and limited exploration.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

LiDAR	Light Detection and Ranging
SLAM	Simultaneous Localization and Mapping
GPS	Global Positioning System
BSP	Binary Space Partitioning
RANSAC	RANdom SAmple Consensus
IMU	Inertial Measurement Unit
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
IEKF	Iterated Extended Kalman Filter
PCA	Principal Component Analysis
NURBS	Non-Uniform Rational B-Spline
SIFT	Scale-Invariant Feature Transform
PHT	Probabilistic Hough Transform
SDF	Signed Distance Function
TSDF	Truncated Signed Distance Function
VDF	Vector Distance Function
ORB	Oriented FAST and Rotated BRIEF
AABB	Axis-Aligned Bounding Boxes
POV	Point Of View
SEM	Standard Error of the Mean
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative

λ_k^q	The x-coordinate of the robot's position in the map referential at time-step k.
ξ_k^q	The y-coordinate of the robot's position in the map referential at time-step k.
φ_k^q	The rotation (orientation) of the robot in the map referential at time-step k.
$\boldsymbol{\Xi}_k$	The robot's pose at time-step k in the map referential.

$\boldsymbol{u}_k$	The known input at time-step k.
$\boldsymbol{\omega}_k$	Represents sensor noise, model uncertainty, or other disturbances at time-step k.
O	The unknown number of obstacles (objects) in the environment where the robot evolves.
θ_c	The curve type for each obstacle's boundary, where $\theta_c \in \Theta = \{\theta_0, \theta_1, \ldots, \theta_C\}$ for C curve types.
$\mathcal{F}_{\theta_c}$	The function defining the boundaries of each obstacle.
$\boldsymbol{x}$	All points that satisfy the boundary function $\mathcal{F}_{\theta_c}$, where $\boldsymbol{x} = [x, y]^T$.
$\boldsymbol{p}$	The parameters of dimension n in the boundary function $\mathcal{F}_{\theta_c}$.
$\boldsymbol{m}_{k,i}$	The LiDAR measurement, where $\boldsymbol{m}_{k,i} = [d_{k,i}, \theta_{k,i}]^T$.
$\tilde{d}_{k,1}, \ldots, \tilde{d}_{k,I}$	The I measurements received by the robot from its LiDAR at each time-step k.
$d_{k,i}$	The true value of the i-th measurement at time-step k.
σ_d^2	The variance of the measurements.
$\boldsymbol{x}_{k,i}$	The world coordinates of the hit-point given by the function F.
L	The transform from LiDAR measurements to the world frame.
$\theta_{k,i}$	The angle of the i-th measurement at time-step k.
$\mathcal{D}_{k,o}$	The set grouping the LiDAR hitpoints that are categorized as belonging to a specific object o at each time-step k.
$\hat{c}_{k,o}$	The estimated implicit function type for object o at time-step k.
$\hat{\boldsymbol{p}}_{k,o}$	The estimated parameters for object o detected by LiDAR at time-step k.
$\hat{\boldsymbol{\Xi}}_k$	The sequence of pose estimates at time-step k.
$\boldsymbol{u}_k$	The sequence of known inputs at each time-step k.
$\tilde{\boldsymbol{m}}_{k,i}$	The sequence of measurements received by the robot from its LiDAR at each time-step k.
$\hat{\mathbf{O}}_k$	The estimated state of objects in the environment - in other words, the map.
$\boldsymbol{X}_k$	The state space in the online SLAM problem at time-step k.
x	The x-coordinate in the super-ellipse equation.
y	The y-coordinate in the super-ellipse equation.
a	The parameter that scales the super-ellipse along the x-axis.
b	The parameter that scales the super-ellipse along the y-axis.
ϵ	The parameter that modifies the shape of the super-ellipse.
x_c	The x-coordinate of the center of the super-ellipse.

y_c	The y-coordinate of the center of the super-ellipse.
ϕ	The parameter that determines the orientation of the super-ellipse.
ψ	The angle used in the rotation matrix for the rotated decentered implicit form of the super-ellipse.
$\mathcal{G}$	The function describing the centered form of the super-ellipse.
$\mathcal{F}_s$	The function describing the rotated decentered implicit form of the super-ellipse.
$\mathbf{p}^s$	The parameter vector for the super-ellipse, which includes x_c, y_c, ϕ, a, b, and ϵ.

CHAPTER 1 INTRODUCTION

1.1 Context

In the realm of robotics, the development of efficient and accurate mapping algorithms is important for tasks such as structural inspection and inventory management in indoor environments. Robots equipped with mapping capabilities are indispensable for assessing the structural integrity of buildings, navigating warehouse spaces, and performing detailed inspections of infrastructural elements. These tasks, often repetitive and structured, lend themselves to methods that exploit geometric regularities in the environment, potentially reducing computational overhead and enhancing the accuracy of the robotic mapping systems.

Traditional approaches to robotic mapping often rely on generic models that do not capitalize on the inherent structured nature of built environments. This research posits that by focusing on specific geometric regularities—such as the predominance of straight lines and uniform shapes in man-made structures—a specialized mapping system can be more effective. The core of this dissertation revolves around the development and implementation of an advanced mapping algorithm that leverages these regularities through the use of superellipses—a geometric form that offers a compact and versatile representation for a wide range of structural features. The main research objective of this work can therefore be defined as:

Develop, Optimize, and Experimentally Validate an Embedded, Real-Time Mapping System Exploiting Geometric Regularities.

This thesis is structured as follows: Chapter 1 introduces the context and objective of the work. Chapter 2 reviews the existing literature, highlighting the current technologies in robotic mapping and detailing the research objectives. In Chapter 3, the problem statement is refined, including the mathematical formulations for leveraging superellipses in environmental modeling. Chapter 4 explores the specifics of single-scan segmentation and parameter estimation. Chapter 5 extends the discussion to multi-scan feature refinement. The final chapter summarizes the findings, discusses the limitations of the current system, and outlines directions for future research.

CHAPTER 2 LITERATURE REVIEW

2.1 Introduction

This introduction provides an overview of the fundamental concepts and challenges associated with SLAM in mobile robotics. The focus is on structured environments where geometric regularities can be exploited for improved mapping and navigation.

To navigate successfully within unfamiliar settings, a mobile robot must not only construct a model of its environment but also accurately estimate its own state within that environment. This dual requirement of environmental mapping and state estimation forms the core of what is collectively known as Simultaneous Localization and Mapping (SLAM). In the realm of mobile robotics, a robot's state often comprises various parameters like its pose, which includes both its position and orientation, as well as other factors such as velocity and sensor parameters.

The importance of the environmental map cannot be understated, as it serves as a spatial representation that includes critical features like landmarks and obstacles. Such maps have dual utility. On one hand, they are indispensable for tasks like path planning. On the other, they provide an avenue for error correction during state estimation processes. This is crucial for the application of error correcting techniques, such as loop closure, which in turn counteract common issues in pose estimation like sensor drift.

While the need for mapping might be mitigated in certain cases by a priori knowledge of landmark locations, this is often not feasible in indoor robotic applications where GPS-based localization is impractical. The inability to use GPS in such situations necessitates the deployment of SLAM techniques, offering a robust alternative to reliance on user-generated maps or dedicated localization infrastructures. This has significantly driven the need to solve the SLAM problem in the field of indoor mobile robotics.

The sensing modalities employed for environmental data acquisition vary widely, encompassing laser scanners, sonar devices, vision systems, compasses, gyroscopes, and GPS units. Each of these sensors comes with its own limitations and inherent measurement noise. For instance, ranging sensors based on light or sound are unable to penetrate through solid barriers such as walls.

These sensors can further be broadly classified into active and passive categories. Active sensors function by emitting energy into the environment and measuring the return signals. Conversely, passive sensors, such as cameras, rely on capturing ambient energy to produce measurements. Cameras themselves can be sub-categorized into monocular and binocular, or stereo, types, each offering distinct advantages and limitations.

In the realm of SLAM technology, cameras and Light Detection and Ranging (LiDAR) sensors are predominantly employed as external sensing elements. LiDAR sensors work by emitting laser beams into the environment. These beams bounce off objects and return to the sensor, where the time taken for each beam to return is measured. By calculating the time difference between the emission and reception of the laser beam, the LiDAR system can determine the distance to the objects. This process is repeated multiple times in a scanning manner where laser beams scan circularily in a slice for 2D LiDAR sensors and with more degrees of freedom for 3D LiDAR sensors. In the presented context, a low-cost 2D LiDAR sensor is particularly well-suited for the task. LiDAR's high spatial resolution and direct acquisition of range data make it an ideal choice for exploiting the regularities in these structured settings. Unlike cameras, which require complex preprocessing to extract depth information, LiDAR provides direct distance measurements [5].

2.2 Solving the Mapping Problem

The architecture of a SLAM system is traditionally divided into two primary components: the front end and the back end. The front end serves as the preliminary interface that abstracts raw sensor data into mapping-friendly models, or features. Given the inherent complexities in practical robotics, where it is often challenging to analytically represent sensor measurements, the front end takes on the role of feature extraction. This pre-processing is intrinsically sensor-specific, as it must adapt to the nature and format of the incoming data stream. Furthermore, the front end can offer initial guesses for subsequent optimization processes.

Data association is another crucial element, linking measurements to specific landmarks within the environment. This is accomplished through short-term data association, which tracks features between consecutive sensor measurements, and long-term data association, commonly referred to as loop closure, which associates new measurements with previously identified landmarks.

While front end systems often intersect with fields such as computer vision and signal processing, back end systems involves a different set of disciplines, including geometry, graph theory, optimization, and probabilistic estimation. Back ends perform inference based on the abstracted data provided by the front end. Moreover, these two have a bi-directional relationship, as the back end supplies feedback for various correction techniques [6].

2.3 Environment Representations

This section examines the various methods for representing the environment in SLAM systems. It begins with spatial-partitioning representations, discussing how objects are segmented into primitives. It then explores sparse representations based on low-dimensional features, detailing how these capture simple environmental characteristics. Following this, boundary representations which can explicitly represent surfaces and volumes are analyzed.

To achieve the research objectives, this literature review specifically focuses on LiDAR-based SLAM systems and their categorization based on environment representations or how the features are modelled. Metric map models are one such representation. They capture the physical layout of an area. These models make it easier to create topological maps, which focus on identifying key locations and how they are connected, and can be integrated in a graph denoting their accessibility through edges. Metric map model features are therefore quantifiable and measurable, such as distances and angles, and define the geometry of the environment.

In contrast, semantic mapping categorizes these locations based on semantic descriptors [6]. Semantic mapping features are typically labels or descriptors that provide context to the physical features within the environment. These may include categories such as 'door', 'window', 'vehicle', and 'pedestrian'. Semantic features go beyond mere geometry to include the meaning of the objects in the environment, thus allowing for more human-like understanding and interaction with the space. Due to the limitations of LiDAR-only systems in generating semantic maps, only metric map models will be discussed.

2.3.1 Spatial-partitioning representations

Spatial-partitioning representations describe objects as combinations of non-intersecting, contiguous primitives. The most common form is simply using identical voxels arranged in a grid. Voxels are three-dimensional pixels that represent data in a grid format within a three-dimensional space. Each voxel contains volumetric information and can be used to model

complex structures by defining the presence or absence of material at specific locations in space, essentially creating a 3D map of an environment. Occupancy grid maps are a variation of grid-mapping that serve as probabilistic variants by incorporating a measure of occupancy likelihood in every Voxel [7].

Some data structures are adapted for more efficiency in sparse environments or faster traversal, including octree structures [8] and Binary Space-Partitioning trees [9]. Octree structures are hierarchical tree-based data structures where each node represents a cubic volume of space, known as an octant, and has up to eight children. This allows for efficient representation of 3D spaces by recursively subdividing the space into smaller regions only where detail is needed, thus saving memory and processing power in sparse areas. Binary Space-Partitioning (BSP) divide space into convex subsets by recursively applying hyperplanes, which can efficiently organize and query spatial data. BSP trees are particularly useful for rendering and collision detection tasks, as they allow for fast lookups and can easily handle dynamic environments.

2.3.2 Low Dimensional Feature-based Sparse Representations

The utilization of sparse features is a prevalent approach in the majority of SLAM techniques for depicting the environment. These reference points are associated with distinctive characteristics, such as wall edges or corners. These points are represented using simple geometric features such as lines, line segments or arcs. The fundamental assumption underlying this representation is the ease of distinguishing these features in large environments.

The LeGO-LOAM SLAM framework [10] first segments raw data points using simple clustering before feature extraction. Segmentation refers to the process of dividing raw data points from a sensor into clusters or groups based on certain criteria. This step is crucial for separating different objects or surfaces in the environment, which helps in the subsequent extraction of meaningful features. Then, planar features are extracted from the ground to obtain z-axis translation, roll and pitch axis rotations. In the subsequent step, the remaining transformations (x and y-axis translations, yaw rotation) are obtained by matching edge features which are extracted according to their gradient (roughness). To achieve even feature extraction from all directions, the range image is horizontally divided into equal sub-images. Points within each row of these sub-images are sorted based on their roughness values, with points having larger roughness than a threshold considered as edge features, and those with roughness smaller than the threshold as planar features. Finally, edge features with the maximum roughness, excluding ground points, are extracted from each row, along with planar features

with the minimum roughness, which are ground points. Newer LOAM methods incorporate neural networks into a generalized iterative closest point algorithm for data association [11].

LIO-SAM [12] is a LiDAR+IMU system that utilizes the same concepts for features but implements some differences such as local scan-matching. Another similar system is HDL-Graph-SLAM [13], employing similarily classical LOAM via preprocessing and scan-matching. A notable difference is that the floor detection uses RANSAC to robustly detect floor planes. HDL-Graph-SLAM optimizes its results using the combined odometry and floor plane data.

Particle filtering with geometric models has been used to enable model-aware segmentation and extraction [14]. Adams et al. established an implicit line model for use in particle filters for feature extraction and tracking [14]. Roumeliotis et al. pioneered the dual EKF (Extended Kalman Filter) architecture to address the trade-off between the sensitivity of a singular filter and its flexibility, i.e., its adaptability to features of completely different parameters within the same scan [15]. Zhang et al. use EKF for segmentation and least squares to fit line/circle features in outdoor structured environments, determining the nature of the feature (line, circle, or clutter) based on EKF prediction error [16]. Zhang et al. propose two methods for feature extraction. The first method uses a single UKF (Unscented Kalman Filter) with a line system model and classified features into lines or arcs (edges or circles) using the UKF residual directly. In the second method, they innovate multi-model extraction with filtering for line and circle features. Each different model is tracked using its UKF (which has a state model based on the geometric model of the feature). The decision to select one model or another is done using the first three measurements of a new feature (after a breakpoint). These measurements are used to calculate the center and radius of an assumed circle. If the radius is large (greater than 0.8m), the next segment is considered to be a line (or cluster) and is tracked using the line UKF [17].

The previously cited SEGMENTS algorithm was tested on line segment features using the Mahalanobis distance as a validation gate for switching between EKFs [15]. The Mahalanobis distance is a metric for evaluating the distance between a point and a distribution [18]. Inside the EKFs it is calculated using the observation and prediction residual and its covariance. The same metric is again investigated by Borges et al. for breakpoint detection in line extraction using line kernels [19]. It is also used by Núñez et al. for segmentation before applying their adaptive curvature function to every segment to extract line segment, curve segment and corner features [20]. A different approach is proposed by Adams et al., where circle and line extraction is performed using smoothing, the Mahalanobis distance is used as a diffusion

coefficient in the smoothing masks [21]. This metric has a proven track record of being used as a number indicating the compliance of a data point to a given geometric model.

2.3.3 Boundary representations

Beyond the use of basic elements like points, clusters or rudimentary features, boundary representations explicitly depict surfaces and volumes [6]. Boundary representations characterize objects by their surface boundaries. These can be simplified, plane-based models, or more comprehensive representations based on curves, and implicit surface models. Boundary representations are categorized into open and closed surface models. Open surface models depict shapes without enclosing volumes, suitable for objects with edges and openings. Conversely, closed surface models represent fully enclosed volumes, defining solid objects with both an interior and exterior.

Open surface models

B-splines are parametric curves defined by control points and a knot vector. These curves can represent a wide range of shapes, combining both lines and curves through the manipulation of their control points and knot vectors.

B-spline SLAM (BS-SLAM) [1] [22] is a framework that leverages B-splines to represent non-linear boundaries in environments. The frameowrk is illustrated in Figure 2.1. The control polygon vertices, serving as control points, are used to represent the state of the environment. The curve fitting process relies on a system of equations to express data points in terms of the B-spline curve. A least squares solution is computed using the pseudoinverse matrix of the B-spline. Obtaining splines in the BS-SLAM framework is a three-stage process. Initially, the data stream is segmented into pieces, resulting in primary segmentation objects. Subsequently, secondary segmentation takes place, where the relative positions of consecutive data points are analyzed. This step allows for the detection of close points, corners, and non-segment points, leading to the identification of secondary segmentation features. Finally, each secondary segmentation object is fitted with a cubic spline, forming the basis for further processing.The next aspect involves spline association, data association between existing map splines and new ones. It is done by measuring distances between control points and simplifying the map by selecting splines in proximity to the robot's position. The matching of observed splines to map splines is based on distance. The state model in BS-SLAM encompasses both the robot and cubic splines features. The state vector encapsulates the control polygon vertices, enabling the expression of positions and orientations in a global

reference system. The observation model in BS-SLAM is responsible for calculating expected measurements based on laser beam interactions with map splines.

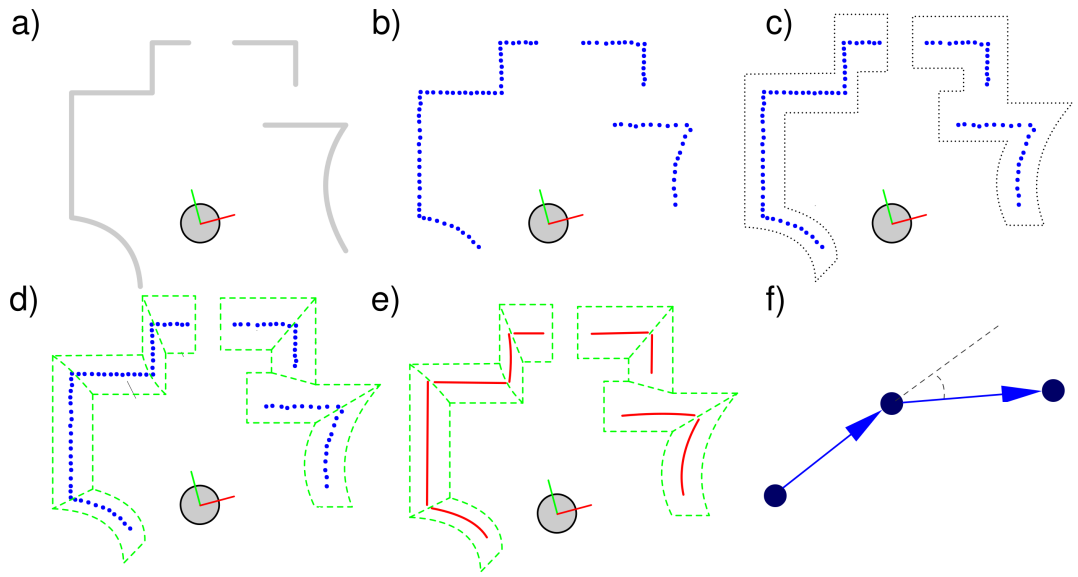


Figure 2.1 BS-SLAM framework [1], (a) The robot and the environment setup. (b) Acquisition of a new data set. (c) Initial segmentation of the data. (d) Secondary segmentation showing detected splines and fitted data points. (e) Final splines from the secondary segmentation. (f) Relative positions of data points.

B-Spline surface maps have been created using mutually perpendicular LiDARs [23]. The process begins with extracting line segments from sensor scan data, grouping data based on distance and angle criteria, and selecting one line segment from each data group. Line segment parameters are expressed by minimizing the distance between raw data points and the expected line segment. An Iterative Extended Kalman Filter (IEKF) incorporates principal component analysis (PCA) for data association.

Raw data is segmented into categories based on continuous rotation and continuous translation thresholds. Robot motion leads to overlapping B-spline surfaces, requiring a solution for merging them. Overlapping B-spline surfaces are handled by iteratively projecting boundary points onto each other until termination conditions are met.

A more modern implementation of 2D B-spline SLAM [24] integrates a novel fitting scheme and probabilistic occupancy. Unlike traditional spline regression methods that use ordinary least squares and have a computational complexity of $O(n^3)$, this approach introduces a re-

cursive method to reduce complexity to $O(n)$. Occupancy probabilities are updated through Bayesian inference and log odds, with control point adjustments based on new measurements. Localization focuses on scan-to-map alignment by establishing a cost function for pose estimation, approximated using Taylor expansion to yield a linear least-squares solution.

NURBS (Non-uniform rational basis spline) SLAM [25] utilizes a more generalized form of B-splines and Bézier curves, which incorporate rational control points, a knot vector, and an order to define their shape. In this approach, point cloud data is initially mapped onto a grid graph with edge weights determined by point-to-point distances. A random walk identifies the stationary distribution, iteratively pruning low-weight edges. Greedy entropy maximization segregates the graph into distinct groups based on node weights. Surface fitting employs a salient point-based NURBS technique. The algorithm incorporates node selection at a specific sampling rate, sub-graph creation, and surface fitting within each group.

Systems that use planes as landmarks were previously envisioned to be used in hand-held and wearable cameras [26]. Features are extracted using SIFT and RANSAC is used to estimate the homography with planar database objects to semantically segment and classify the features. Newer systems, such as the structure-aware SLAM using quadrics and planes [27] use other techniques for feature extraction and grouping. This algorithm is built upon ORB-SLAM2, data association across frames is facilitated by matching features in a coarse-to-fine pyramid within a local window surrounding the previous observation. To mitigate the impact of partial occlusions, the system employs Huber kernels to enhance robustness against substantial errors. Huber kernels are integrated into systems as part of the optimization process when adjusting the trajectory and map estimates. Specifically, they are used in the cost function that the SLAM algorithm aims to minimize. The cost function measures the discrepancy between the predicted sensor measurements and the actual measurements. In a standard least squares optimization, all measurement errors would be treated equally, but this can be problematic when outliers are present. Huber kernels modify the cost function such that for small residuals (errors), the cost is quadratic, but for large residuals, the cost becomes linear. The threshold between small and large residuals is a tunable parameter in the Huber kernel.

Ravankar et al. [28] lay out different algorithms for line segment extraction. They employ polar and Cartesian coordinate systems to define a line, and specify its orientation for segment determination. Endpoints of the line segment are scalar values which can be generalized for multiple endpoints on the same line. The study utilizes a modified Progressive

Probabilistic Hough Transform (PHT) algorithm for line extraction by incorporating new random voting points while eliminating previously detected ones. The PHT algorithm is an advanced version of the Hough Transform. The basic idea of the Hough Transform is to transform points from the image space to a parameter space, where a set of points that form a line in the image space corresponds to a peak in the parameter space. The PHT improves upon the classic Hough Transform by progressively adding points to the parameter space and using a probabilistic approach to determine the existence of a line. A weighted line fitting model minimizes a non-linear optimization problem to derive line parameters and covariances. Line segments are sorted and merged based on similarity, with a chi-squared χ^2 test used to ascertain whether they lie on the same line. If so, the lines are merged and new endpoints are derived by projecting the existing ones onto the combined line.

Closed surface models

Signed Distance Functions (SDFs) are mathematical representations used in computer graphics and geometry to describe the distance between a point in space and the closest surface of an object. What makes SDFs particularly valuable is that they can represent not only distances but also the direction to the nearest surface. While not inherently part of their definition, SDFs are frequently employed to model closed surfaces; when evaluated at specific points in space, a negative value indicates the point is inside a solid object, zero signifies the point is on the surface, and a positive value indicates the point is in open space. Figure 2.2 provides a visualization of a signed distance function.

Consider Ω as a specific subset within a metric space X, characterized by a metric d, and let $\partial\Omega$ denote the boundary of Ω. For any point x in X, the minimal distance to the boundary $\partial\Omega$ is defined by

$$\delta(x, \partial\Omega) = \inf\{d(x, y) : y \in \partial\Omega\}, \tag{2.1}$$

where inf denotes the infimum, capturing the closest distance from x to any point on $\partial\Omega$.

The SDF for a point x relative to Ω is determined as follows:

$$g(x) = \begin{cases} \delta(x, \partial\Omega) & \text{if } x \notin \Omega, \\ -\delta(x, \partial\Omega) & \text{if } x \in \Omega, \end{cases} \tag{2.2}$$

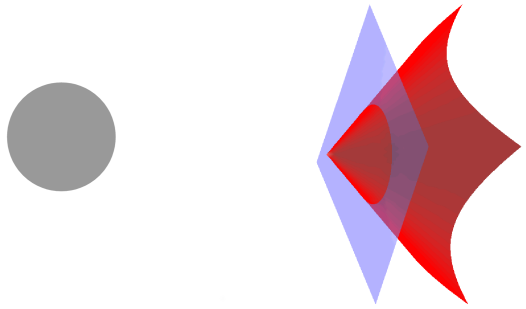


Figure 2.2 Signed distance function visualization [2]

2D-SDF-SLAM [29] introduces an approach centered on utilizing SDFs for map representation, coupled with map gradient-based registration. The map consists of a 2D grid with signed distances to objects. Positive values within cells represent free space, while negative values signify occupied regions. Instead of traditional raytracing, regression is employed to update the signed distance values of cells along the ray's path, with orthogonal Deming regression used to find a line that best fits scan endpoints in a square. Unlike ordinary least squares regression, which minimizes the vertical distance from the data points to the regression line, orthogonal Deming regression minimizes the orthogonal (perpendicular) distances. This makes it more suitable for situations where both variables have measurement error. VoxGraph [7] utilizes SDF properties and formulates the problem as graph optimization while creating independent conditions between sets of localized maps. This approach is sufficiently resource-efficient to recalibrate the entire system whenever a fresh localized map is incorporated.

The front end of the system described by Millane et al. [30] computes occupancy grid submaps which are then transformed into SDFs. Keypoint detection within this system is grounded on identifying high curvature points in the SDF using the Determinant of Hessian method. These keypoints are categorized as maxima, minima, or saddles based on surface topology. Descriptors are compact representations of the features of an object or a point of interest within an image or a map. They are designed to capture essential information about the feature in a way that is easy to compare with other features. The descriptors for the extracted keypoints are assembled by analyzing gradient orientations and magnitudes, resulting in a descriptor that incorporates weighted measurements for matching features.

In contrast to SDFs, Truncated Signed Distance Functions (TSDFs) limit the distance values to a fixed range around the surface. This truncation is useful for fusing multiple depth maps, as it allows for easier handling of noise and occlusions in the data [31]. In range-based TSDF SLAM systems [32] [33] [34] TSDFs are calculated only near the surface of objects and are truncated beyond a specified distance τ.

Daun et al. [32] [33] offer an approach wherein a 2D scene is divided into a uniform grid. Each cell in this grid holds the current TSDF estimate along with a confidence weight. The study introduces a technique for updating TSDF values called the Approximate Euclidean Distance Update. This technique is especially useful for offsetting distance biases when viewing angles are not orthogonal to surfaces. To estimate geometric distances along beams from observation points, scan normals are employed.

ESLAM [35] leverages implicit TSDFs for their fast convergence for higher-quality reconstructions. The method also incorporates a variety of loss functions to accelerate convergence. Per-point losses and rendering losses are implemented, alongside free-space losses that encourage the TSDF to maintain a value of one. In terms of mapping and tracking, pixels are randomly selected from a number of frames for mapping optimization. Another method for mobile robot position estimation [36] utilizes a spline-based approximation to represent the robot's surrounding environment through a vector distance function (VDF). The VDF is an extension of the SDF in 3D. The constituents of the VDF are estimated using cubic B-spline fitting through least-squares.

Alternative approaches represent geometric features without explicitly measuring distance to some object using implicit functions. Implicit functions define the points that lie exactly on the surface of the shape. The only information about the space around the function is whether it is inside or outside the surface. Vandorpe et al. propose a method that represents the environment through lines characterized by parameters p and θ, as well as circles described by their radius and center coordinates [37]. Error quantification is performed using covariance matrices. The line extraction process relies on three conditions: the distance between consecutive points on the line must be below a maximum value; the distance from any new point to the line being extracted must also be below a maximum; and the angular deviation between new and existing points should be within a predefined limit. Lines are calculated using linear regression, and their endpoints are defined by the first and last points added to the line. Clusters of points failing these criteria are represented by circles, whose

parameters are computed by the root mean square of the radius.

Another algorithm focusing on line segment and circle features [16] uses the previously mentioned EKF to identify an edge when the discrepancy between the measured and predicted range exceeds a predefined threshold. A mathematical framework characterized by a second-order differential equation is used to model planar surfaces. The system model employs state-space representation to forecast the next sensor range value. Criteria for identifying new edges or significant discontinuities are established a validation gate on the EKF's prediction. For circles, the algorithm uses the modified Gauss–Newton method, a least-squares optimization technique, to estimate the center and radius. The process iterates until it meets a predefined error threshold and a specified number of iterations.

Pascoal et al. employs superquadrics to represent environments [38]. Superquadrics are a family of geometric shapes described by algebraic equations that generalize quadrics, including ellipsoids, hyperboloids, and cylinders, through the introduction of shape parameters. These parameters allow for the flexible modeling of a wide variety of forms, ranging from sharp-edged to smoothly curved surfaces. The core of superquadrics lies in their implicit and parametric equations, enabling the representation of complex shapes with precision.

The implicit equation of a basic superquadric is given by

$$|x|^r + |y|^s + |z|^t = 1, \tag{2.3}$$

where r, s, and t are positive real numbers that determine the shape's characteristics. The versatility of superquadrics is evident in their ability to morph between different geometrical structures, such as octahedrons, spheres, and cubes, by adjusting these parameters. The variables A, B, and C are added to serve as scaling factors along the x, y, and z axes, respectively,

$$\left|\frac{x}{A}\right|^r + \left|\frac{y}{B}\right|^s + \left|\frac{z}{C}\right|^t = 1, \tag{2.4}$$

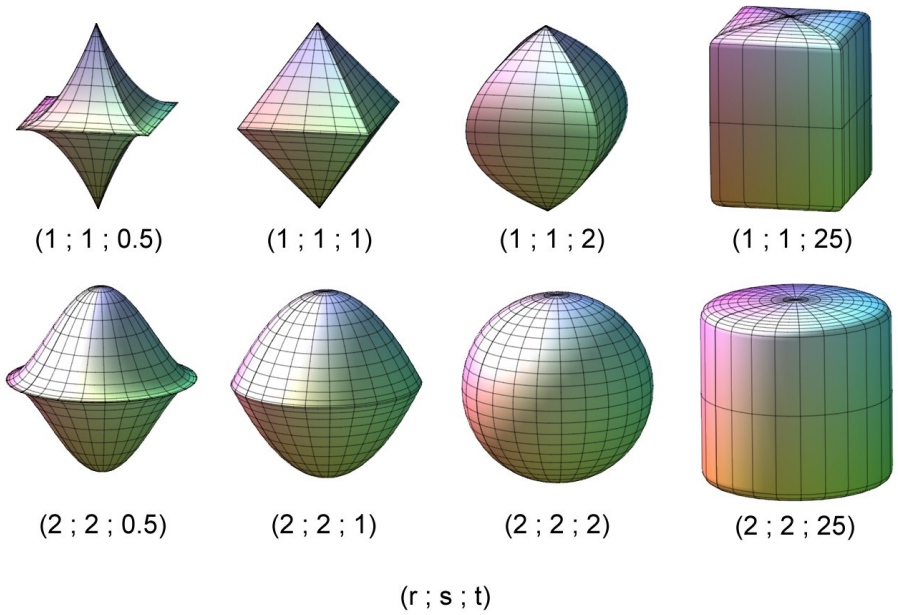


Figure 2.3 Various super-quadratics to show the effects of the parameters in the equation [3]

Figure 2.3 shows the various shapes that can be created by varying the three parameters. This allows for an even more versatile representation of volumes in the environment. Previous similar methods decouple segmentation from shape extraction, which introduces inconsistencies. To address this, an optimization-driven objective function is used that balances factors like superquadrics' volume, distance inaccuracies, and visibility. During this optimization, volume and distance computations hold particular importance, with volume incurring a penalty in the cost function. Self-occlusion scenarios are managed by incorporating dot products between normal vectors of the superquadric surface and unit vectors pointing from the sensor to the range point. This technique is particularly effective when an extended superquadric representation is in play, as it aids in the calculation of the surface normal. The algorithm iteratively fits superquadrics to a set of 2D range-points, discarding those that are successfully classified. It uses midpoints between angularly ordered scanned points for fitting, enhancing the accuracy of segmentation especially in complex scenarios like detecting walls in tunnels.

Zhao et al. [39] introduce an energy term that accounts for observation noise by incorporating a similar covariance matrix. However, because the covariance of implicit functions is not directly obtainable, the observation noise is assumed to be Gaussian. The authors further suggest an enhanced objective function tailored for features with closed shapes to aid in the optimization process. Their method is exemplified using ellipse and line features.

2.4 Clustering and Data Association

Clustering and data association are pivotal elements that bridge the front and back end. Clustering algorithms categorize incoming point clouds into meaningful clusters for later feature extraction. Data association deals with matching these clustered landmarks over consecutive time frames. This process is crucial for tracking the features and enhancing the state estimation.

The method proposed by Zhao et al. [39] operates under the assumptions of known feature count and sparse featured placement, allowing data over multiple scans to be easily grouped into a known number of clusters. The data is then associated using thresholded evaluation of the implicit loss functions defined for lines and ellipses. Another example is the work by Meng et al. [40], where ORB features are segmented by an object detector with bounding boxes in RGB images. These features are subsequently associated during the bundle adjustment pose tracking phase. Although there is no enhancement of existing objects in this method, objects with low confidence are reinitialized.

In the work by Vaskevicius et al. on quadric extraction from noisy point clouds [41], a preprocessing phase aims to identify suitable seeds for new segments through a series of steps. Initially, abrupt changes in range measurements are monitored to detect noisy measurements and potential boundaries. Surface orientation changes are then assessed using a small sliding window, where changes are calculated using the dot-product between consecutive normals. Finally, a wave propagation algorithm identifies potential seeds, which are sorted by distance and serve as initial points for region-growing. During the region-growing phase, seeds are selected from a preference list, and a pixel's neighborhood is used. A point is added to a region based on two criteria: 1) the Mean Square Error (MSE) must not exceed a predefined threshold, and 2) the distance of the point's projection on the fitted surface must also fall within a set threshold. For model refinement, a Maximum Likelihood approach is employed, formulating the task as a nonlinear minimization of negative log-likelihood while incorporating inequality constraints. This process is effective but has a high computational cost due to its nonlinearity and constraints.

Lu et al. [42], utilizes a Lucas-Kanade sliding window least-squares approach for feature tracking. The Lucas-Kanade method's efficacy in visual tracking, based on the premise that pixel intensities of an object remain constant between consecutive frames, mirrors the logic applicable to LiDAR feature tracking, where the physical properties and spatial relationships

of environmental features (e.g., edges, planes) are expected to exhibit minimal change over short intervals. This could allows for the reassociation of LiDAR data points with pre-existing cluster features and the creation of new clusters in a sliding-window fashion. For key points, the algorithm works by reassociating them with existing cluster features. A similar approach is adopted for vanishing points, where reassociation is performed and new cluster features are created when necessary. For lines features, the procedure resembles that of key points but is governed by specifically defined parallax criteria. Furthermore, lines features and cluster features are associated to plane features based on a consensus score.

In the approach outlined by Fossel et al., Gauss-Newton minimization is employed for scan registration, using the map gradient to iteratively align poses [29]. This method contrasts with more recent techniques where mapping and tracking are performed by randomly selecting pixels from multiple frames and optimizing the alignment using the ADAM optimizer [35]. Millane et al. employ place recognition by matching pairs of submaps through nearest-neighbor lookup and using RANSAC to identify inlier correspondences [30]. An alternative approach by Daun et al. [33] leverages a multiresolution scan matching strategy, using two TSDF grids with the same truncation distance but different resolutions. The optimization process in this method, as well as in another study by the same authors, is solved using the Levenberg-Marquardt method with gradients computed through Automatic Differentiation [32]. Finally, Reijgwart et al. detect overlapping submaps using Axis Aligned Bounding Boxes (AABBs), calculated prior to global optimization [7].

Low-dimensional feature tracking can be done in two ways (a) tracking features using their parameters (scan matching using features) or (b) associating new measurements to existing features. The former is common when no odometry information is available while the latter is common when odometry information is provided [43]. The low-dimentional feature extraction algorithms discussed previously do not incorporate any kind of feature tracking, but simple schemes for feature combination and elimination in post-processing are employed. Traditional EKF-based SLAM algorithms associate measurements to previously seen features using the Mahalanobis distance discussed above [44]. Zhang et al. make use of it as a gating criterion for their data association approach in order to limit the number of features considered for association, under its alternative name NIS [45] (Normalized Innovation Squared) [46]. Vázquez-Martín et al. adopt a corner-only representation for a traditional EKF-based SLAM but present an adaptive observation covariance based on the landmark distance [47]. The segmentation algorithm used is the adaptive segmentation algorithm presented by Borges et al., it does not account for the features model [19]. Liu et al. suggest replacing filtering with

optimization for robustness in no odometry feature-based SLAM with curve features [43]. Segmentation is done using naive range and angle thresholds. Zhao et al. present three closely related implementations. The first presents a feature-based SLAM with conics [48]. The second extends that framework to more general implicit functions, proposing improvements to the energy term in feature-based SLAM [49]. The third implements that algorithm with implicit features represented by Fourier series [50]. In all three, segmentation is done via naive clustering as the features are assumed to be sparsely placed in the environment, the odometry is combined with feature tracking for localisation. Tackling the problem of outdoors EKF-SLAM in semi-structured environments, Zhang et al. present an implementation that uses a combination of their previously cited switching multi-model feature extraction with the aforementioned NIS-based feature association [51]. As cited previously, because of the way the switching works, the algorithm is limited to lines and circles [17].

2.5 Contribution

This thesis aims to tackle two perceived shortcomings in the literature. Firstly, in many cases where geometric feature-based SLAM is used, the structure of the environment is not exploited for segmentation [47] [52] [48] [49] [50]. Secondly, the multi-model segmentation algorithms proposed previously use tricks to distinguish between two specific geometric models, they cannot generalize to any model or more than two models [51] [17] [20] [21] [16]. Thus, a model-aware multi-model geometric feature extraction framework is proposed. In the proposed approach, for every implicit model to be used, an adaptation of the dual-EKF architecture proposed in the SEGMENTS algorithm is instantiated. This is done to address the trade-off between the sensitivity of a filter and its flexibility [15]. Choosing between models is done using an FSM (Finite State Machine), with conditions that depend on the Mahalanobis distance (also called NIS) [18] [45]. This is an obvious choice as the metric has a proven track record of being used as a number indicating the compliance of a data point to a given geometric model. The proposed framework therefore builds on the literature to effectively leverage prior assumptions of an arbitrarily varied structured environment. To support these claims, the framework is configured with two feature types: line segments and super-ellipses, then tested for two main metrics. The first metric is the segmentation accuracy in challenging environments. The second metric is the completeness of the generated map. To sum this up, the following Sub Objectives (SO) have been established:

SO1: Develop a framework that can generalize to any implicit geometric model and handle more than two models simultaneously.

SO2: Configure and test the framework with line segments and super-ellipses as feature types.

SO3: Evaluate the framework's single-scan segmentation accuracy in challenging environments.

SO4: Compare the completeness of the generated map over multiple scans with a state-of-the-art 2D LiDAR algorithm.

CHAPTER 3 PROBLEM STATEMENT AND PROPOSED ARCHITECTURE

3.1 Introduction

In this chapter we formulate a mapping problem. The proposed approach for solving the problem estimates the parameters of curves fitting these clustered points, represented using implicit functions. It uses those features for aligning scans, a process also known as scan registration, using data association techniques.

3.2 Formulation

It's essential to note some assumptions made during the research. Firstly, the environment is assumed to be structured, i.e. all objects and structures can be approximately represented by the chosen implicit models. We choose to focus on environments containing walls as well as convex symmetrical objects. The walls can be represented using line segments, and the convex symmetrical objects can be represented using super-ellipses (defined in Section 4.2.1.) Secondly, objects in this environment are separated by a minimum distance, a critical consideration for solving the segmentation or clustering problem. Additionally, the map is static and does not change during the mapping time-frame. Lastly, a Gaussian sensor noise model is used for the LiDAR measurement model but only for the distance component of the measurements, keeping the angle component noise-free.

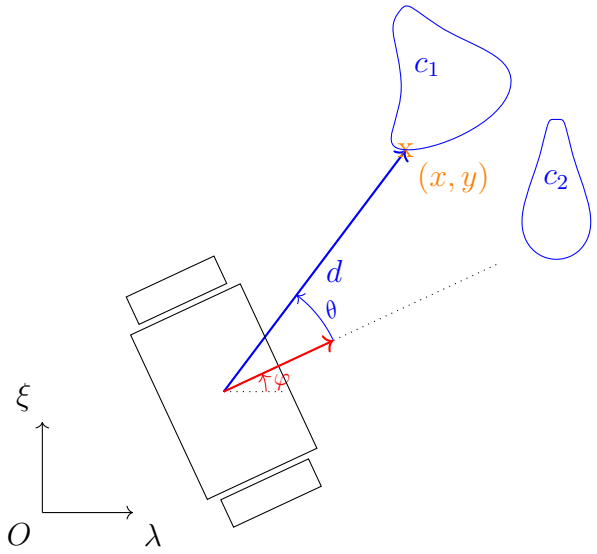


Figure 3.1 The robot in the world frame with a laser beam hitting one of two objects.

The robot is shown in Figure 3.1 moving in an environment with an unknown number of objects. The robot's pose in 2D at time-step k in a fixed frame of reference, termed the world frame, is denoted $\boldsymbol{\Xi}_k = [\lambda_k, \xi_k, \varphi_k]^T$. Here, λ_k signifies the lateral position of the robot within the world frame, typically aligned with the primary direction of movement or the forward-facing boundary of the environment. ξ_k denotes the longitudinal position of the robot within the world frame, aligned perpendicular to λ_k and often representing the depth or progression through the environment. φ_k represents the orientation or angular position of the robot within the world frame, indicating the direction the robot is facing relative to the direction of λ_k. The pose evolves as:

$$\boldsymbol{\Xi}_{k+1} = f(\boldsymbol{\Xi}_k, \boldsymbol{u}_k, \boldsymbol{\omega}_k)$$

where $\boldsymbol{u}_k$ is the known input at time-step k, and $\boldsymbol{\omega}_k$ represents sensor noise, model uncertainty —particularly those arising from discretization —at time-step k. At each time-step k, the robot receives a scan from its LiDAR consisting of $I(k)$ measurements. As the robot moves through the environment, the number of measurements changes depending on the time-step k at which it was taken:

$$\boldsymbol{m}_{k,i} = [d_{k,i}, \theta_{k,i}]^T. \tag{3.1}$$

These measurements are constituted of $I(k)$ range measurements $\tilde{d}_{k,i}$ for $i \in \{1, \ldots, I(k)\}$ with true values $d_{k,i}$ and variances σ_d^2, and $I(k)$ angle measurements $\theta_{k,i}$ for $i \in \{1, \ldots, I(k)\}$.

The world frame coordinates of the hit-point (visible in orange in Figure 3.1) are given by:

$$\boldsymbol{x}_{k,i} = L(\boldsymbol{\Xi}_k, \boldsymbol{m}_{k,i}).$$

with L the transform from LiDAR measurements to the world frame and $\boldsymbol{\Xi}_k$ the robot's pose. This leads to the following hit-point equations:

$$L(\boldsymbol{\Xi}_k, \boldsymbol{m}_{k,i}) = \boldsymbol{x}_{k,i} = \begin{bmatrix} x_k + d_{k,i}\cos(\varphi_k + \theta_{k,i}) \\ y_k + d_{k,i}\sin(\varphi_k + \theta_{k,i}) \end{bmatrix} \tag{3.2}$$

We assume that each object's boundary can be represented by a parameterized curve $c \in C$, with C a finite set. For example, lines, ellipses, etc. A curve of type c can be represented implicitly as the 1-level set of a function $\mathcal{F}_c : \mathbb{R}^2 \to \mathbb{R}$. Finally, an object's boundary curve of type c has parameters $\mathbf{p}$:

$$\mathbf{p} \in \mathbb{R}^{n_c} : O_c(p) = \Big\{\mathbf{x} \in \mathbb{R}^2 | \mathcal{F}_c(\mathbf{x}, \mathbf{p}) = 1\Big\}.$$

The objective of the online mapping system is to estimate the map of the environment in real-time, as opposed to offline mapping where all data is processed a posteriori. Given the robot's motion model and sensor measurements as described above, the online mapping algorithm must provide an incremental update to the map m at each time-step k. Therefore, at each time-step k, the goal is to generate:

- A set of objects $\hat{\boldsymbol{O}}_k$;
- A sequence of sets $\{\mathcal{D}_{k,o}\}_{k\geq 0, o\in\hat{\boldsymbol{O}}_k}$, where

  $$\mathcal{D}_{k,o} = \{\tilde{\boldsymbol{x}}_{k,1}, \tilde{\boldsymbol{x}}_{k,2}, \ldots, \tilde{\boldsymbol{x}}_{k,N_{k,o}}\}$$

  groups the $N_{k,o}$ LiDAR hitpoints that are categorized as belonging to a specific object o at time-step k;
- Curve type estimates for objects: $\{\hat{c}_{k,o}\}_{k\geq 0, o\in\hat{\boldsymbol{O}}_k}$;
- Parameter estimates for objects: $\{\hat{\boldsymbol{p}}_{k,o} : \mathbb{R}^{n_{c_{k,o}}}\}_{k\geq 0, o\in\hat{\boldsymbol{O}}_k}$;

With I the number of measurements taken at each time-step, given sequences $\{\boldsymbol{u}_k\}$ and $\{\tilde{\boldsymbol{m}}_{k,i}\}_{i=1,\ldots,I}$.

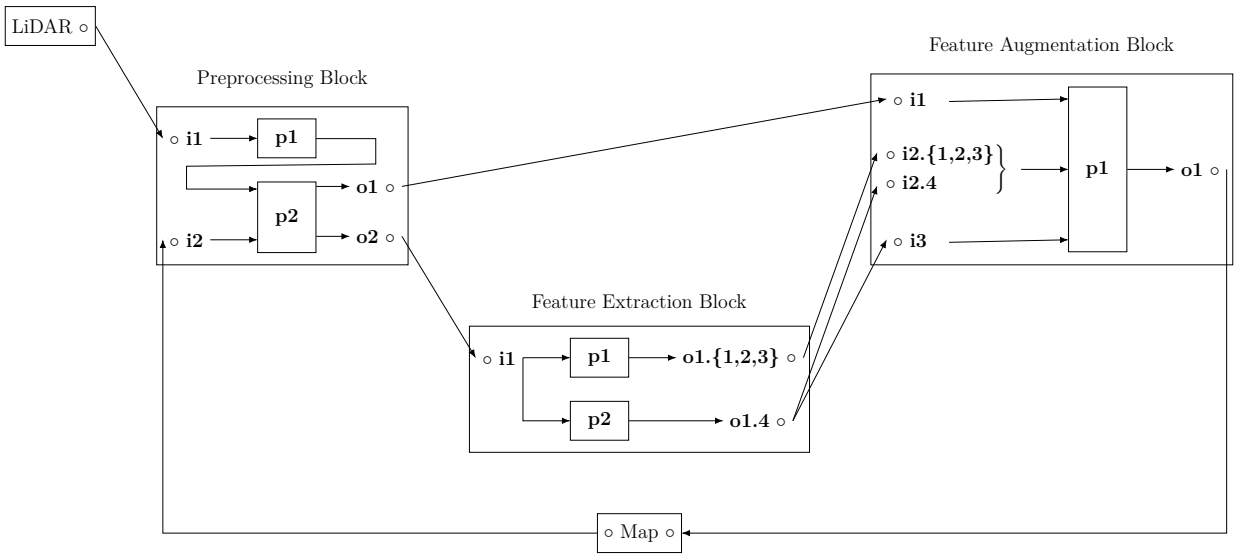
3.3 Proposed Architecture

The architecture of the system is conceived as an assembly of functional blocks. This structure facilitates data flow and allows for each block to be fine-tuned as needed. In the design of these blocks, the primary objective is to segregate all the tasks necessary for the front-end of an online mapping system. Each block is made to carry out a function completely distinct from the others.

These blocks are coordinated together using a system of inputs and outputs, involving various data structures, and the data structures themselves are designed to be encapsulated into each other. This encapsulation ensures that they can be reused across different blocks without requiring modification.

The main functions that are implemented within this architecture are represented in Figure 3.2, they include the pre-processing of raw data (block described in Section 5.4.1 and algorithm in Section 5.2) the extraction of features (block described in Section 5.4.1 and algorithm in Section 4.4.3), and the updating of these features (described in Sections 5.4.1 and 5.3). Key data structures are presented in the their respective blocks.

Figure 3.2 Global Architecture



CHAPTER 4 SINGLE-SCAN SEGMENTATION AND PARAMETER ESTIMATION

4.1 Introduction

This chapter outlines the explored solutions for the fitting and segmentation of scan points. As discussed previously, segmentation in the context of mapping is the process of grouping raw sensor data into different clusters to extract map features. In the case of a single LiDAR scan, the segmentation stage must separate the measurements into groups of measurements obtained from the same obstacle. In order to create the implicit function based map, the segmented clusters must be estimated as a given curve $\hat{c} \in C$. The implicit function $\mathcal{F}_c$ associated with the curve type must also have its parameters estimated using the points in the given cluster. As specified in Section 3.2, one of the initial assumptions is that all objects and structures can be approximately represented by the chosen implicit models, line segments and super-ellipses.

The parameter estimation method is first investigated for the main chosen implicit functions assuming pre-segmented clusters and partial visibility. Superellipse fitting performance is quantified using a series of Monte-Carlo experiments. For segmentation, a method for function-aware segmentation is investigated and an algorithm for function-aware simultaneous segmentation and parameter estimation is presented as a finite state machine. The performance of the segmentation algorithm is then evaluated for various user-settable parameters and an example of tuning for a given environment type is presented.

4.2 Super-Ellipse Parameter Estimation

4.2.1 Super-Ellipse Parameter Estimation

Super-ellipses are higher-order primitives originally defined by Gabriel Lamé with the equation $|x/a|^n + |y/b|^n = 1$. The influence of n is visually represented in Figure 4.1.

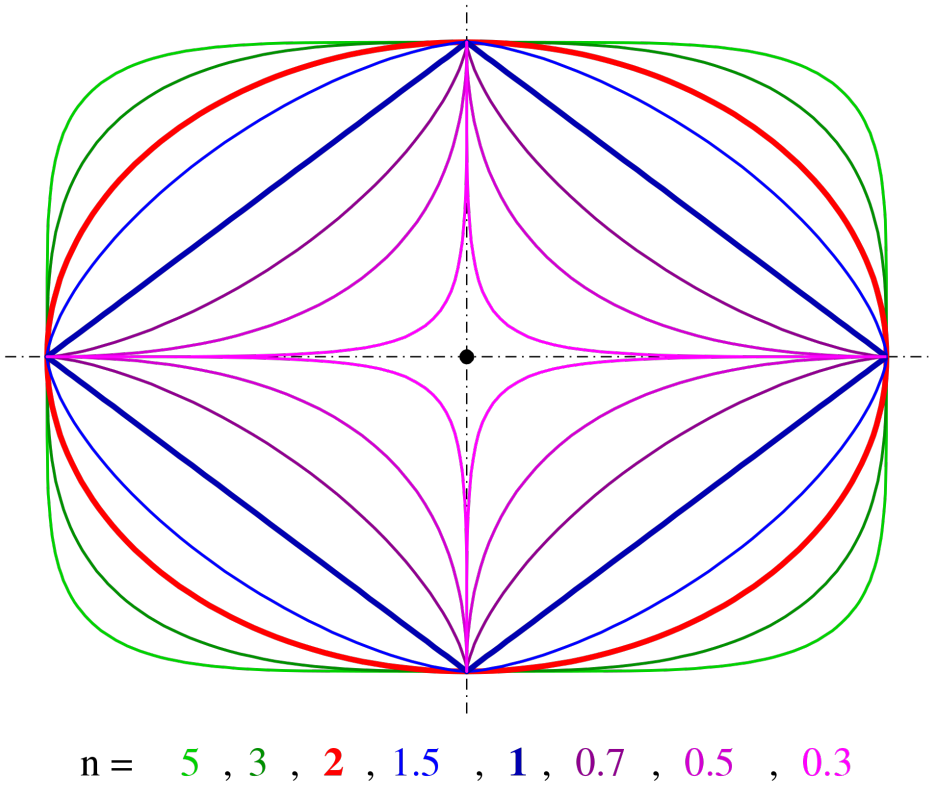


Figure 4.1 Influence of n on Lamé curve [4].

In this work, n is re-parameterized as $\frac{2}{\epsilon}$. These curves are described in their centered form by the equation

$$\left(\frac{x}{a}\right)^{\frac{2}{\epsilon}} + \left(\frac{y}{b}\right)^{\frac{2}{\epsilon}} = 1.$$

To offset the shape from the center and scale it vertically or horizontally, the equation becomes

$$\left(\frac{x - x_c}{a}\right)^{\frac{2}{\epsilon}} + \left(\frac{y - y_c}{b}\right)^{\frac{2}{\epsilon}} = 1;$$

where parameters x_c, y_c control the center of the super-ellipse and a, b scale the super-ellipse along the horizontal and vertical axes. To account for orientation, a rotation matrix is introduced, leading to the final rotated decentered implicit form

$$\begin{aligned} \mathcal{F}_s\left(\boldsymbol{x}, \boldsymbol{p}^s\right) = & \left(\frac{(x_{k,i} - x_c)\cos\phi + (y_{k,i} - y_c)\sin\phi}{a}\right)^{2/\epsilon} \\ & + \left(\frac{(x_{k,i} - x_c)\sin\phi - (y_{k,i} - y_c)\cos\phi}{b}\right)^{2/\epsilon}; \end{aligned} \tag{4.1}$$

with ϕ determines its orientation. The contour is represented by

$$\{\boldsymbol{x} : \mathcal{F}_s(\boldsymbol{x}, \boldsymbol{p}^s) = 1\}.$$

The parameter vector is evidently

$$\boldsymbol{p}^s = \begin{bmatrix} x_c & y_c & \phi & a & b & \epsilon \end{bmatrix}^T,$$

For $\epsilon > 2$, the super-ellipse becomes non-convex; in this work, ϵ is restricted to the range $]0,\ 2[$. This super-ellipse construct extends the ellipse's domain to include a plethora of shapes, such as rectangular and diamond-like shapes, via the addition of the parameter ϵ. The difficulty of parameter estimation increases considerably with this addition, making a closed-form solution challenging to derive [53]. The use of super-ellipse parameter estimation is prevalent in many fields. In computer vision, the data is often complete, contrasting with the field of robotics where the data is often partial; occluded robot POV (Point Of View) portrayed in Figure 4.2.

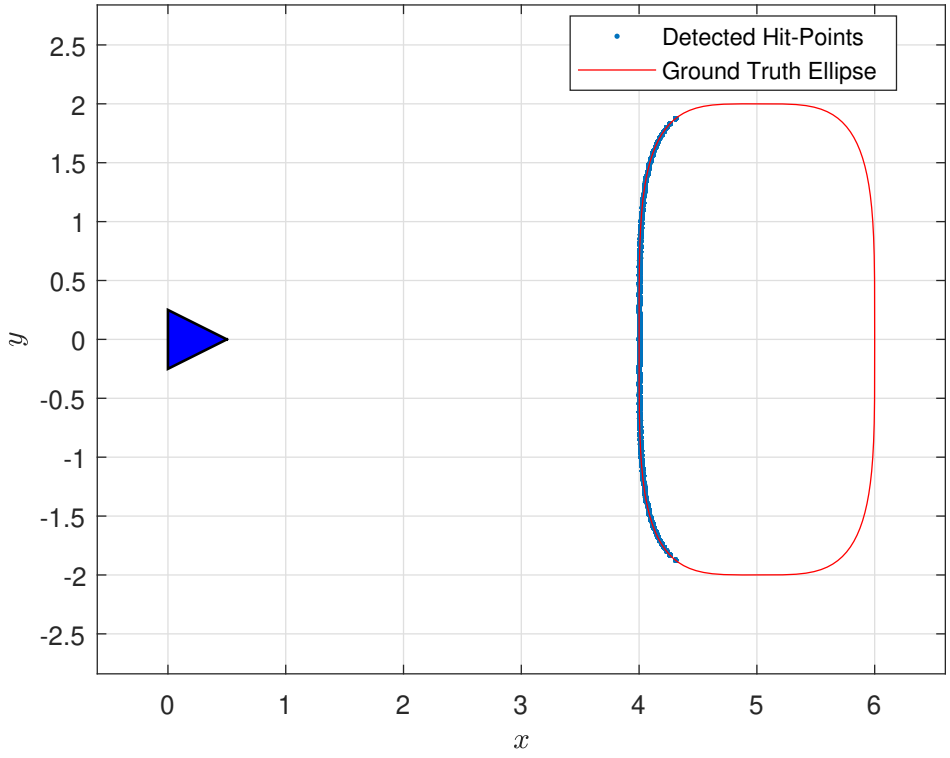


Figure 4.2 Robot in partially occluded scene (the back of the object is not visible to the robot).

4.2.2 Cost Functions for Solving the Super-Ellipse Parameter Estimation Problem

Solving the Super-Ellipse Parameter Estimation Problem involves finding the best fit, or minimizing the geometric distance, between a set of given points and the contour of the estimated curve as defined in (4.2.1). In this context, least-squares minimization problems are tackled using the popular Levenberg–Marquardt algorithm [53].

Simple Cost function

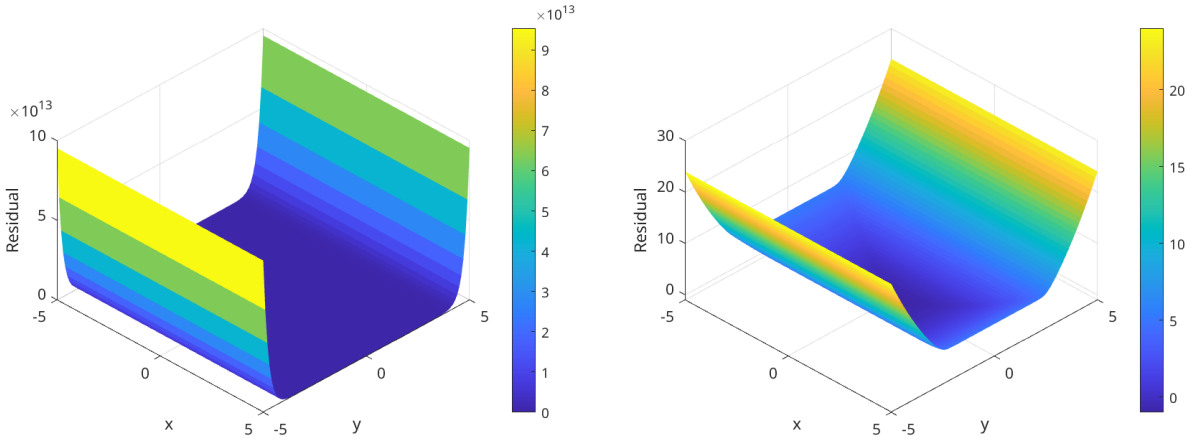
The simplest cost function has a one-dimensional residual and is directly defined as the previously presented implicit function in ((4.1)). The minimization problem is explicitly:

$$\min_{\hat{\boldsymbol{p}}^s} \frac{1}{2} \sum_{\tilde{\boldsymbol{x}} \in \mathcal{D}_{k,o}} \left| \mathcal{F}_s \left(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{p}}^s \right) - 1 \right|^2$$

This solves the parameter estimation of a cluster of hit-points $\mathcal{D}_{k,o}$ on the given contour. An important detail is that the time-step does not yet appear in the equation. This is discussed later when integrating data from multiple scans, the focus of the present chapter is limited to data obtained from a single scan (hence the consideration for limited visibility).

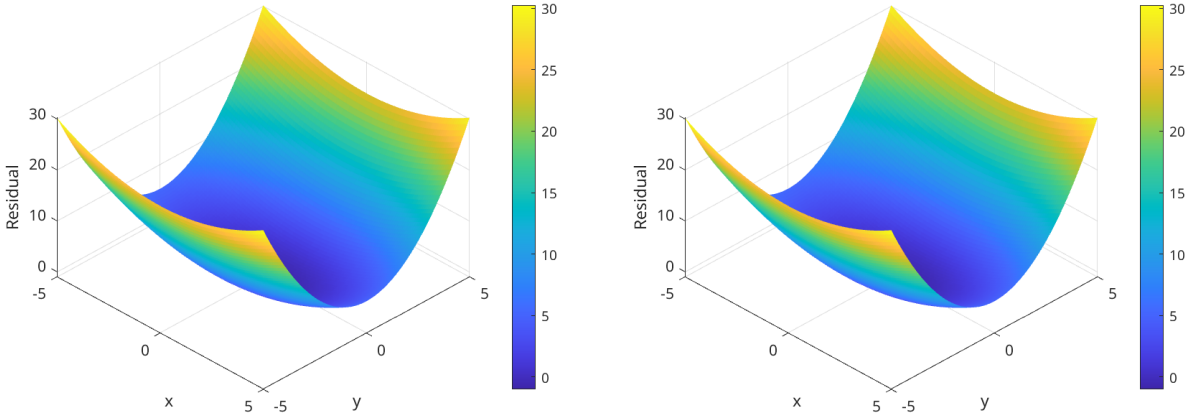
Robust Cost Function

The cost function in (4.2.2) is non-linear. This non-linearity is especially visible in the case of ϵ with values going away from $\epsilon = 1$. Geometrically, this is the value for which the super-ellipse is simply an ellipse. However, for values $0 < \epsilon < 1$ (rectangular) or values $1 < \epsilon < 2$ (diamond-shaped), the shape of $\mathcal{F}_s$ is considerably more uneven, with much stronger slopes. This is apparent in Figures 4.3a, 4.4a and 4.5a.



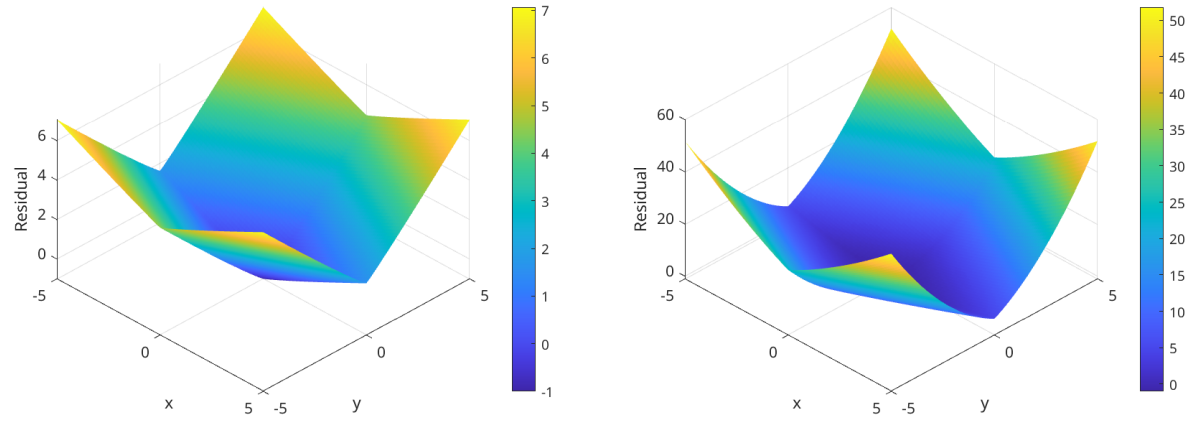
(a) $\mathcal{F}_s - 1$ (b) $\mathcal{F}_s^\epsilon - 1$

Figure 4.3 Residuals with $\epsilon = 0.1$



(a) $\mathcal{F}_s - 1$ (b) $\mathcal{F}_s^\epsilon - 1$

Figure 4.4 Residuals with $\epsilon = 1.0$



(a) $\mathcal{F}_s - 1$ (b) $\mathcal{F}_s^\epsilon - 1$

Figure 4.5 Residuals with $\epsilon = 1.9$

The cost function is subsequently modified as described by Rosin et al., an additional exponent is added [53]. The minimization problem becomes:

$$\min_{\hat{\boldsymbol{p}}^s} \frac{1}{2} \sum_{\tilde{\boldsymbol{x}} \in \mathcal{D}_{k,o}} \left| \mathcal{F}_s \left(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{p}}^s \right)^\epsilon - 1 \right|^2$$

We refer to the above cost function as the robust cost function. The ϵ exponent normalizes the residual. To illustrate, consider the residual at a point while varying ϵ. We choose a point at an equivalent geometric distance from the super-ellipse, regardless of ϵ. For a centered, non-rotated super-ellipse, all points on the x or y axis are valid and have the same geometric distance. Consider $\mathbf{x} = [0 \quad 3]^T$ and super-ellipse parameters $\mathbf{p}^s = [0 \quad 0 \quad 0 \quad 2 \quad 1 \quad \epsilon]$. This is illustrated in Figure 4.6.

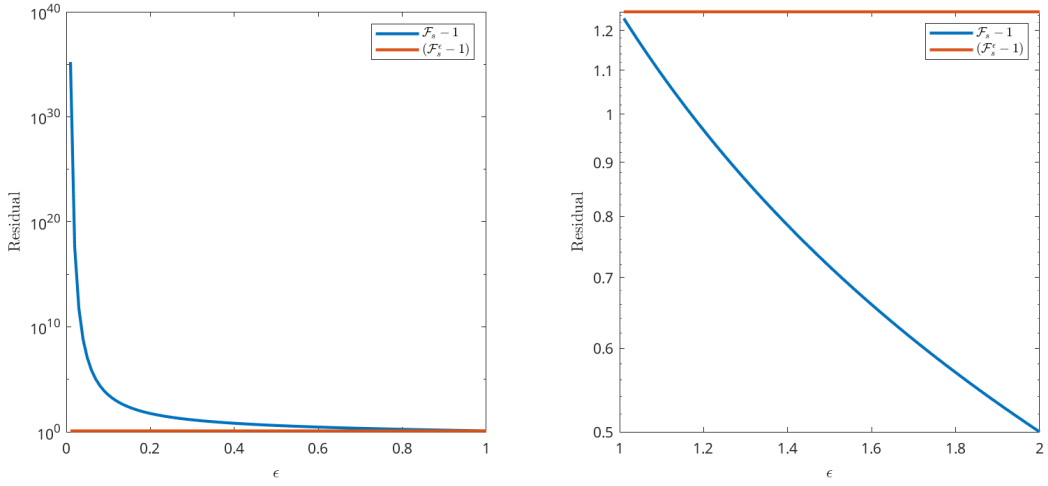


Figure 4.6 Residuals with $\boldsymbol{x} = [0 \quad 3]^T$ and $\boldsymbol{p} = [0 \quad 0 \quad 0 \quad 2 \quad 1 \quad \epsilon]$

This demonstrates the normalizing effect of ϵ on the residuals of the robust cost function. As ϵ varies, the residual grows exponentially near 0 and approaches 0 as ϵ increases. Ideally, the residual should remain constant for a fixed geometric distance, regardless of ϵ. The introduction of ϵ dampens large changes in residual values, achieving a constant residual for all ϵ values, which is more desirable.

This effect is further visualized by plotting residuals with various fixed ϵ values. Figures 4.3, 4.4, and 4.5 show the residuals of both cost functions side by side for varying ϵ. The shape of the cost function impacts the convergence behavior of the Levenberg-Marquardt algorithm. When ϵ deviates from 1, the topology of $\mathcal{F}_s$ becomes uneven, causing a gradient with varying magnitudes. The gradient is stronger near 0 and weaker near 2, leading to issues like overshooting or stagnation in optimization.

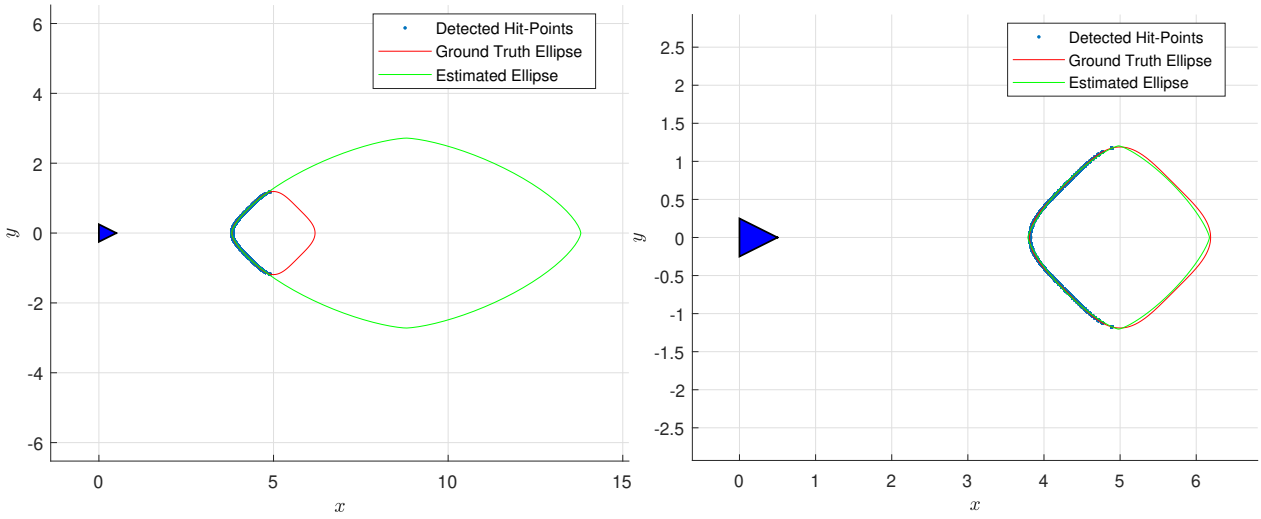
The robust cost function normalizes the surface shape, making the gradient magnitude more consistent across different ϵ values. This results in improved and more reliable convergence behavior in a properly tuned minimization algorithm.

Area Minimizing Cost Function

Adding a term to the cost function for area minimization has been proposed by Jaklic et al. [54]. The Area Minimizing Cost Function, also used by Chaudonneret [55], aims to minimize the area of the super-ellipse while still fitting the data points. By introducing the square root of the product of a and b, the cost function now incorporates a term that is directly related to the area of the super-ellipse. This term acts as a regularizer, discouraging the algorithm from finding solutions with unnecessarily large areas. The minimization problem for the Area Minimizing Cost Function is:

$$\min_{\hat{\boldsymbol{p}}} \frac{1}{2} \sum_{\tilde{\boldsymbol{x}} \in \mathcal{D}_{k,o}} ab |\mathcal{F}_s\left(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{p}}^{\boldsymbol{s}}\right)^{\epsilon} - 1|^2$$

This cost function aims to balance the fit of the super-ellipse to the data points with the minimization of its area. A simple demonstration is illustrated in Figure 4.7, where the minimization is conducted under identical conditions—same robot pose, hit-points, super-ellipse, and initialization parameters. The Area Minimizing Cost Function effectively yields super-ellipses that are much closer to the original data points while avoiding solutions with unnecessarily large areas. Specific results and comparative analyses demonstrating the efficacy of this approach are provided in Section 4.2.4.



(a) $\mathcal{F}^{\epsilon} - 1$ (b) $\sqrt{ab}(\mathcal{F}^{\epsilon} - 1)$

Figure 4.7 Comparison of estimated super-ellipses using the robust and area minimizing cost functions (left to right)

An important note is that the inclusion of $\sqrt{ab}$ as a multiplicative term in front of the residual block is critical for the functioning of the Area Minimizing Cost Function. Placing it in a different dimension of the residual block proved ineffective. One reason for this might be the behavior of the term $(\mathcal{F}_s(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{p}}^{\boldsymbol{s}})^{\epsilon} - 1)$, which can approach zero. In contrast, $\sqrt{ab}$, serving as a proxy for the area, cannot readily approach zero for physically meaningful super-ellipses.

By multiplying $\sqrt{ab}$ with the residual term directly, the optimization algorithm is more strongly guided to balance the dual objectives of data fitting and area minimization. Making it a separate term in a two-dimensional residual block might not offer this balancing act as effectively, because separating $\sqrt{ab}$ as an independent term in a two-dimensional residual block would decouple it from the data-fitting term. Separating $\sqrt{ab}$ as an independent term in a two-dimensional residual block would result in the optimizer treating area minimization and data fitting as independent objectives. This lack of coupling could lead the optimization algorithm to converge to solutions that excel in either data fitting or area minimization, but not both. Therefore, integrating $\sqrt{ab}$ as a multiplicative term with the residual ensures that both objectives are simultaneously considered during the optimization process.

Minimization Constraints for Auto-Occlusion

In parameter estimation for superellipses, another challenge arises known as the auto-occlusion problem. When the observed section of the super-ellipse lacks sufficient curvature, the optimization algorithm may erroneously parameterize the super-ellipse as oriented in the wrong direction, effectively considering the observed portion as the hidden part of the super-ellipse [56]. This issue requires the ability for the algorithm to discard superellipses whose measurements would be implausible due to points being unobservable. A visual depiction of this issue is provided in Figure 4.8.

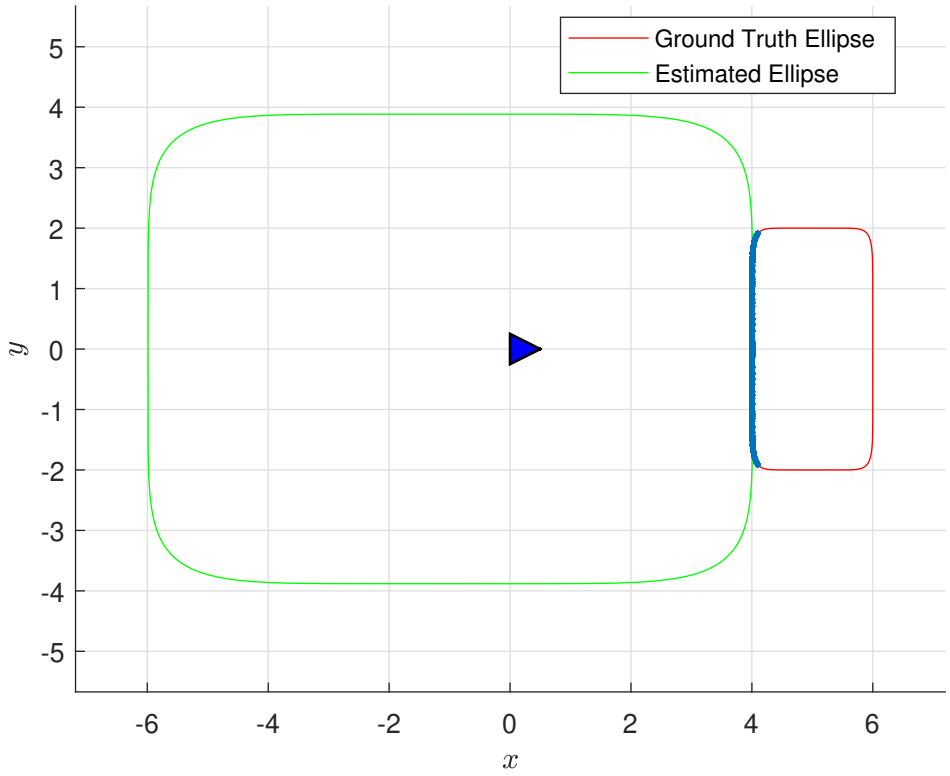


Figure 4.8 Super-ellipse auto-occlusion in parameter estimation

Pascoal et al. propose a method for managing auto-occlusion by using a weighted term in the cost function that transitions smoothly between occluded and visible points [38]. This is controlled by a parameter that adjusts the sharpness of the transition. The method aims to minimize auto-occlusion in optimization. However, the added complexity affects performance [55].

In the present algorithm, preference is given to constraining the minimization problem parameters to fit the physical constraints of the scenario. This is illustrated in Figure 4.9, where the super-ellipse's center coordinates x_c and y_c are constrained to always be *behind* the detected hit-points. This is achieved using the mean of the hit-points on every axis. If the current estimated center of the super-ellipse is close to these mean positions, a blended value of the mean and the current estimate is used to set the constraint. The algorithm to achieve this is presented in Algorithm 1.

Algorithm 1 Determine constraints on the center coordinates of the super-ellipse

Require: Robot's current coordinates (λ, ξ), mean hit-point positions $(\bar{x}, \bar{y})$

Ensure: Constraints on the center coordinates (x_c, y_c) of the super-ellipse

Determine relative position on x-axis
if $\lambda > \bar{x}$ **then**
    Robot is to the right of the shape
    Set upper bound on x_c: $x_c \leq \lambda$
else
    Robot is to the left of the shape
    Set lower bound on x_c: $x_c \geq \lambda$
end if
Determine relative position on y-axis
if $\xi > \bar{y}$ **then**
    Robot is above the shape
    Set upper bound on y_c: $y_c \leq \xi$
else
    Robot is below the shape
    Set lower bound on y_c: $y_c \geq \xi$
end if

This accommodates scenarios where the robot's sensors and the object are closely aligned. These constraints effectively limit the solution space for the optimization problem, guiding it towards solutions where the super-ellipse's center is always "behind" the detected hit-points, thus removing the risk of auto-occlusion. By doing so, the algorithm avoids solutions that would result in auto-occlusion, without adding computational complexity to the optimization problem. Numerically, this method proves viable when using the Gauss-Newton minimization algorithm, as described by Kanzow et al. [57]. This algorithm is implemented in the *Ceres* solver library [58], it is the solver of choice for our implementation. The effectiveness of the constraints is showcased in Figure 4.9.

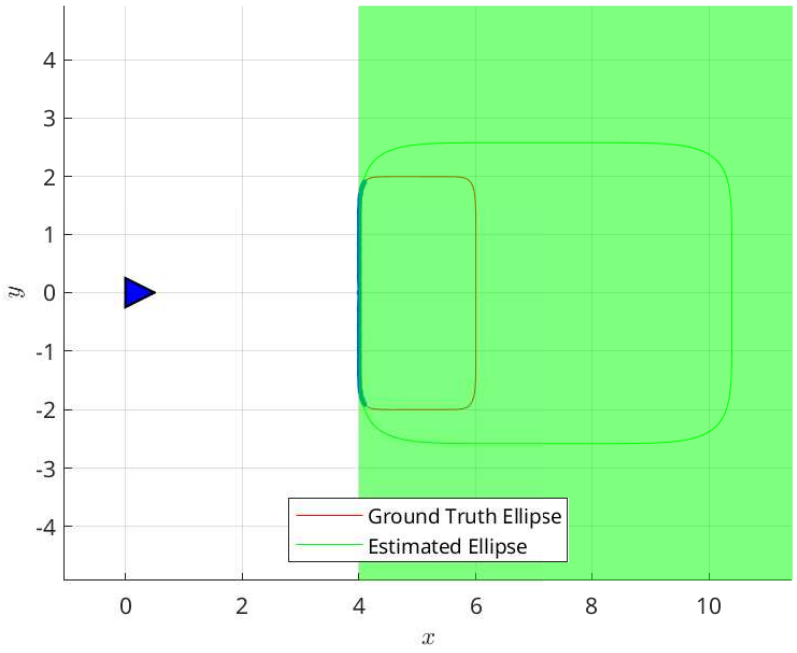


Figure 4.9 Super-ellipse optimized with constraints on coordinates

4.2.3 Initialization Considerations for Solving the Parameter Estimation Problem

Initialization strategies are crucial when dealing with non-linear least squares problems, such as super-ellipse fitting, to avoid local minima. The Levenberg-Marquardt algorithm [59] is often used for solving these types of problems and employing multiple initializations is a commonly used strategy to increase the chances of global convergence. In the context of this work, where local minima are a common issue — as shown in previous sections discussing area minimization and auto-occlusions — multiple initializations are critical for maximizing parameter estimation accuracy. For a non-convex feasible set, initializing with different sets of parameter values for x_c, y_c, ϕ, a, b, and ϵ can guide the optimization algorithm to different parts of the solution space, increasing the likelihood of reaching a global optimum. Below are discussed some effects of various parameter initialization, as well as a methodology for multiple initializations.

Initializing Parameters a, b and ϕ

As in Chaudonneret's work [55], Principal Component Analysis (PCA) [60] is employed to obtain initial estimates for the size parameters $\hat{a}_0$ and $\hat{b}_0$, as well as the orientation $\hat{\phi}_0$ of the

super-ellipse.

Let $M \in \mathbb{R}^{2\times 2}$ be the covariance matrix of the measured hit-point coordinates $(x_i)_{i\in[1,K]}$ and $(y_i)_{i\in[1,K]}$:

$$M = \begin{bmatrix} \mathrm{Var}((x_i)_{i\in[1,K]}) & \mathrm{Cov}((x_i)_{i\in[1,K]}, (y_i)_{i\in[1,K]}) \\ \mathrm{Cov}((x_i)_{i\in[1,K]}, (y_i)_{i\in[1,K]}) & \mathrm{Var}((y_i)_{i\in[1,K]}) \end{bmatrix}$$

Since M is a real, symmetric matrix, it can be eigen-decomposed as:

$$M = VDV^T$$

where V is the orthogonal matrix of eigenvectors and D is the diagonal matrix of eigenvalues (λ_1, λ_2).

From the eigenvalues, the initial size parameters $\hat{a}_0$ and $\hat{b}_0$ can be determined as:

$$\hat{a}_0 = \sqrt{2\lambda_2}$$
$$\hat{b}_0 = \sqrt{2\lambda_1}$$

The initial orientation $\hat{\phi}_0$ is also given by:

$$\hat{\phi}_0 = \mathrm{atan2}(V_{2,1}, V_{1,1})$$

Visually, the PCA initialization is presented in Figure 4.10.

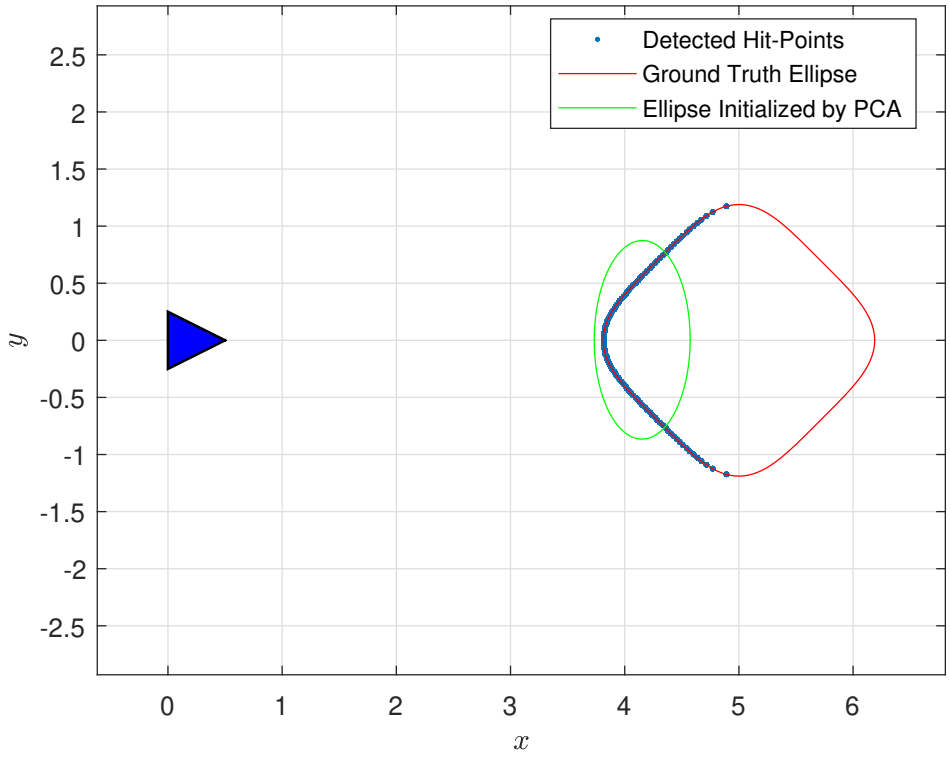


Figure 4.10 PCA initialization

Initializing Parameters x_c, y_c and ϵ

To determine an initial estimate $\hat{\boldsymbol{p}}_0$, the point cloud centroid coordinates are classically [53] used for $\hat{x}_{c,0}$ and $\hat{y}_{c,0}$. The curvature parameter ϵ can be initialized at 1, the midpoint of the considered range $]0, 2[$ for the parameter.

Multiple Initializations

A multi-initialization strategy is employed to address the issue of local minima in the parameter estimation problem [59]. Instead of random initialization within a predefined range, an empirical method for setting initial parameter values is proposed.

For the center parameters x_c, y_c, an alternative initialization strategy is used. The initial center point is shifted by $\sqrt{\hat{a}_0^2 + \hat{b}_0^2}$ in the direction of the vector that connects the robot's position to the centroid of the detected hit-points. This initialization is deemed relevant as the sensor data is partial and captures only one side of the superellipse. It is assumed that the actual object is located *behind* the detected hit-points. This strategy is illustrated in Figure 4.11, we discuss in Section 4.2.4 how it enhances the performance of the least-squares

optimization.

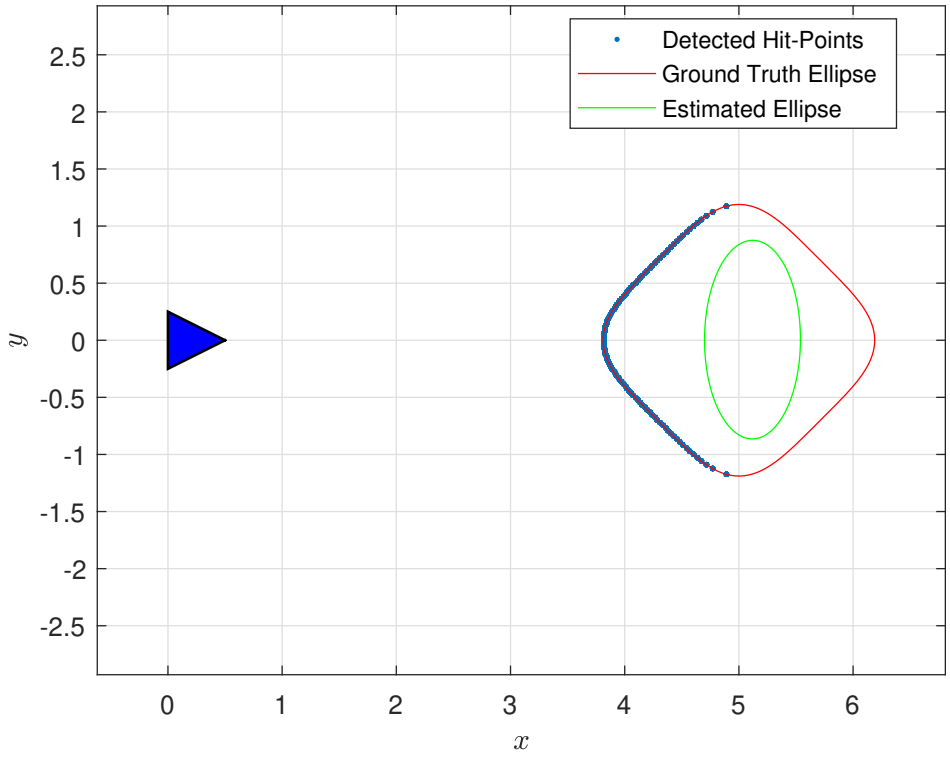


Figure 4.11 Alternative initialization for x_c and y_c

For the ϵ parameter, multiple initializations are also used. The ϵ parameter is particularly susceptible to local minima. To mitigate this, ϵ is initialized at both ends of the considered range $]0,\ 2[$ as well as at the midpoint. This approach is found to increase the likelihood of escaping local minima and converging to a global optimum.

To comprehensively explore the solution space, sets of initializations are defined with two variations for the center parameters x_c, y_c and three variations for the shape parameter ϵ, resulting in a total of $2 \times 3 = 6$ different initialization sets. The variations for x_c, y_c are: 1) initialized at the centroid of the detected hit-points, and 2) initialized at a point shifted by $\sqrt{\hat{a}_0^2 + \hat{b}_0^2}$ in the direction from the robot's position to the centroid of the detected hit-points. The variations for ϵ are: 1) $\epsilon = 0$ (lower bound of the range), 2) $\epsilon = 1$ (midpoint of the range), and 3) $\epsilon = 2$ (upper bound of the range). The sets of initializations are:

Set	x_{c_0}	y_{c_0}	ϕ_0	a_0	b_0	ϵ_0
1	centroid_x	centroid_y	ϕ_0	a_0	b_0	0
2	centroid_x	centroid_y	ϕ_0	a_0	b_0	1
3	centroid_x	centroid_y	ϕ_0	a_0	b_0	2
4	$\text{centroid}_x + \sqrt{a_0^2 + b_0^2}$	$\text{centroid}_y + \sqrt{a_0^2 + b_0^2}$	ϕ_0	a_0	b_0	0
5	$\text{centroid}_x + \sqrt{a_0^2 + b_0^2}$	$\text{centroid}_y + \sqrt{a_0^2 + b_0^2}$	ϕ_0	a_0	b_0	1
6	$\text{centroid}_x + \sqrt{a_0^2 + b_0^2}$	$\text{centroid}_y + \sqrt{a_0^2 + b_0^2}$	ϕ_0	a_0	b_0	2

Table 4.1 Initialization Sets for Parameters

Each of these 6 sets is processed independently using the Levenberg-Marquardt algorithm. Upon convergence, the solution with the lowest loss is selected as the optimal parameter set.

4.2.4 Performance

The effectiveness of the parameter estimation algorithm for super-ellipses is evaluated using a numerical approach. Key performance metrics are defined and the Monte-Carlo method is used as the basis for this assessment.

Methodology and Metrics

We want to evaluate how close our estimated super-ellipses are to the ground truth super-ellipses. Due to the complexity of obtaining an analytical solution for overlapping super-ellipse areas, a numerical approach is employed for performance evaluation. The super-ellipses are placed on a high-resolution grid, and each grid point is evaluated to determine its location relative to the ground truth and estimated super-ellipses.

Two key performance metrics are defined:

1. The first metric, Ratio_1, is calculated as the ratio of the number of grid points that lie inside both the ground truth and estimated super-ellipses to the number of grid points that lie inside the ground truth super-ellipse. Mathematically,

$$\text{Ratio}_1 = \frac{\text{Number of points inside both ground truth and estimated (True Positive)}}{\text{Number of points inside ground truth}}$$

2. The second metric, Ratio_2, is the ratio of the number of grid points that lie inside the estimated super-ellipse but outside the ground truth super-ellipse to the number of grid

points that lie inside the estimated super-ellipse only. Mathematically,

$$\text{Ratio}_2 = \frac{\text{Number of points inside estimated but outside ground truth (False Positive)}}{\text{Number of points inside estimated}}$$

In Figure 4.12, true positives, false positives, false negatives and true negatives are visually illustrated. The scene is the same as the one in Figure 4.11.

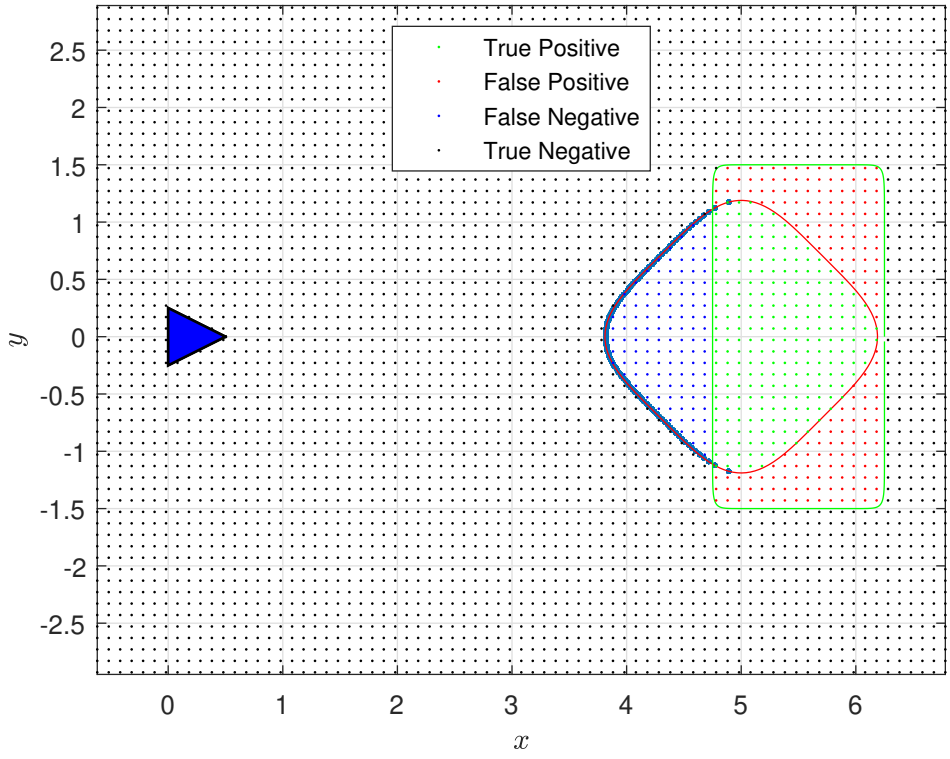


Figure 4.12 True positives, false positives, false negatives, and true negatives in the same scene as Figure 4.11.

Ratio_1 aims to be maximized towards 1, indicating a perfect overlap between the estimated and ground truth super-ellipses. A Ratio_1 value close to 1 signifies that the estimated super-ellipse accurately captures the shape and area of the ground truth, thereby validating the effectiveness of the parameter estimation process. Lower values of Ratio_1 indicate that the estimated super-ellipse fails to encompass the ground truth adequately, leading to false negatives.

On the other hand, Ratio_2 aims to be minimized towards 0. A Ratio_2 value close to 0 indicates that there are relatively few or no points that are incorrectly included in the estimated super-ellipse but not in the ground truth. This minimizes the false positive rate, ensuring that the estimated super-ellipse does not overextend beyond the actual shape. Higher values of Ratio_2

would imply that the estimated super-ellipse is too inclusive, capturing areas that are not part of the ground truth.

Together, these ratios provide a balanced measure of both the sensitivity and specificity of the parameter estimation process. A high Ratio_1 and a low Ratio_2 would signify an optimal parameter set and perfect estimate.

The Monte-Carlo method is a statistical technique for assessing the effectiveness of various algorithms [61]. The Monte-Carlo method involves running multiple simulations with random inputs within specified ranges to approximate the overall performance of the algorithm. This approach allows for a comprehensive understanding of the algorithm's behavior under various conditions.

A custom simulation software was developed to implement the Monte-Carlo method for this study. The software generates physically plausible scenarios based on user-defined ranges for ground truth parameters. Each scenario is then processed through the parameter estimation algorithm, and the performance metrics Ratio_1 and Ratio_2 are computed. The software ensures that all generated scenarios are physically possible, adhering to the constraints outlined previously.

Analysis and Interpretation

Mean values for Ratio_1 and Ratio_2 were calculated using Monte Carlo simulations, estimating the central tendency of the algorithm's performance.

The Standard Error of the Mean (SEM) for Ratio_1 and Ratio_2 was computed to estimate how far the sample mean is from the true population mean: $\text{SEM} = \frac{\sigma}{\sqrt{n}}$, where σ is the sample standard deviation and n is the sample size. The performance is expressed in terms of the number of standard deviations (σ) from the mean: $\text{Interval} = \text{Mean} \pm k \times \text{SEM}$. We choose $k = 2$.

As the number of Monte Carlo iterations increases, the sample size grows, reducing the SEM and narrowing the interval. This increased precision provides a more reliable assessment of the algorithm's performance for Ratio_1 and Ratio_2. Therefore, more iterations contribute to a more accurate evaluation of the parameter estimation algorithm for super-ellipses.

Results

The results are presented in tabular form (Tables 4.2 and 4.3) and in box plots (Figures 4.13 and 4.14), displaying the mean, SEM, and confidence interval for each ratio and each combination of initialization sets (Table 4.1) and cost functions. The tables also include a column for the optimal set, which evaluates all six sets and automatically selects the one with the lowest loss.

For Ratio_1 and looking at the optimal sets, the Simple cost function generally shows the highest mean values, indicating that it covers a large portion of the ground truth area. However, when considering Ratio_2, the Simple cost function exhibits significantly high mean values, ranging from approximately 45% to 97%. This indicates an unacceptable rate of false positives, where a large portion of the estimated super-ellipse extends much beyond the ground truth area.

The Robust cost function demonstrates an improvement over the Simple cost function, with slightly lower mean values for Ratio_1 but a considerable reduction in mean values for Ratio_2. This suggests that the Robust cost function achieves a better balance between covering the ground truth area and minimizing false positives, making it a more reliable option than the Simple cost function.

The Area Minimizing cost function further improves upon the Robust cost function, showing the lowest mean values for Ratio_2 among all three cost functions. This indicates the fewest false positives and the most conservative estimation of the super-ellipse. However, the mean values for Ratio_1 are also lower for the Area Minimizing cost function, suggesting that it may be more conservative in covering the ground truth area compared to the Robust cost function.

Regarding the confidence intervals, the results from the 1000 Monte-Carlo simulations provide a solid statistical basis for our analysis. The SEM and the confidence intervals calculated for both Ratio_1 and Ratio_2 indicate that our estimates are statistically robust. This level of confidence, coupled with the relatively narrow confidence intervals, demonstrates that 1000 simulations are sufficient to provide reliable and precise estimates of the performance metrics for our parameter estimation algorithm.

In conclusion, the performance analysis of the cost functions reveals that the Area Minimizing cost function is the most suitable for our application. Although it tends to slightly underestimate the ground truth area, this conservative approach results in the lowest rate of

false positives, making it highly reliable. Given that our mapping process involves multiple scans, any underestimation can be compensated for by tuning the refinement algorithm in subsequent steps.

Table 4.2 Cost Function Statistics for Ratio_1 over 1000 Monte-Carlo iterations

Cost Function / Set	Mean of R_1	SEM of R_1	CI of R_1
Simple / Set 1	0.6666	0.0127	0.6666 ± 0.0254
Simple / Set 2	0.7316	0.0128	0.7316 ± 0.0256
Simple / Set 3	0.5933	0.0135	0.5933 ± 0.0270
Simple / Set 4	0.4003	0.0095	0.4003 ± 0.0190
Simple / Set 5	0.7767	0.0123	0.7767 ± 0.0246
Simple / Set 6	0.7499	0.0132	0.7499 ± 0.0264
Simple / Optimal Set	0.7857	0.0126	0.7857 ± 0.0252
Robust / Set 1	0.7242	0.0130	0.7242 ± 0.0260
Robust / Set 2	0.7492	0.0126	0.7492 ± 0.0252
Robust / Set 3	0.7174	0.0132	0.7174 ± 0.0264
Robust / Set 4	0.7282	0.0127	0.7282 ± 0.0254
Robust / Set 5	0.8064	0.0116	0.8064 ± 0.0232
Robust / Set 6	0.8205	0.0114	0.8205 ± 0.0228
Robust / Optimal Set	0.7897	0.0125	0.7897 ± 0.0250
Area Minimizing / Set 1	0.2394	0.0094	0.2394 ± 0.0188
Area Minimizing / Set 2	0.5841	0.0131	0.5841 ± 0.0262
Area Minimizing / Set 3	0.2855	0.0101	0.2855 ± 0.0202
Area Minimizing / Set 4	0.3772	0.0126	0.3772 ± 0.0252
Area Minimizing / Set 5	0.5503	0.0130	0.5503 ± 0.0260
Area Minimizing / Set 6	0.5237	0.0131	0.5237 ± 0.0262
Area Minimizing / Optimal Set	0.6336	0.0129	0.6336 ± 0.0258

Table 4.3 Cost Function Statistics for $Ratio_2$ over 1000 Monte-Carlo iterations

Cost Function / Set	Mean of R_2	SEM of R_2	CI of R_2
Simple / Set 1	0.6439	0.0130	0.0260
Simple / Set 2	0.6156	0.0141	0.0282
Simple / Set 3	0.4453	0.0136	0.0272
Simple / Set 4	0.6708	0.0109	0.0218
Simple / Set 5	0.8232	0.0109	0.0218
Simple / Set 6	0.9741	0.0030	0.0060
Simple / Optimal Set	0.8302	0.0109	0.0218
Robust / Set 1	0.6688	0.0134	0.0268
Robust / Set 2	0.6237	0.0140	0.0280
Robust / Set 3	0.6421	0.0138	0.0276
Robust / Set 4	0.8715	0.0086	0.0172
Robust / Set 5	0.7937	0.0116	0.0232
Robust / Set 6	0.9605	0.0035	0.0070
Robust / Optimal Set	0.8080	0.0114	0.0228
Area Minimizing / Set 1	0.2238	0.0093	0.0186
Area Minimizing / Set 2	0.1716	0.0099	0.0198
Area Minimizing / Set 3	0.2118	0.0093	0.0186
Area Minimizing / Set 4	0.2151	0.0099	0.0198
Area Minimizing / Set 5	0.1966	0.0103	0.0206
Area Minimizing / Set 6	0.1648	0.0097	0.0194
Area Minimizing / Optimal Set	0.1436	0.0097	0.0194

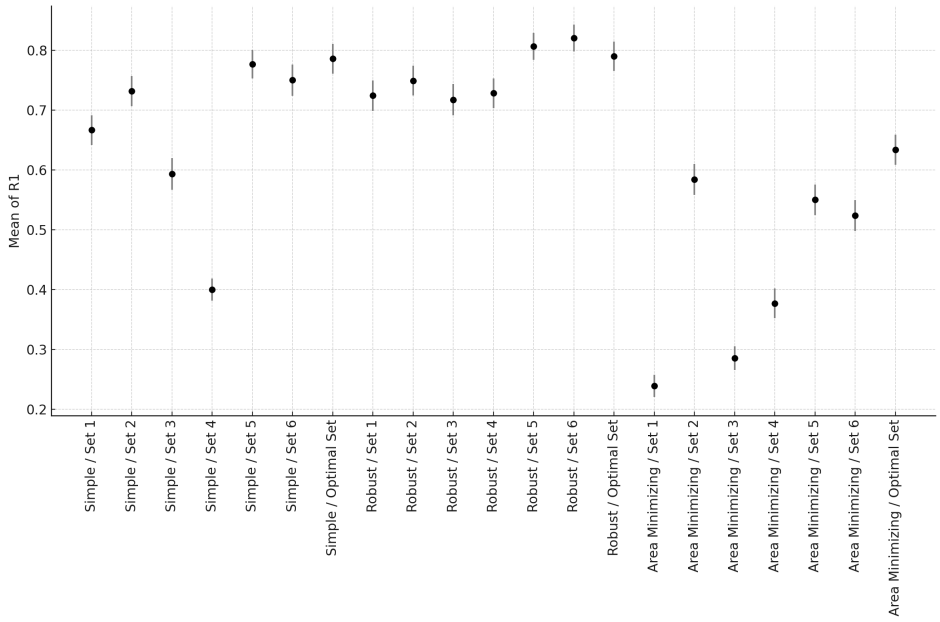


Figure 4.13 Cost Function Statistics for Ratio_1 over 1000 Monte-Carlo iterations

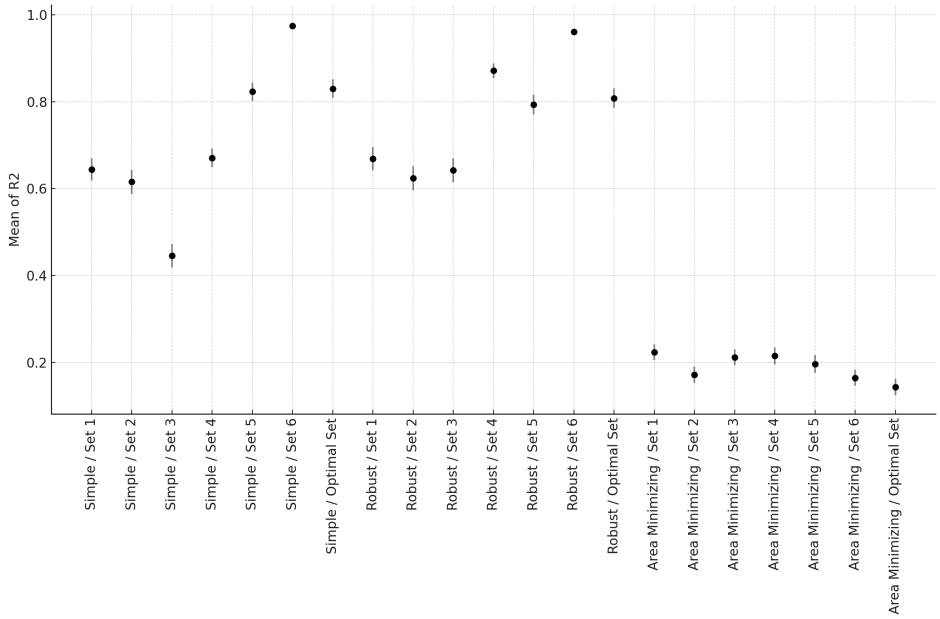


Figure 4.14 Cost Function Statistics for Ratio_2 over 1000 Monte-Carlo iterations

4.3 Sub-Line Parameter Estimation

Parameterizing sub-lines builds upon the parameter estimation of infinite lines. A sub-line is a subset of a line, confined using two additional parameters.

4.3.1 The Line Parameter Estimation Problem

In a manner analogous to the super-ellipse, a line can also be parameterized implicitly. Given a line in the plane, let θ be the angle that a vector perpendicular to the line makes with the x-axis, and let r be the perpendicular distance from the origin to the line.

The implicit equation for a line not passing by the origin has parameter vector:

$$\boldsymbol{p}^l = \begin{bmatrix} \theta & r \end{bmatrix}^T,$$

and can be expressed as in [55] by:

$$\mathcal{F}_l\left(\boldsymbol{x}_{k,i}, \boldsymbol{p}^l\right) = \frac{1}{r}(x_{k,i}\cos\theta + y_{k,i}\sin\theta).$$

The implicit form is made similar to the super-ellipse presented prior:

$$\mathcal{F}_l\left(\boldsymbol{x}_{k,i}, \boldsymbol{p}^l\right) = 1.$$

The implicit line equation is visually represented in Figure 4.15.

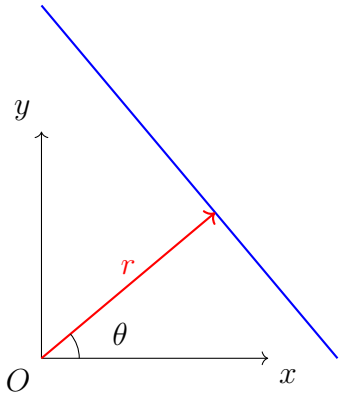


Figure 4.15 Implicit line (blue) visualization

The minimization problem for parameterizing line segments can be formulated as:

$$\min_{\boldsymbol{q}} \frac{1}{2} \sum_{\boldsymbol{x}_{k,i} \in \mathcal{D}_{k,l}} \left| \mathcal{F}_l \left(\boldsymbol{x}_{k,i}, \boldsymbol{q} \right) - 1 \right|^2$$

Here, the objective is to minimize the squared residuals of the implicit line function $\mathcal{F}_l$ for each hit-point $\boldsymbol{x}_{k,i}$ in the set $\mathcal{D}_{k,l}$. The problem can be reformulated in a matrix form. The intermediate parameter vector $\boldsymbol{z}$ is defined as $\boldsymbol{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$, where $z_1 = \frac{1}{r}\cos(\theta)$ and $z_2 = \frac{1}{r}\sin(\theta)$. This formulation is linear with respect to the intermediate parameter vector $\boldsymbol{z}$. The problem formulated as a standard linear least squares problem is

$$\min_{\boldsymbol{z}} \left| \boldsymbol{A}\boldsymbol{z} - \boldsymbol{b} \right|^2$$

Here, $\boldsymbol{A}$ is constructed from the hit-point coordinates $\boldsymbol{x}_{k,i}$ with the implicit line function $\mathcal{F}_l$, and $\boldsymbol{b}$ is a vector of ones. The intermediate parameter vector $\boldsymbol{z}$ can be calculated using:

$$\boldsymbol{z} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$$

Obtaining the actual parameter vector $\boldsymbol{p}^l = \begin{bmatrix} \theta & r \end{bmatrix}^T$ from $\boldsymbol{z}$ is trivial:

$$\theta = \arctan 2(z_2, z_1),$$
$$r = \frac{1}{\sqrt{z_1^2 + z_2^2}}.$$

4.3.2 Handling Sub-Lines

To parameterize a sub-line as a subset of a line, two additional parameters are introduced: α and β, which are angles that define the beginning and the end of the sub-line. This is illustrated in Figure **??**.

Additional Sub-Line Parameters

The parameter vector for the sub-line is then defined as an extension of the parameterized lines:

$$\boldsymbol{p}^{sl} = \begin{bmatrix} \theta & r & \alpha & \beta \end{bmatrix}^T$$

A point $\boldsymbol{x}_{k,i}$ lies on the sub-line if:

$$\mathcal{F}_l\left(\boldsymbol{x}, \boldsymbol{p}^l\right) = 0 \quad \text{and} \quad \alpha \leq \angle(\boldsymbol{x}) \leq \beta$$

Here, $\angle(\boldsymbol{x})$ is the angle between the vector $\boldsymbol{x}$ and the x-axis. The angles α and β define the bounds of the sub-line; any point within these angular bounds and satisfying the line equation is part of the sub-line. The parameter estimation of the sub-line, using α and β measured with respect to the x-axis, can be visually represented in Figure 4.16.

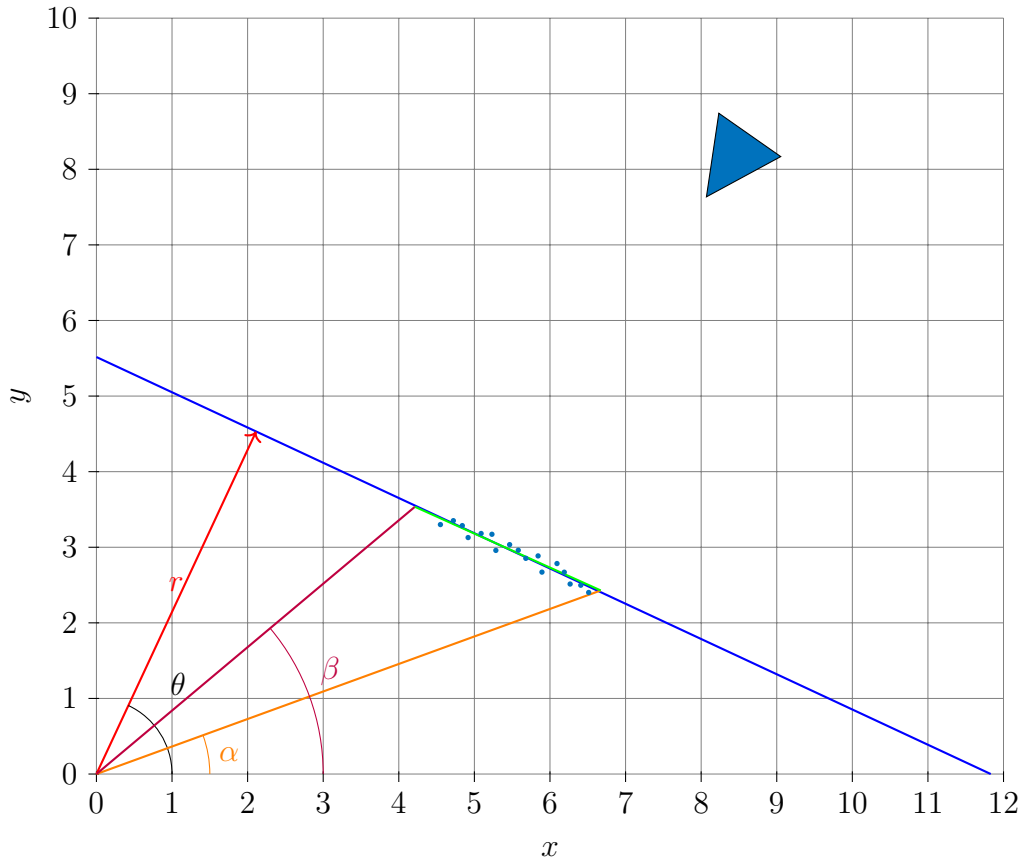


Figure 4.16 Sub-line (green) parameter estimation

To obtain the additional parameters α and β, critical hit-points from the set $\mathcal{D}_{k,l}$ are determined with respect to the angle parameter θ in $\boldsymbol{q}$. Specifically, the critical hit-points are those that minimize and maximize the angular difference relative to θ.

Let $\Delta\theta = \angle(\boldsymbol{x}) - \theta$, then α and β can be defined as:

$$\alpha = \min_{\boldsymbol{x} \in \mathcal{D}_{k,l}} \Delta\theta$$

$$\beta = \max_{\boldsymbol{x} \in \mathcal{D}_{k,l}} \Delta\theta$$

Combining Implicit Sub-Lines

Two sub-lines are considered overlapping if they share the same θ and r parameters within a certain tolerance ϵ, and if their angular boundaries α and β intersect. Mathematically, this is expressed as:

$$\begin{aligned}
&(|\theta_1 - \theta_2| < \epsilon_\theta, \quad |r_1 - r_2| < \epsilon_r) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{and} \\
[((\alpha_1 \mod 2\pi) \leq (\beta_2 \mod 2\pi) &\text{ and } (\alpha_2 \mod 2\pi) \leq (\beta_1 \mod 2\pi)) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{or} \\
((\beta_1 \mod 2\pi) \leq (\alpha_2 \mod 2\pi) &\text{ and } (\beta_2 \mod 2\pi) \leq (\alpha_1 \mod 2\pi)) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{or} \\
((\alpha_1 \mod 2\pi) \leq (\alpha_2 \mod 2\pi) &\text{ and } (\beta_2 \mod 2\pi) \leq (\beta_1 \mod 2\pi))].
\end{aligned}$$

If the overlapping condition is met, the sub-lines can be combined parametrically into a new sub-line. The θ and r parameters for the new sub-line are averaged:

$$\theta_{\text{new}} = \frac{\theta_1 + \theta_2}{2}, \quad r_{\text{new}} = \frac{r_1 + r_2}{2}$$

The angular boundaries for the new sub-line are then defined as the minimum of the α parameters and the maximum of the β parameters from the overlapping sub-lines:

$$\alpha_{\text{new}} = \min(\alpha_1, \alpha_2), \quad \beta_{\text{new}} = \max(\beta_1, \beta_2)$$

The parameter vector for the combined sub-line is then:

$$\boldsymbol{p}^{sl}_{\text{new}} = \begin{bmatrix} \theta_{\text{new}} & r_{\text{new}} & \alpha_{\text{new}} & \beta_{\text{new}} \end{bmatrix}^T$$

4.4 Simultaneous Function-Aware Parameter Estimation and Segmentation

Segmentation plays a pivotal role in mapping systems that differentiate objects. Function-aware segmentation aims to associate observed data points to a member of different objects.

The objective is including environmental knowledge into the segmentation step, with an inherent mechanism for outlier rejection, as data that does not conform to the function within a certain tolerance can be readily identified and discarded. A mechanism for function-aware parameter estimation and segmentation should be model-agnostic, it should not be specific to the implicit functions used or the post-processing step of every model. Assumptions about the environment have previously been presented in Section 3.1.

In previous studies, Kalman filters have been utilized for function-aware segmentation with geometrical functions that are simpler (have fewer parameters) than super-ellipses [17], [14]. The EKF is used in a framework for implementing function-aware segmentation for non-linear functions. EKFs are well-suited for problems where the state dynamics and observation models are non-linear, a condition met by the super-ellipse implicit function forms, as well as complex geometrical functions at large. EKFs inherently account for noise in both the state and observations, offering a probabilistic foundation for the framework.

4.4.1 EKF for Implicit Function Parameter Prediction

The problem presented in Section 4.2.1 is approached using an EKF. EKFs and other particle filters process data sequentially. This is desirable for function-aware segmentation. All equations and variables discussed are in the context of the k^{th} scan, and they evolve according to index i representing the i^{th} measurement. The ordering i is important for our 2D mapping method, it plays the role of time in the standard operations of EKFs. The implicit function at hand is any implicitly defined $\mathcal{F}_c = 1$ with parameters $\boldsymbol{p}^c$ being the EKF's state (abbreviated to $\mathcal{F}$ and $\boldsymbol{p}$). The measurement vector is defined in (3.1). The measurement vector with noise becomes:

$$\tilde{\boldsymbol{m}} = \boldsymbol{m} + \boldsymbol{\nu}$$

with $\boldsymbol{\nu}$ the measurement noise, with only one non-zero element for the distance component:

$$\boldsymbol{\nu} = \begin{bmatrix} \nu_d \\ 0 \end{bmatrix}$$

Equation (3.2) is used to get the noised hit-points:

$$L(\boldsymbol{\Xi}, \tilde{\boldsymbol{m}}) = \tilde{\boldsymbol{x}}.$$

State Evolution

The state vector $\boldsymbol{p}$ evolves according to the following equation:

$$\boldsymbol{p}_{i+1} = \boldsymbol{p}_i + \boldsymbol{\omega}_p \tag{4.2}$$

where $\boldsymbol{\omega}_p \sim \mathcal{N}(0, \boldsymbol{Q})$ is the noise vector associated to the state estimation.

Prediction

$$\begin{aligned} \text{State Prediction:} \quad & \hat{\boldsymbol{p}}_{i+1|i} = \hat{\boldsymbol{p}}_{i|i} + \boldsymbol{\omega}_p \\ \text{Covariance Prediction:} \quad & \boldsymbol{P}_{i+1|i} = \boldsymbol{P}_{i|i} + \boldsymbol{Q} \end{aligned}$$

Linearization

The Jacobian matrix $\boldsymbol{H}_{i+1}$ is defined as:

$$\boldsymbol{H}_{i+1} = \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{p}}\right|_{\tilde{\boldsymbol{x}}_i - \nu_i, \hat{\boldsymbol{p}}_{i+1|i}} \tag{4.3}$$

The Jacobian matrix $\boldsymbol{J}_{i+1}$ related to the noise $\boldsymbol{\nu}$ is defined as:

$$\boldsymbol{J}_{i+1} = \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{\nu}}\right|_{\tilde{\boldsymbol{x}}_i - \boldsymbol{\nu}_i, \hat{\boldsymbol{p}}_{i+1|i}} \tag{4.4}$$

The observation noise covariance matrix $\boldsymbol{R}_{i+1}$ is then given by:

$$\boldsymbol{R}_{i+1} = \boldsymbol{J}_{i+1} \boldsymbol{W} (\boldsymbol{J}_{i+1})^\top \tag{4.5}$$

where $\boldsymbol{W}$ is the noise covariance matrix for the measurement noise $\boldsymbol{\nu}$:

$$\boldsymbol{W} = \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & 0 \end{bmatrix}$$

where σ_d^2 is the variance of the distance component ν_d of the measurement noise vector ν.

Update

A pseudo-measurement z is derived from the implicit function $\mathcal{F}$. Unlike direct measurements, it does not come from sensors but is computed based on the function and state estimates. In this context, $z = 0$ ensures that the observed LiDAR hit-points conform to the shape defined by $\mathcal{F}_s$. Using $z = 0$ is the direct evaluation of the real measured hit-points $\hat{\boldsymbol{x}}$ according to the implicit function. It is defined as follows:

$$\hat{z} = \mathcal{F}(L(\boldsymbol{\Xi}_k, \hat{\boldsymbol{m}}_{k,i}), \hat{\boldsymbol{p}}_{i+1|i}) - 1$$

Subsequently, the update equations are:

$$\begin{aligned}
\text{Kalman Gain:} \quad & \boldsymbol{K}_{i+1} = \boldsymbol{P}_{i+1|i}(\boldsymbol{H}_{i+1})^\top (\boldsymbol{H}_{i+1}\boldsymbol{P}_{i+1|i}(\boldsymbol{H}_{i+1})^\top + \boldsymbol{R})^{-1} \\
\text{State Update:} \quad & \boldsymbol{p}_{i+1|i+1} = \hat{\boldsymbol{p}}_{i+1|i} + \boldsymbol{K}_{i+1}(0 - z) \\
\text{Covariance Update:} \quad & \boldsymbol{P}_{i+1|i+1} = (\boldsymbol{I} - \boldsymbol{K}_{i+1}\boldsymbol{H}_{i+1})\boldsymbol{P}_{i+1|i}
\end{aligned}$$

Initialization

Kalman filters require both the estimated state and their covariance to be initialized. Using a number of hit-points assumed as belonging to a single object, an initial estimate is made using the least-squares methods outlined in Section 4.2. Quantifying the error on the non-linear least-squares estimate is not a trivial task. H. Gavin [59] provides an approach to estimate this error using the covariance matrix of the parameter estimates using the propagation model. Based on H. Gavin's method, the equation can be formulated as:

$$\boldsymbol{P}_{0|0} = (\boldsymbol{J}^\top \boldsymbol{W} \boldsymbol{J})^{-1}$$

Here, $\boldsymbol{P}_{0|0}$ represents the initial estimate of the parameter covariance matrix, $\boldsymbol{J}$ is the Jacobian matrix of the system, and $\boldsymbol{W}$ is the measurement error covariance matrix. S. Chaudonneret [55] suggests the incorporation of the LiDAR measurement model from Equation (3.2) into the parameter covariance calculation in order to account for a non-linear noisy observation. With $\boldsymbol{H}$ the Jacobian matrix related to the state model defined in (4.3), $\boldsymbol{\Sigma}_{mes}$ is the measurement error covariance matrix derived from the LiDAR model, represented as

$$\Sigma_{\text{mes}} = \boldsymbol{B}\sigma_d^2\boldsymbol{B}^\top$$

where the Jacobian w.r.t. the measurement is:

$$\boldsymbol{B} = \frac{\partial \mathcal{F}}{\partial \boldsymbol{m}}$$

Incorporating S. Chaudonneret's suggestion to include the LiDAR measurement model into the parameter covariance calculation, the equation can be extended as:

$$\boldsymbol{P}_{0|0} = \left(\boldsymbol{H}^\top \boldsymbol{\Sigma}_{\text{mes}}^{-1} \boldsymbol{H}\right)^{-1}$$

This amounts to a covariance weighted by both observation and measurement models.

4.4.2 The Mahalanobis Distance for LiDAR Feature Classification

As mentioned previously, choosing between models is done using conditions that depend on the Mahalanobis distance (also called NIS) [18] [45]. This is an obvious choice as the metric has a proven track record of being used as a number indicating the compliance of a data point to a given geometric model. The Mahalanobis distance threshold is defined as a multi-dimensional Euclidean distance between a point and a distribution that is weighed by the inverse of the covariance. Given a probability distribution D on $\mathbb{R}^N$ of mean $\boldsymbol{\mu}$ and a positive-definite covariance matrix S, the Mahalanobis distance of the point $\boldsymbol{x}$ to the distribution D is defined in [62] as follows:

$$d_M(\boldsymbol{x}, D) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T S^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}$$

Upon examination of this definition, it can be observed that if a dimension is strongly correlated with another in the distribution (high covariance between them), the Mahalanobis distance is affected by this strong correlation. Specifically, when dimensions are strongly correlated, the ellipsoid will be squeezed along the direction of the correlation. The distance of a point in this direction will then be scaled down, as the Mahalanobis distance takes into account this correlation, unlike the Euclidean distance. In other words, the structure of the data is accounted for by the Mahalanobis distance, and strong correlation between dimensions will cause the distance measure to be less sensitive to variations along the direction of correlation. Points that lie along the direction of strong correlation may therefore have a smaller Mahalanobis distance from the mean of the distribution than they would if the dimensions were uncorrelated.

The sensitivity to the covariance is desired for classifying individual LiDAR measurements

as belonging or not to an existing feature according to pre-determined criteria. In order to accomplish this, the following must be achieved:

(a) A distribution for the parameter uncertainties must be established.

(b) The ability to map the LiDAR measurements to parameter uncertainties must be ensured.

(c) The capability to correlate the uncertainties between parameters must be provided.

Therefore, the Mahalanobis distance between a LiDAR measurement $\boldsymbol{m}_{k,i}$ and a feature is defined, highlighting the above requirements. For the implicit functions $\mathcal{F}(\boldsymbol{\Xi}_k, \hat{\boldsymbol{p}}_{k,i})$, the Mahalanobis distance is a scalar defined as follows:

$$d^M_{\left[\boldsymbol{m}_{k,i} \to \mathcal{F}(\boldsymbol{\Xi}_k, \boldsymbol{p})\right]} = \frac{r^2_{k,i}}{S_{k,i}}$$

with:

$$r_{k,i} = \mathcal{F}\left(L(\boldsymbol{\Xi}_k, \boldsymbol{m}_{k,i}), \boldsymbol{p}\right) \tag{4.6}$$

$$S_{k,i} = \boldsymbol{H}_{i+1} \boldsymbol{P}_{i+1|i} (\boldsymbol{H}_{i+1})^\top + \boldsymbol{R}, \tag{4.7}$$

With $\boldsymbol{R}$ defined in (4.5), $\boldsymbol{H}$ defined in (4.3), and $\boldsymbol{J}$ defined in (4.4). It can be immediately noticed that the Euclidean distance in (4.4.2) is reproduced using the algebraic error of the point with respect to the feature's implicit function in (4.6). The covariance matrix in (4.7) is designed in a way to integrate requirements (b) and (c).

The Mahalanobis distance threshold is therefore used to classify individual LiDAR measurements in relation to existing features. The threshold ensured that the three key requirements (a), (b) and (c) are met by incorporating the given measures of uncertainty

4.4.3 Dual EKFs for Segmentation

One of the challenges in function-based segmentation is the difficulty in initializing new aggregate functions due to not knowing if the initialization hit-points accurately correspond to the chosen function. To address this issue, the dual EKF strategy employs two identical EKFs successively, each with distinct flexibility settings during different phases of segmentation. The dual EKF architecture was first introduced in [15] for handling line and circle functions. This approach was later extended to accommodate more complex super-ellipse functions

in [55]. These foundational works form the basis for the segmentation methodology presented here.

In the dual EKF approach, two EKFs, namely the flexible filter and the strict filter, are employed successively to enhance function-based segmentation. The threshold used is the previously discussed Mahalanobis distance threshold, here denoted τ. Figure 4.17 provides a state-machine representation that captures the transitions between these various states. The methodology is articulated as follows:

1. **Initialization**: This is simply the reset state. The variable n is set to 0.

2. **Continuity**: At this state, the incoming measurements are assessed for continuity. Angular ($\Delta\theta$) and depth (Δd) variations are employed for this evaluation. A discontinuity is identified if either $\Delta\theta$ or Δd exceed user-defined thresholds. If a discontinuity is identified, the state machine reverts to the Initialization stage. N is a user-defined constant. If N hit-points are detected to be continuous, an initial state vector $\mathbf{p}$ is derived through a least-squares fit based on the first N measurements.

3. **Flexible EKF**: Measurements are initially processed by the flexible EKF to ascertain if they are outliers or signify the onset of a new entity. The filter's flexibility is controlled by adjusting the parameter noise covariance matrix $\boldsymbol{Q}$ from (4.2). If the Mahalanobis distance criterion with threshold τ_{flex} is not met, the measurements are considered to be outliers and discarded.

4. **Strict EKF**: If the flexible criterion is satisfied, the measurements are passed to the strict EKF for function parameter refinement and breakpoint detection. The strict filter operates with a lower Mahalanobis distance threshold τ_{strict} compared to the flexible filter. As the strict filter processes measurements, the state estimates evolve and gain precision. If the strict Mahalanobis distance criterion is not met at any point, a possible breakpoint is identified, and the operation switches to the least-squares refinement state - to confirm or cancel the breakpoint.

5. **Least-Squares**: An additional state is incorporated into the state-machine for further refinement using least-squares when a possible breakpoint is detected. This state allows for re-estimation of function parameters and re-evaluation of the strict Mahalanobis distance criterion. If the re-evaluated point is confirmed as a breakpoint, the aggregate is closed and the state machine goes into the reset state, a new entity is confirmed and added to the entity list; otherwise, the control is passed back to the strict filter which resumes with updated parameters.

The issue of lower accuracy state estimates from the EKFs is addresses by incorporating the least-squares methodology explored previously. Re-evaluating the hit-point aggregate at hand via least-squares before re-testing the Mahalanobis threshold is advantageous as the least-squares method presented previously tests estimates that are far from the running EKF estimate, while filtering is unable to make those kinds of large adjustments, even if the error grows due to the inaccuracies introduced with the linearization. Figure 4.18 showcases some examples of different segmentations of the same scene using the super-ellipse and line segment models. The big variation in the estimated objects from one segmentation to another is due to the tuning of the various parameters of the segmentation algorithm, including the initialization type and the chosen cost function. The influence of the Mahalanobis distance thresholds and the parameter noise covariance matrix are investigated in the next section.

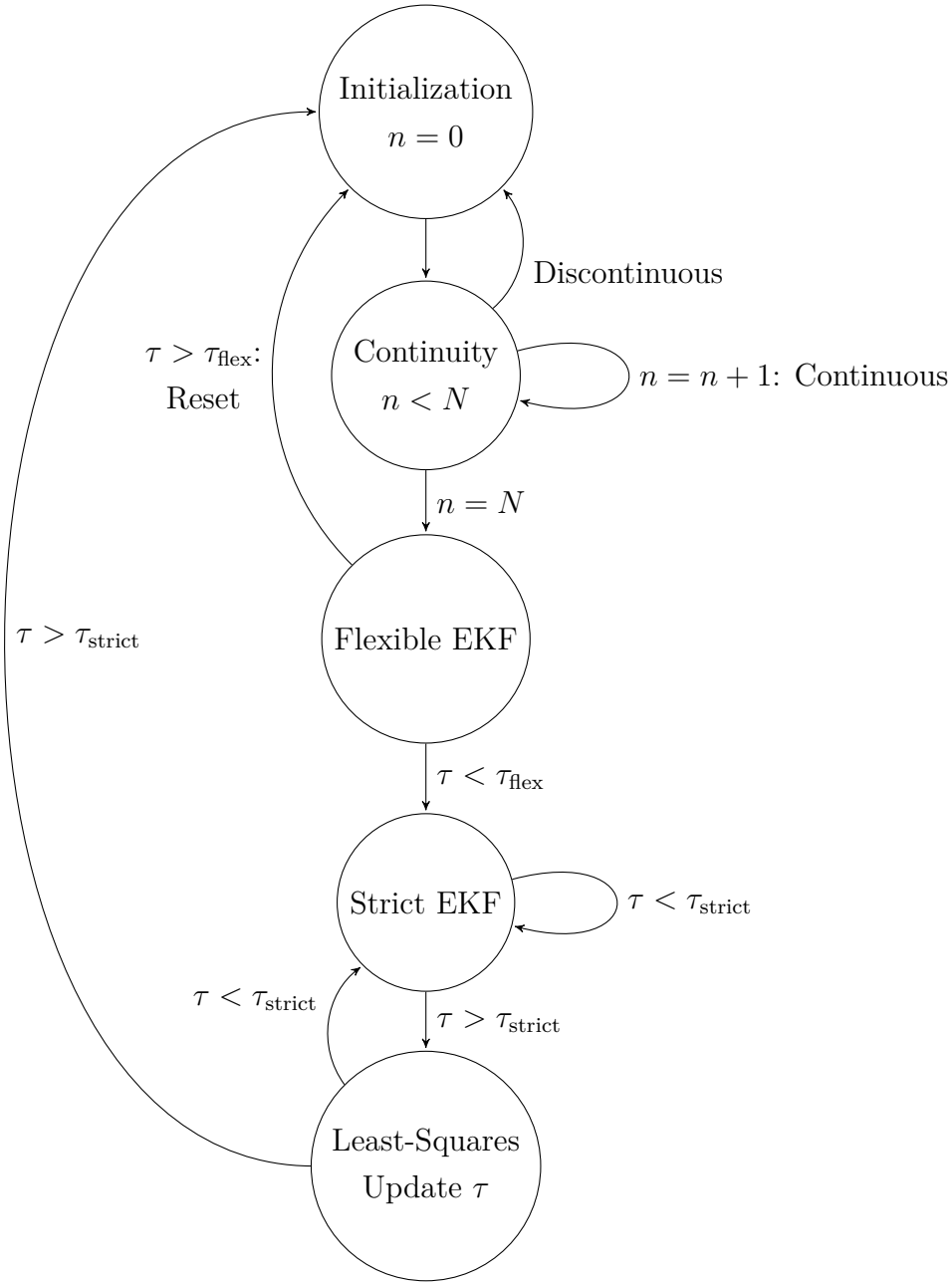
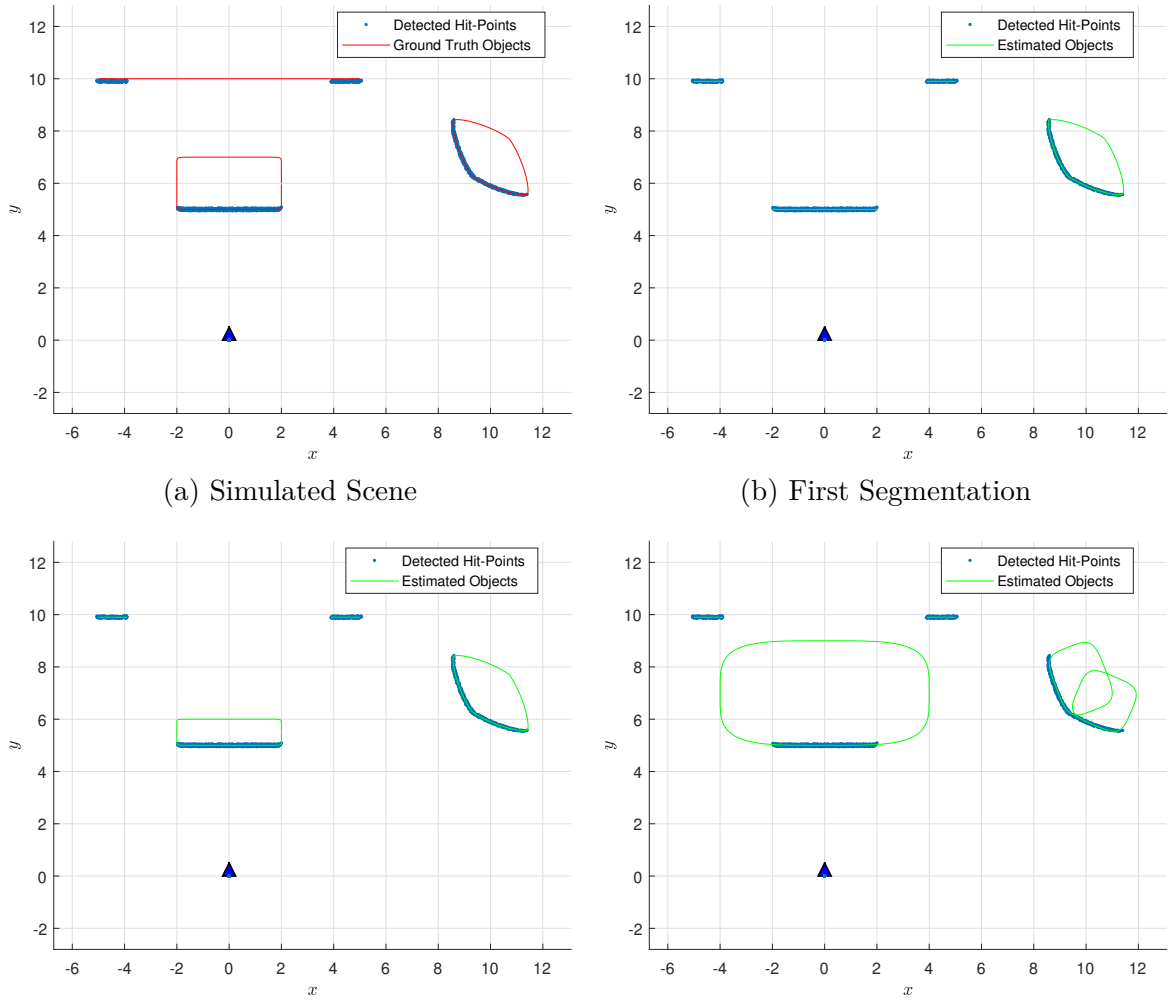


Figure 4.17 State machine representation of the dual EKF approach for segmentation.



(a) Simulated Scene

(b) First Segmentation

(c) Second Segmentation

(d) Third Segmentation

Figure 4.18 Examples of different segmentations of the same scene using super-ellipse and line segment models.

4.4.4 Performance

The performance of the segmentation algorithm is judged using only super-ellipses. Super-ellipses are harder to segment than lines or sub-lines, as that traditionally only involves using point-line distance thresholds.

Methodology and Metrics

We use the word segment to denote a group of hit-points constructed using a segmentation algorithm. In defining what constitutes a «good» segmentation for the performance analysis of the algorithm, the focus is on two aspects. First, the segmentation must accurately cap-

ture and differentiate distinct objects or features in the environment, ensuring each identified segment aligns with a unique ground truth object, or part of it. This aspect prioritizes the precision in identifying different objects, where the accuracy of segments in differentiating objects is key, irrespective of over-segmentation. Secondly, the segmentation should maintain a count of segments as close as possible to the ground truth number of distinct objects present.

As in previous performance analysis, the Monte-Carlo method of simulation is used to quantify the performance of the segmentation algorithm. In the case of the segmentation algorithm, the performance of many user-set variables needs to be adjusted; this can be made much easier with simulations. The simulation software from Section 4.2.4 is extended to include multiple objects, and a segmentation ground truth is produced. Some considerations are specified for the simulated environment to stay within research considerations and physical constraints: all assumptions previously mentioned in Section 3.2 are maintained, generated objects do not overlap, and the number of super-ellipses varies randomly within a predetermined range.

Segmentation is often performed in the context of image processing, where pixels are assigned to various objects in detection [63]. Evaluation metrics for segmentation - in simulated environments where the ground truth is readily available - aim to take into account these main factors:

- **Over-segmentation.** The estimate segments one object from the ground truth into multiple.
- **Under-segmentation.** The estimate combines multiple objects from the ground truth into one.

Over-segmentation is generally considered to be much less serious than under-segmentation, as objects can be merged easily in post-processing but under-segmentation points towards a failure in the segmentation algorithm. The most commonly used segmentation criteria in images can be categorized as being boundary-based or region-based. The latter evaluates the accuracy in terms of the number of regions, their locations and their sizes while the prior evaluates the accuracy in terms of both localization and shape accuracy of extracted boundaries [63]. A choice is made in this research to steer from image segmentation criteria towards string segmentation criteria. The string form of representing segmented data naturally stands out when dealing with an ordered set of data. This representation is illustrated in Figure 4.19.

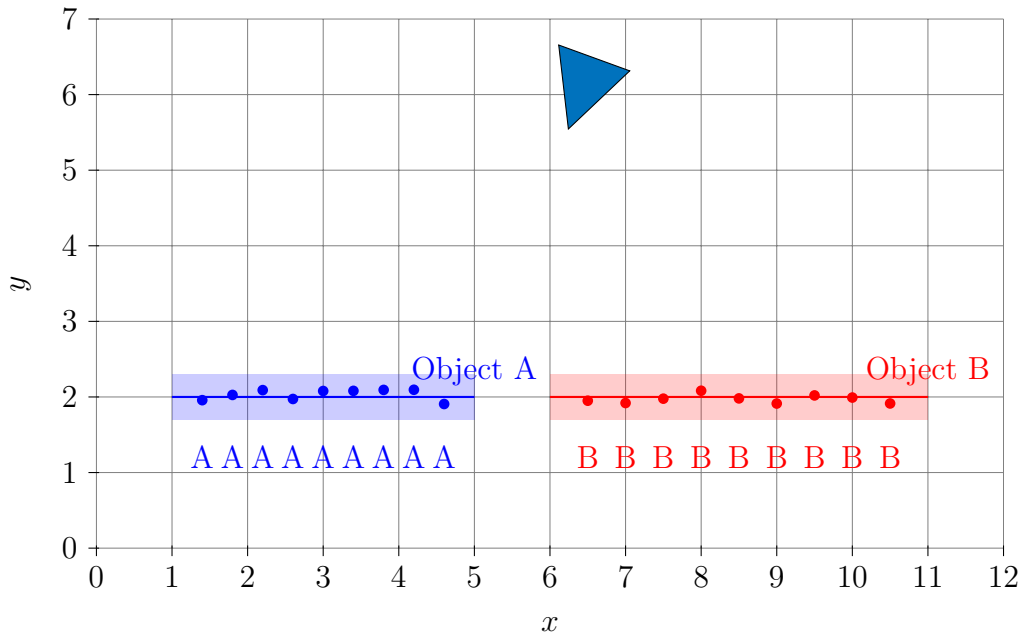


Figure 4.19 String representation of a segmentation

While these [63] various performance evaluations are applicable to the scenario at hand, a hybrid between modified Levenshtein distance and over-segmentation ratio is chosen. The ratio can be simply calculated by:

$$\text{Ratio}_{\text{seg}} = \frac{\text{Number of estimated segments}}{\text{Number of ground truth segments}}$$

The modified Levenshtein distance for segmentation is a measure of the difference between two sequences, which represent the segmentation results. It is a variant of the traditional Levenshtein distance, specifically adapted to evaluate segmentation by allowing the transformation of entire segments (or families) to match another, rather than evaluating on a point-by-point basis.

Consider two sequences of identical length $S = s_1 s_2 ... s_n$ and $T = t_1 t_2 ... t_n$, where each s_i and t_i represents the segmentation classification at the i^{th} position, and n is the total number of positions. The modified Levenshtein distance is calculated as follows in Algorithm 2:

Algorithm 2 Modified Levenshtein Distance for Segmentation

Require: Two sequences of identical length $S = s_1 s_2 \dots s_n$ and $T = t_1 t_2 \dots t_n$, where s_i and t_i represent segmentation classification at position i, and n is the total number of positions.

Ensure: The modified Levenshtein distance d_{mod} between sequences S and T.

Initialize $d = 0$ ▷ Initialize the distance
for each position i from 1 to n **do**
    if $s_i \neq t_i$ **then**
        Increment d by 1 ▷ Count initial differences
    end if
end for
for each possible transformation $T' \in \text{Transformations}(T)$ **do**
    Compute the Levenshtein distance $d(T', S)$
    Update $d = \min(d, d(T', S))$ ▷ Consider the minimum distance after transformation
end for
Normalize $\text{Ratio}_{\text{L}_{\text{dis}}} = \frac{d}{n}$
    return $\text{Ratio}_{\text{L}_{\text{dis}}}$ ▷ Return the modified Levenshtein distance ratio

We give an example with two sequences S and T aligned with each other. Each position where the segments differ is marked, and the transformation is applied to T to match S are be shown, highlighting how entire segments change as a group.

Consider the two sequences S and T:

$$S = \text{AAABBBCC}$$

$$T = \text{AABBCCCC}$$

We will calculate the modified Levenshtein distance between S and T using the algorithm.

1. Initialize the distance d to 0.
2. Compare each position i from 1 to n:
    - $S_1 = A$ and $T_1 = A \rightarrow$ No difference, $d = 0$
    - $S_2 = A$ and $T_2 = A \rightarrow$ No difference, $d = 0$
    - $S_3 = A$ and $T_3 = B \rightarrow$ Difference, $d = 1$
    - $S_4 = B$ and $T_4 = B \rightarrow$ No difference, $d = 1$

- $S_5 = B$ and $T_5 = C \rightarrow$ Difference, $d = 2$
- $S_6 = B$ and $T_6 = C \rightarrow$ Difference, $d = 3$
- $S_7 = C$ and $T_7 = C \rightarrow$ No difference, $d = 3$
- $S_8 = C$ and $T_8 = C \rightarrow$ No difference, $d = 3$

3. The initial distance d is 3.

4. Next, consider possible transformations of T to minimize the distance to S:

   - Possible transformations are:
     - Change B to A: $T' = \text{AAAAACCC}$
     - Change C to B: $T' = \text{AABBBBBB}$
   - Calculate the Levenshtein distance for each transformation:
     - For $T' = \text{AAAAACCC}$:

$$\begin{array}{cccccccccc} S & = & A & A & A & B & B & B & C & C \\ T' & = & A & A & A & A & A & C & C & C \end{array}$$

       Differences at positions 4, 5, 6, 7: distance = 4
     - For $T' = \text{AABBBBBB}$:

$$\begin{array}{cccccccccc} S & = & A & A & A & B & B & B & C & C \\ T' & = & A & A & B & B & B & B & B & B \end{array}$$

       Differences at positions 3, 7, 8: distance = 3
   - Update $d = \min(3, 4, 3) = 3$.

5. Normalize the distance:

$$\text{Ratio}_{\text{L}_{\text{dis}}} = \frac{d}{n} = \frac{3}{8} = 0.375$$

Therefore, the modified Levenshtein distance ratio between S and T is 0.375.

This modified distance is particularly useful in segmentation because it allows for the evaluation of the segmentation's correctness based on the structure of the objects being segmented, rather than just the raw number of matching and non-matching positions. It also takes into account the practical aspect of segmentation, where misclassified points that form a contiguous region can often be reclassified as a whole by combining features together in post-processing. Together, these two metrics accurately highlight the performance.

The variations are defined based on two user-set parameters: the noise covariance matrix $\boldsymbol{Q}$ and the strict Mahalanobis threshold τ_{strict}. The noise covariance matrix $\boldsymbol{Q}$ indicates the allowable distance between the prediction and the current estimated state. The diagonal of τ_{strict} represents the permissible variation for each parameter during each prediction.

The evaluation of the segmentation algorithm involves 9 scenarios, each defined by a unique combination of the parameter noise covariance matrix $\boldsymbol{Q}$ and the strict Mahalanobis threshold τ_{strict}.These variations are designed to test the algorithm's robustness and accuracy under different levels of state prediction uncertainty (represented by $\boldsymbol{Q}$) and the strictness of the segmentation threshold (represented by τ_{strict}). Table 4.4 presents these variations

Table 4.4 Segmentation Variations Based on $\boldsymbol{Q}$ and τ_{strict}

Scenario	**Description**	**Noise Covariance ($\boldsymbol{Q}$)**	$\tau_{\textbf{strict}}$
S1	Low noise, Low strictness	$0.1\boldsymbol{I}$	0.5
S2	Low noise, Medium strictness	$0.1\boldsymbol{I}$	5.0
S3	Low noise, High strictness	$0.1\boldsymbol{I}$	50.0
S4	Medium noise, Low strictness	$0.5\boldsymbol{I}$	0.5
S5	Medium noise, Medium strictness	$0.5\boldsymbol{I}$	5.0
S6	Medium noise, High strictness	$0.5\boldsymbol{I}$	50.0
S7	High noise, Low strictness	$1.0\boldsymbol{I}$	0.5
S8	High noise, Medium strictness	$1.0\boldsymbol{I}$	5.0
S9	High noise, High strictness	$1.0\boldsymbol{I}$	50.0

Each scenario in the table represents a unique combination of uncertainty in state prediction and segmentation strictness. The $\boldsymbol{Q}$ matrix values vary from low (0.1) to very high (2.0), representing increasing levels of allowed deviation between the prediction and the current estimated state. The τ_{strict} values, ranging from 0.5 to 50.0, indicate the varying levels of strictness in segment classification. The medium strictness medium noise scenarios are calibrated to ensure best performance, it is expected that they yield the best results. No parameter configuration is inherently better than another, it depends entirely on the scene, more specifically: the type, location and size of the objects in the scene. The objective of the following analysis is a portrayal and break-down of the influence of varying parameters on segmentation metrics. Understanding the reasons behind such variations guides the reader into making informed decisions when tuning parameters for a specific environment and use-case.

Analysis and Interpretation

As in Section 4.2.4, the mean values for $\text{Ratio}_{\text{seg}}$ and $\text{Ratio}_{\text{L}_{\text{dis}}}$ as well as the SEM and CI were calculated based on the Monte-Carlo simulations. These mean values serve as an estimate of the central tendency of the algorithm's performance across various scenarios.

Results

Table 4.5 Segmentation Performance Statistics for $\text{Ratio}_{\text{seg}}$ over 1000 Monte-Carlo iterations

Scenario	**Mean of $\text{Ratio}_{\text{seg}}$**	**SEM of $\text{Ratio}_{\text{seg}}$**	**CI of $\text{Ratio}_{\text{seg}}$**
S1	0.9713	0.0132	0.9713 ± 0.0259
S2	1.2124	0.0087	1.2124 ± 0.0171
S3	1.5984	0.0091	1.5984 ± 0.0178
S4	0.7932	0.0095	0.7932 ± 0.0186
S5	0.9956	0.0099	0.9956 ± 0.0194
S6	1.1923	0.0103	1.1923 ± 0.0202
S7	0.5192	0.0107	0.5192 ± 0.0210
S8	0.7829	0.0111	0.7829 ± 0.0218
S9	0.9881	0.0115	0.9881 ± 0.0225

Observing the mean values of $\text{Ratio}_{\text{seg}}$, in Table 4.5 we notice several key trends:

- **Low Noise (S1, S2, S3):** As the strictness increases from S1 to S3, the mean of $\text{Ratio}_{\text{seg}}$ rises from 0.9713 to 1.5984. This indicates that as the segmentation becomes stricter, the algorithm tends to over-segment, dividing ground truth objects into more segments than necessary. This is most noticeable in S3, where the mean $\text{Ratio}_{\text{seg}}$ is significantly above 1, indicating a high degree of over-segmentation.
- **Medium Noise (S4, S5, S6):** Here, the trend is less clear but still points towards over-segmentation with increasing strictness. S4 starts with a mean $\text{Ratio}_{\text{seg}}$ of 0.7932, which is under-segmented. However, by S6, the ratio increases to 1.1923, moving towards slight over-segmentation.
- **High Noise (S7, S8, S9):** With high noise, the algorithm tends to under-segment significantly, as seen in S7 with a mean $\text{Ratio}_{\text{seg}}$ of 0.5192. As strictness increases, this under-segmentation decreases, with S9 nearly reaching a balanced segmentation with a mean $\text{Ratio}_{\text{seg}}$ of 0.9881.

The variations in $\text{Ratio}_{\text{seg}}$ demonstrate the sensitivity of the segmentation algorithm to noise and strictness parameters. Note that these parameters tend to cancel each other out, S1 is similar to S5 and S9. This can be inferred with the following understanding: **Lower noise and higher strictness generally lead to fewer predicted segments, independently from each other.**

The standard error of the mean (SEM) and the confidence intervals (CI) for $\text{Ratio}_{\text{seg}}$ suggest that the measurements are precise and the variance is reasonably controlled across the simulations. For example, in S1, the SEM is 0.0132 and the CI is ± 0.0259 around the mean, indicating tight clustering of the results.

Table 4.6 Segmentation Performance Statistics for $\text{Ratio}_{\text{L}_{\text{dis}}}$ over 1000 Monte-Carlo iterations

Scenario	**Mean of $\text{Ratio}_{\text{L}_{\text{dis}}}$**	**SEM of $\text{Ratio}_{\text{L}_{\text{dis}}}$**	**CI of $\text{Ratio}_{\text{L}_{\text{dis}}}$**
S1	0.1298	0.0087	0.1298 ± 0.0171
S2	0.0318	0.0089	0.0318 ± 0.0174
S3	0.0576	0.0092	0.0576 ± 0.0180
S4	0.2168	0.0094	0.2168 ± 0.0184
S5	0.0122	0.0097	0.0122 ± 0.0190
S6	0.0289	0.0100	0.0289 ± 0.0196
S7	0.4782	0.0102	0.4782 ± 0.0200
S8	0.1983	0.0105	0.1983 ± 0.0206
S9	0.0121	0.0108	0.0121 ± 0.0212

Continuing the analysis with the performance metrics for the modified Levenshtein distance $\text{Ratio}_{\text{L}_{\text{dis}}}$ in Table 4.6, we can draw further insights into the segmentation quality. From the mean values of $\text{Ratio}_{\text{L}_{\text{dis}}}$ across the scenarios, several observations can be made:

- **Low Noise (S1, S2, S3):** The mean $\text{Ratio}_{\text{L}_{\text{dis}}}$ values decrease as strictness increases, with S1 having a mean of 0.1298 and S3 a mean of 0.0576. This trend suggests that higher strictness, despite causing over-segmentation as seen for $\text{Ratio}_{\text{seg}}$, actually helps in aligning the segments more closely with the ground truth boundaries, thereby reducing the Levenshtein distance. An example of over-segmentation can be seen in Figure 4.18d, where the upper right diamond-shaped object in the ground truth is segmented as two rectangular shaped objects.

- **Medium Noise (S4, S5, S6):** S4 starts with a relatively high $\text{Ratio}_{\text{L}_{\text{dis}}}$ mean of 0.2168, indicating poor segment matching with the ground truth. As strictness increases, the mean $\text{Ratio}_{\text{L}_{\text{dis}}}$ significantly drops to 0.0122 by S5 and slightly rises to 0.0289 by S6. This indicates an improvement in segmentation quality as the segments better match the ground truth.
- **High Noise (S7, S8, S9):** High noise scenarios show a marked improvement in segment matching as strictness increases. S7 has a high mean $\text{Ratio}_{\text{L}_{\text{dis}}}$ of 0.4782, indicating significant mismatch with the ground truth. This mismatch is drastically reduced by increasing strictness, as seen in S9, where the mean $\text{Ratio}_{\text{L}_{\text{dis}}}$ reaches 0.0121, close to the optimal performance seen in medium noise scenarios.

The relationship between $\text{Ratio}_{\text{seg}}$ and $\text{Ratio}_{\text{L}_{\text{dis}}}$ is validated here. **Scenarios that tend towards over-segmentation (higher $\text{Ratio}_{\text{seg}}$) generally have lower $\text{Ratio}_{\text{L}_{\text{dis}}}$**, indicating that while the segments are more numerous than the ground truth, they are correctly capturing the shape and location of the true segments. Conversely, **under-segmentation (lower $\text{Ratio}_{\text{seg}}$) leads to higher $\text{Ratio}_{\text{L}_{\text{dis}}}$**, showing that capturing the shape and location of the true segments is impossible with the under-segmented segmentations.

The standard error of the mean (SEM) and the confidence intervals (CI) for $\text{Ratio}_{\text{L}_{\text{dis}}}$ again show that the data points are tightly clustered around the mean, indicating stable performance across the Monte-Carlo simulations. For instance, S5 has an SEM of 0.0097 and a CI of $\pm$ 0.0190.

In summary, the choice between focusing on $\text{Ratio}_{\text{seg}}$ or $\text{Ratio}_{\text{L}_{\text{dis}}}$ is highly environment-specific. Additionally, in scenarios where capturing the exact shape and location of segments is critical, higher strictness levels (leading to higher $\text{Ratio}_{\text{seg}}$ but lower $\text{Ratio}_{\text{L}_{\text{dis}}}$) may be preferred. Conversely, if over-segmentation is less desirable and fewer, more precise segments are required, lower strictness levels (resulting in lower $\text{Ratio}_{\text{seg}}$) might be more appropriate despite the potential for higher $\text{Ratio}_{\text{L}_{\text{dis}}}$. The optimal balance depends on the specific requirements and noise levels of the application environment.

4.5 Conclusion

In this chapter, the exploration of single-scan segmentation and parameter estimation for LiDAR data using implicit functions, specifically super-ellipses and lines, was comprehensively detailed. The goal was to explore methodologies for fitting these geometric shapes to raw

scan points for map feature extraction from single LiDAR scans.

The first key insight was the complexity introduced by super-ellipses, which extends the basic ellipse model, allowing the model to represent a wider variety of shapes. The difficulty in estimating parameters for super-ellipses was tackled using a series of Monte-Carlo experiments to quantify fitting performance and through the application of various cost functions designed to minimize geometric distance between scan points and the curve. Among these, the robust cost function and the area minimizing cost function in Section 4.2 proved effective.

Furthermore, initialization considerations for parameter estimation were explored, emphasizing the importance of proper initialization to avoid local minima. Multiple initialization strategies using PCA-based estimates and centroid shifts, were implemented to enhance the performance of the parameter estimation process. This multi-initialization approach significantly improved the results of the parameter estimation process.

For segmentation, an EKF-based framework was used for simultaneous segmentation and parameter estimation. This is model-agnostic, capable of handling various implicit functions beyond super-ellipses and lines. The dual EKF strategy, incorporating both flexible and strict filters, enabled balancing initial flexibility with subsequent precision. This method effectively addressed the challenge of distinguishing between contiguous hit-points belonging to different objects and those forming part of a single object.

The performance of the segmentation algorithm was rigorously evaluated through Monte-Carlo simulations, considering the ratio of estimated segments to ground truth segments ($\text{Ratio}_{\text{seg}}$) and a modified Levenshtein distance ($\text{Ratio}_{\text{L}_{\text{dis}}}$) as metrics. The results provided valuable insights into the trade-offs between segment count and segment accuracy, guiding parameter tuning for specific environmental conditions.

CHAPTER 5 MULTI-SCAN FEATURE REFINEMENT

5.1 Introduction

In this chapter, multi-scan feature refinement is addressed. Ideally, for the attainment of the research objectives set forth at the beginning of this work, a minimal amount of hit-points would be kept from the scans as they come in. To this end, the chapter will first delve into methods for associating new hit-points to existing features, revisiting the concept of Mahalanobis distance in the context of the hit-point to feature association problem. Following this, attention is directed towards algorithms aimed at improving features based on newly associated hit-points, including gradient descent applied to a non-linear objective function, an iterative Kalman filter method, and strategies for mitigating the loss of information. Finally, the chapter presents a comprehensive overview of the global architecture used for multi-scan feature refinement and concludes with results that compare the developed methods with the state of the art.

5.2 Associating New Hit-points to Existing Features

This section explores the methodologies for associating incoming hit-points with existing map features. Associating new hit-points accurately is crucial for effective feature refinement and maintaining the integrity of the map.

5.2.1 The Hit-point to Feature Association Problem

The feature association problem in mapping involves determining whether a newly acquired hit-point from a LiDAR scan corresponds to an existing feature in the map. Mathematically, this problem can be formalized as follows: Given a set of existing features $\{F_1, F_2, \ldots, F_n\}$ in the map, each characterized by a set of parameters $\boldsymbol{p}_i$ and a hit-point $\boldsymbol{x}_{\text{new}}$ obtained from the latest LiDAR scan, the task is to ascertain which, if any, of the existing features F_i the hit-point $\boldsymbol{x}_{\text{new}}$ belongs to.

This decision is based on the proximity of $\boldsymbol{x}_{\text{new}}$ to each feature F_i, taking into account the feature's geometric characteristics and parameter uncertainties. The challenge is to establish a robust criterion that accurately associates $\boldsymbol{x}_{\text{new}}$ with the correct feature, despite the presence of measurement noise and uncertainties in feature estimation.

The association problem can be formulated as:

$$\text{Associate } \boldsymbol{x}_{\text{new}} \text{ with } F_i \text{ if } \underset{i}{\operatorname{argmin}}\, D(\boldsymbol{x}_{\text{new}}, F_i) < \tau$$

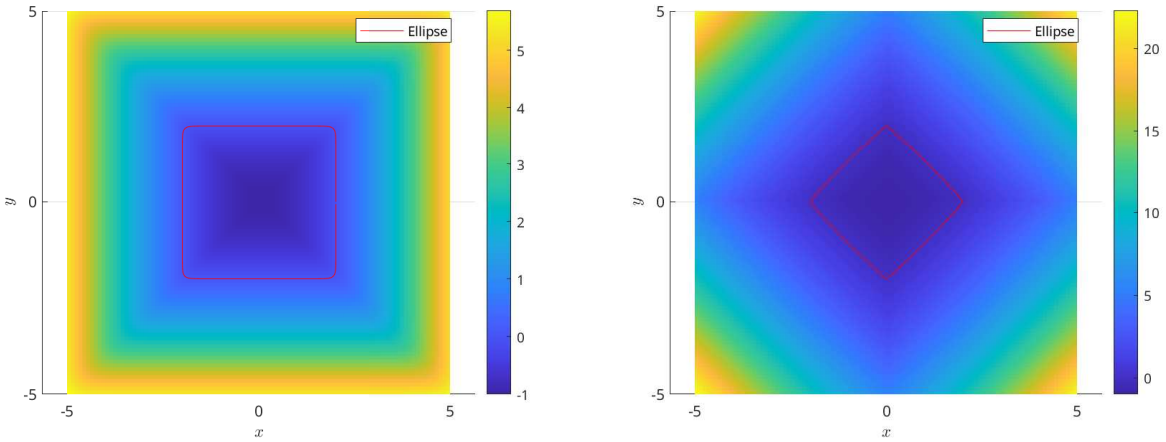
where D is a distance metric measuring the discrepancy between the hit-point and the feature, and τ is a predefined threshold.

The distance metric D should be designed to account for various aspects, such as the geometrical shape of the feature, the uncertainties in its estimated parameters according to the pose of the robot, and the measurement noise. This requires more than just a simple geometrical distance; it necessitates a statistical measure that can effectively handle the uncertainties and ambiguities in the data.

5.2.2 The Mahalanobis Distance Revisited

The Mahalanobis distance has already been introduced in 4.4.2 as a means of classifying individual LiDAR measurements as belonging or not to an existing feature according to pre-determined criteria. To address the unique challenges posed by multi-scan hit-point association, the Mahalanobis distance is revisited and adapted to the context. In the dual EKF architecture, the Mahalanobis distance serves as a threshold for hit-point association, but its application in multi-scan scenarios raises certain issues. Specifically, for super-ellipse models, the Mahalanobis distance can become highly divergent in specific regions due to the inclusion of the implicit function in its equation (4.6).

For super-ellipses that are closer to rectangular shapes ($\epsilon \to 0$), the cost function (in our case the robust cost function defined in Section 4.2.2) in the Mahalanobis distance tends to create extreme strips in an $\times$ shape. For super-ellipses that are closer to diamond shapes ($\epsilon \to 2$), these strips manifest in a + (cross) shape, as illustrated in Figures 5.1a and 5.1b. The extreme values generated along these strips can lead to unintended hit-point associations that deviate significantly from the actual feature.



(a) $\epsilon = 0.1$ (b) $\epsilon = 1.9$

Figure 5.1 Residuals for $\mathcal{F}_s^\epsilon - 1$

This issue is particularly problematic in the context of multi-scan hit-point association - where the hit-points from multiple scans are to be cohesively integrated - it is less likely to occur during the initial segmentation step, where objects are first detected, due to the fact that there are two prior geometric thresholds in the continuity test. Thus the extreme regions of the Mahalanobis distance did not clearly manifest an impact in that aspect of the proposed solution.

The only viable solution is to set a relatively low threshold for association, typically below 0.1. While this approach increases the likelihood of generating multiple estimated entities for a single ground truth object, it effectively prevents degeneration caused by incorrect point associations. Such degeneration arises when the algorithm calculates excessively high uncertainty matrices for estimates based on erroneous associations, thereby perpetuating a cycle of parameter adjustments and further erroneous associations.

5.3 Improving Features with Associated Hit-points

The process of feature improvement over multiple scans is done once hit-point association has been completed. We define $\tilde{\mathbf{M}}_{k,o}$ as the matrix formed from measurements that have

hit-points which been categorized as belonging to a specific object o at timestep k.

$$\tilde{\mathbf{M}}_{k,o} = \begin{bmatrix} \tilde{\mathbf{m}}_{k,1} \\ \tilde{\mathbf{m}}_{k,2} \\ \dots \end{bmatrix}$$

With $\mathbf{m}_{k,i}$ defined in (3.1). The relationship between measurements and hit-points remains (3.2). This set $\tilde{\mathbf{M}}_{k,o}$ is assumed to be associated with a specific feature characterized by an implicit function type $\hat{c}_{k,o}$.

Beyond the set $\tilde{\mathbf{M}}_{k,o}$, information about the feature's previously estimated parameters is available, $\hat{\boldsymbol{p}}_{k-1,o}$, and its covariance $\mathbf{P}_{k-1,o}$. The objective is to refine $\mathbf{P}_{k-1,o}$ and $\hat{\boldsymbol{p}}_{k,o}$ using $\tilde{\mathbf{M}}_{k,o}$, $\hat{\boldsymbol{p}}_{k-1,o}$, and $\Sigma_{k-1,o}$, aiming for a more accurate representation of the object o using the newly acquired and associated hit-points.

5.3.1 The Feature refinement Problem

Given a prior estimate of the state vector $\hat{\mathbf{p}}_{k-1}$ with its associated error covariance $\boldsymbol{P}_{k-1}$ for the feature o that uses an implicit function $\mathcal{F}$, at the beginning of the k^{th} laser scan measurement, the feature refinement uses the prior known attributes of the feature as well as all measurements associated to the feature $\tilde{\mathbf{M}}_{k,o}$, at the beginning of scan k. The associated cost function is as follows:

$$\mathcal{L}_k(\hat{\mathbf{p}}_k, \hat{\mathbf{M}}_{k,o}) = \left(\hat{\mathbf{M}}_{k,o} - \tilde{\mathbf{M}}_{k,o}\right)^T \boldsymbol{W} \left(\hat{\mathbf{M}}_{k,o} - \tilde{\mathbf{M}}_{k,o}\right) + (\hat{\mathbf{p}}_k - \hat{\mathbf{p}}_{k-1})^T (\boldsymbol{P}_{k-1})^{-1} (\hat{\mathbf{p}}_k - \hat{\mathbf{p}}_{k-1}) \tag{5.1}$$

With the minimization problem being:

$$\min_{\hat{\mathbf{p}}_k, \hat{\mathbf{M}}_{k,o}} ||\mathcal{L}_k(\hat{\mathbf{p}}_k, \hat{\mathbf{M}}_{k,o})||^2 \tag{5.2}$$

subject to the non-linear constraint:

$$\mathcal{F}\left(L(\boldsymbol{\Xi}_k, \hat{\mathbf{M}}_{k,o}), \hat{\mathbf{p}}_k\right) - 1 = 0 \tag{5.3}$$

Linearizing the Feature refinement Problem

As done by T. Dang [64], the problem is linearized by expanding the constraint function $\mathcal{F}\left(L(\boldsymbol{\Xi}_k, \hat{\mathbf{M}}_{k,o}), \hat{\mathbf{p}}_k\right)$ about a nominal point $(\hat{\mathbf{p}}_{k-1}, \tilde{\mathbf{m}}_k)$. It is then approximated by its first-order Taylor series expansion, which involves calculating the Jacobian matrices of partial derivatives with respect to both $\hat{\mathbf{p}}_k$ and $\hat{\mathbf{M}}_{k,o}$. This linearization allows to transform the non-linear constraint into a linear one, thus simplifying the minimization problem into a quadratic programming problem that can be solved with standard optimization techniques. The solution to the linearized problem provides an update to the estimate of the state vector $\hat{\mathbf{p}}_k$ and the feature measurements $\hat{\mathbf{M}}_{k,o}$, which minimizes the cost function $\mathcal{L}_k$ under the linearized constraint.

The constraint is therefore linearized about the operation point $(\hat{\mathbf{p}}_{k-1}, \tilde{\mathbf{m}}_k)$:

$$\mathcal{F}\left(L(\boldsymbol{\Xi}_k, \hat{\mathbf{M}}_{k,o}), \hat{\mathbf{p}}_k\right) \approx \boldsymbol{H}_k\left(\hat{\mathbf{p}}_k - \hat{\mathbf{p}}_{k-1}\right) + \boldsymbol{B}_k\left(\hat{\mathbf{M}}_{k,o} - \tilde{\mathbf{m}}_k\right) + \mathcal{F}\left(L(\boldsymbol{\Xi}_k, \hat{\mathbf{M}}_{k,o}), \hat{\mathbf{p}}_{k-1}\right) \quad (5.4)$$

with the Jacobian w.r.t. the parameters $\mathbf{H}$ defined in (4.3) but evaluated at the operation point and the Jacobian w.r.t. the measurements $\mathbf{B}$ defined in (4.4.1). The operation point is set according to our best knowledge of the true parameters and observations, and thus is initialized at $\left(\hat{\mathbf{p}}_{k-1}, \tilde{\mathbf{M}}_{k,o}\right)$. The linearized cost function with Lagrange multipliers for the feature refinement problem can be written as follows:

$$\begin{aligned}\mathcal{J}_k(\hat{\mathbf{p}}_k, \hat{\mathbf{M}}_{k,o}, \boldsymbol{\lambda}) =& \mathcal{L}_k(\hat{\mathbf{p}}_k, \hat{\mathbf{M}}_{k,o}) \\ &- \boldsymbol{\lambda}^T\left(\boldsymbol{H}_k\left(\hat{\mathbf{p}}_k - \hat{\mathbf{p}}_{k-1}\right) + \boldsymbol{B}_k\left(\hat{\mathbf{M}}_{k,o} - \tilde{\mathbf{m}}_k\right) + \mathcal{F}\left(L(\boldsymbol{\Xi}_k, \hat{\mathbf{M}}_{k,o}), \hat{\mathbf{p}}_{k-1}\right) - 1\right)\end{aligned} \quad (5.5)$$

Here, $\mathcal{J}_k$ represents the refined cost function that includes the original cost function $\mathcal{L}_k$ and the linearized constraint weighted by the Lagrange multiplier $\boldsymbol{\lambda}$. This formulation allows for the simultaneous minimization of the cost function while ensuring that the linearized constraint is satisfied. The Lagrange multiplier $\boldsymbol{\lambda}$ adjusts the weight given to the constraint in the optimization process. This approach is commonly used in constrained optimization problems to find a solution that balances the objective function and the constraints.

5.3.2 An IEKF to Solve the Feature Refinement Problem

The recursive least-squares formulated in Section 5.3.1 is addressed using an IEKF (Iterative Extended Kalman Filter). This involves iteratively refining the estimates of state vector $\hat{\mathbf{p}}_k$ and feature measurements $\hat{\mathbf{M}}_{k,o}$ based on the linearized model of the constraints. The IEKF approach was done by Dang et al. [64], it allows for an effective handling of nonlinearities present in the feature refinement problem by iteratively updating the state and measurement estimates. The process begins by using the Lagrange multipliers to solve for $\hat{\mathbf{p}}_k$ and $\hat{\mathbf{M}}_{k,o}$:

$$\begin{aligned}\hat{\mathbf{p}}_k &= \hat{\mathbf{p}}_{k-1} + \boldsymbol{P}_{k-1}\boldsymbol{H}_k^T\boldsymbol{\lambda} \\ \hat{\mathbf{M}}_{k,o} &= \tilde{\mathbf{M}}_{k,o} + \boldsymbol{W}^{-1}\boldsymbol{B}_k^T\boldsymbol{\lambda}\end{aligned}$$

With the constraints:

$$\begin{aligned}\boldsymbol{z} = & \\ & \boldsymbol{H}_k\hat{\mathbf{p}}_k + \boldsymbol{B}_k\hat{\mathbf{M}}_{k,o} + \mathcal{F}\left(L(\boldsymbol{\Xi}_k, \tilde{\mathbf{M}}_{k,o}), \hat{\mathbf{p}}_{k-1}\right) - 1 \\ & = 0\end{aligned}$$

It's then possible to extract the corrected observations and state parameters:

$$\begin{aligned}\hat{\mathbf{p}}_k &= \hat{\mathbf{p}}_{k-1} + \boldsymbol{P}_{k-1}\boldsymbol{H}_k^T(\boldsymbol{H}_k\boldsymbol{P}_{k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}\left(\boldsymbol{z} - \boldsymbol{H}_k\tilde{\mathbf{M}}_{k,o}\right) \\ \hat{\mathbf{M}}_{k,o} &= \tilde{\mathbf{M}}_{k,o} + \boldsymbol{W}^{-1}\boldsymbol{B}_k^T(\boldsymbol{B}_k\boldsymbol{W}^{-1}\boldsymbol{B}_k^T + \boldsymbol{R}_k)^{-1}\left(\boldsymbol{z} - \boldsymbol{H}_k\tilde{\mathbf{M}}_{k,o}\right)\end{aligned} \tag{5.6}$$

with $\boldsymbol{R}_k$ defined in (4.5) and $\boldsymbol{J}_k$ defined in (4.4). The Kalman gain is then defined as:

$$\boldsymbol{K}_k = \boldsymbol{P}_{k-1}\boldsymbol{H}_k^T\left(\boldsymbol{H}_k\boldsymbol{P}_{k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k\right)^{-1} \tag{5.7}$$

and the covariance update:

$$\boldsymbol{P}_k^+ = \left(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k\right)\boldsymbol{P}_{k-1}\left(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k\right)^T + \boldsymbol{K}_k\boldsymbol{R}_k\boldsymbol{K}_k^T \tag{5.8}$$

as well as the Lagrangian multiplier update:

$$\lambda^+ = \lambda^- + \alpha\left(\boldsymbol{H}_k\hat{\mathbf{p}}_k + \boldsymbol{B}_k\hat{\mathbf{M}}_{k,o} + \mathcal{F}(L(\boldsymbol{\Xi}_k, \tilde{\mathbf{M}}_{k,o}), \hat{\mathbf{p}}_{k-1}) - 1\right) \tag{5.9}$$

where α is a step size parameter, and the prediction step:

$$\hat{\mathbf{p}}_k = f(\hat{\mathbf{p}}_k, \mathbf{u}_k) \tag{5.10}$$

$$\tilde{\mathbf{m}}_{k+1} = \mathbf{m}_k \tag{5.11}$$

$$\boldsymbol{P}_k^- = \boldsymbol{F}_k \boldsymbol{P}_k^+ \boldsymbol{F}_k^T + \boldsymbol{G}_k \boldsymbol{Q}_k \boldsymbol{G}_k^T \tag{5.12}$$

The iterative algorithm is outlined below:

1. **Initialization:** Set the initial estimates $\hat{\mathbf{p}}_0$ and $\tilde{\mathbf{m}}_0$ based on available data. Initialize the covariance matrix $\boldsymbol{P}_0$.
2. **Prediction Step:** Predict the next state $\hat{\mathbf{p}}_k$ and measurements $\tilde{\mathbf{m}}_{k+1}$ using the state transition (5.10) and measurement functions.
3. **Update Step:**
   - Compute the Jacobians $\boldsymbol{H}_k$ and $\boldsymbol{B}_k$ at the operation point.
   - Linearize the constraint and set up the refined cost function $\mathcal{J}_k$.
   - Solve the optimization problem to find the updated estimates $\hat{\mathbf{p}}_k$ and $\hat{\mathbf{M}}_{k,o}$ using (5.6).
   - Update the Kalman gain $\boldsymbol{K}_k$ from (5.7), the covariance matrix $\boldsymbol{P}_k^+$ from (5.8), and the Lagrangian multiplier λ from (5.9).
4. **Correction Step:** Correct the state estimate and the covariance matrix with the updated values.
5. **Iterate:** Repeat the prediction and update steps for each time step k.

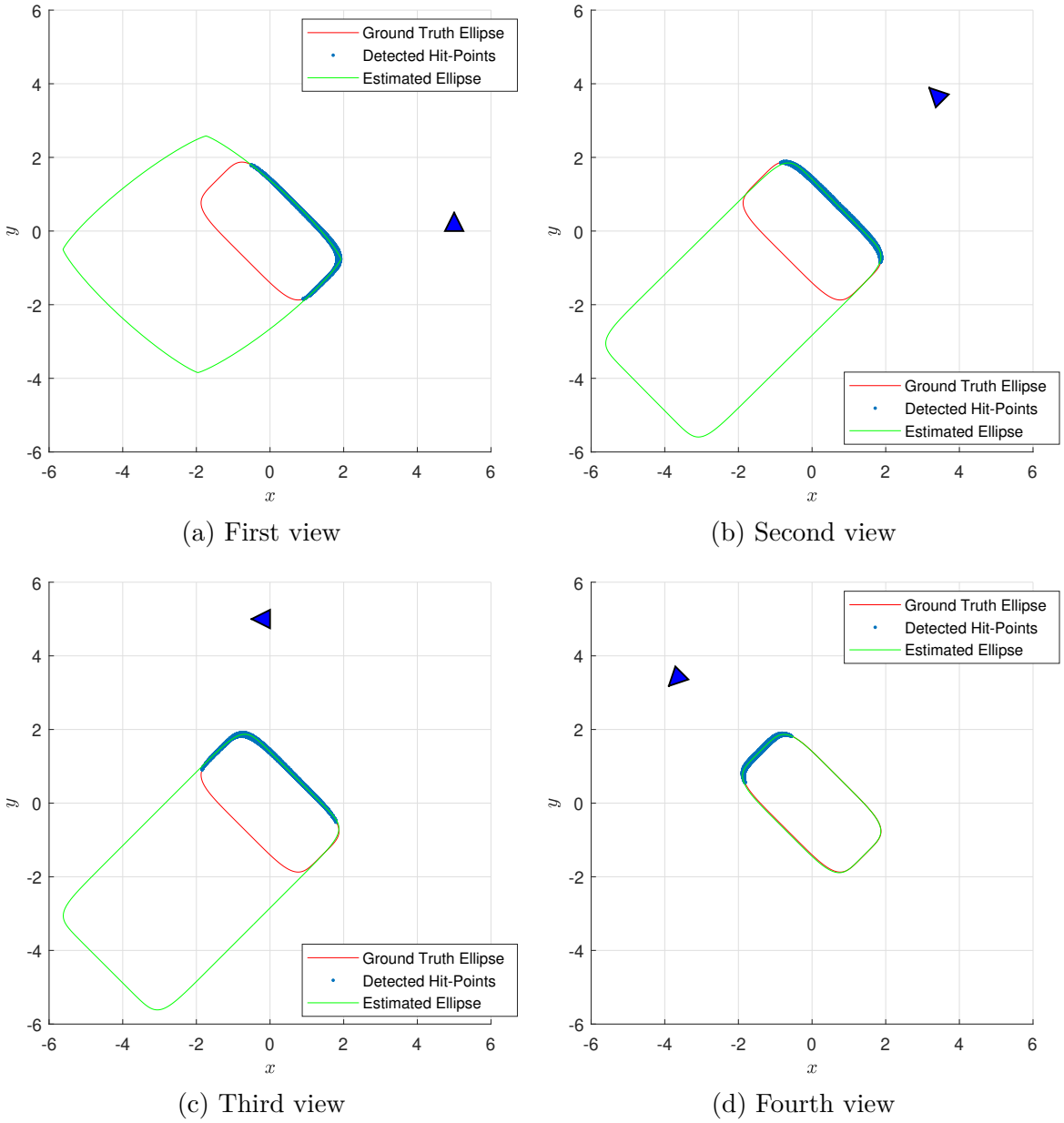
5.3.3 Least-Squares Minimization to Solve the Feature refinement Problem

Here, the focus shifts from the iterative approach of the IEKF to directly solving the least-squares problem using non-linear optimization. The non-linear optimization problem involves minimizing the cost function $\mathcal{L}_k$ under the non-linear constraint. In the least-squares minimization framework, the non-linear constraints are integrated into the optimization equation. This is done using a Lagrange multiplier $\boldsymbol{\lambda}$, as previously. However the constraints are not linearized, the refined cost function can be expressed as:

$$\mathcal{L}'_k(\hat{\mathbf{p}}_k, \hat{\mathbf{M}}_{k,o}, \boldsymbol{\lambda}) = \mathcal{L}_k(\hat{\mathbf{p}}_k, \hat{\mathbf{M}}_{k,o}) + \boldsymbol{\lambda} \left(\mathcal{F} \left(L(\boldsymbol{\Xi}_k, \hat{\mathbf{M}}_{k,o}), \hat{\mathbf{p}}_k \right) - 1 \right)^2 \tag{5.13}$$

This is minimized using non-linear optimization techniques. *Ceres* [58], as used previously, is used to iteratively adjusts $\hat{\mathbf{p}}_k$ and $\hat{\mathbf{M}}_{k,o}$ to find the minimum of $\mathcal{L}'_k$, subject to the constraint being satisfied. The Langrangian multiplier value λ is set dynamically by *Ceres* when

its multiplier is configured as a constraint function. A working example of the algorithm described above is portrayed in Figure 5.2.



(a) First view

(b) Second view

(c) Third view

(d) Fourth view

Figure 5.2 Sequential scans of a rectangular object from different angles.

5.3.4 Information Decay in the Covariance

Covariance decay is particularly prevalent in systems characterized by non-linear observation and state propagation models, as detailed in Section 4.4.1. The iterative nature of these methods, coupled with the necessity of linearizing non-linear models, often leads to a gradual

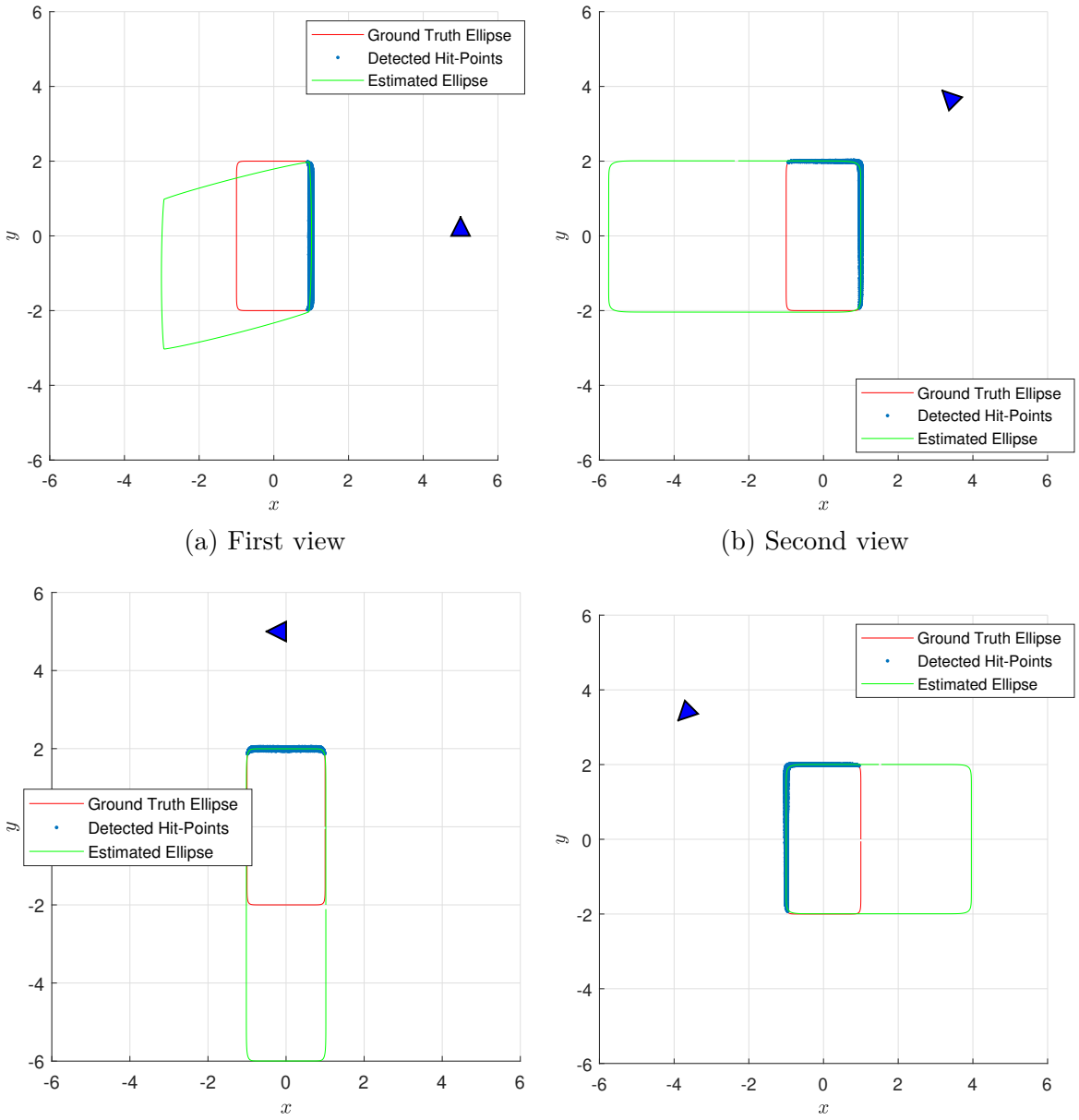
reduction in the representational accuracy of the covariance matrix. In the following paragraphs, we investigate the causes, evidence, and mitigation strategies related to information decay in the covariance matrix in the present context.

Evidence of Information Decay

Empirical observations in certain scenarios have distinctly highlighted the phenomenon of information decay in the estimation of object shapes by a robotic system. These cases are illustrated to demonstrate the limitations of relying solely on covariance information from previous scans, particularly when the robot's perspective or the object's assumption changes significantly between observations.

In one instance, depicted in Figure 5.3, the robot initially detects a rectangular obstacle from an angle showing an edge and two faces of the rectangle. Subsequent scans occur as the robot moves to a position where only one face is visible. The previous scans' covariance, providing error estimates on the center coordinates (x_c, y_c) and the radius parameters (a, b), is inadequate for the next estimation phase. Adjustments to the center position and radius parameters are necessary to accurately expand the perceived shape along the visible face. However, this level of detail is not encapsulated in the covariance matrix. Consequently, the shape's a-posteriori minimization moves the estimated shape towards the newly detected points, with minimal growth and a loss of information regarding the previously observed edge.

Another case, shown in Figure 5.4, involves the robot initially perceiving the edge of an object, leading to an erroneous estimation of the shape as a rectangle, whereas it is actually a diamond. As the robot navigates around the object, it begins to detect hit-points indicative of the diamond shape. However, due to the substantial difference between the previous rectangle parameters and the new diamond parameters needed for accurate representation, the system struggles to adapt. This scenario underscores how the initial hit-point information (the edge) that suggested a rectangular shape is no longer present in the form of parameter covariance. The subsequent estimation effectively becomes a re-evaluation, as the existing error covariance fails to convey the necessary parameter adjustments.



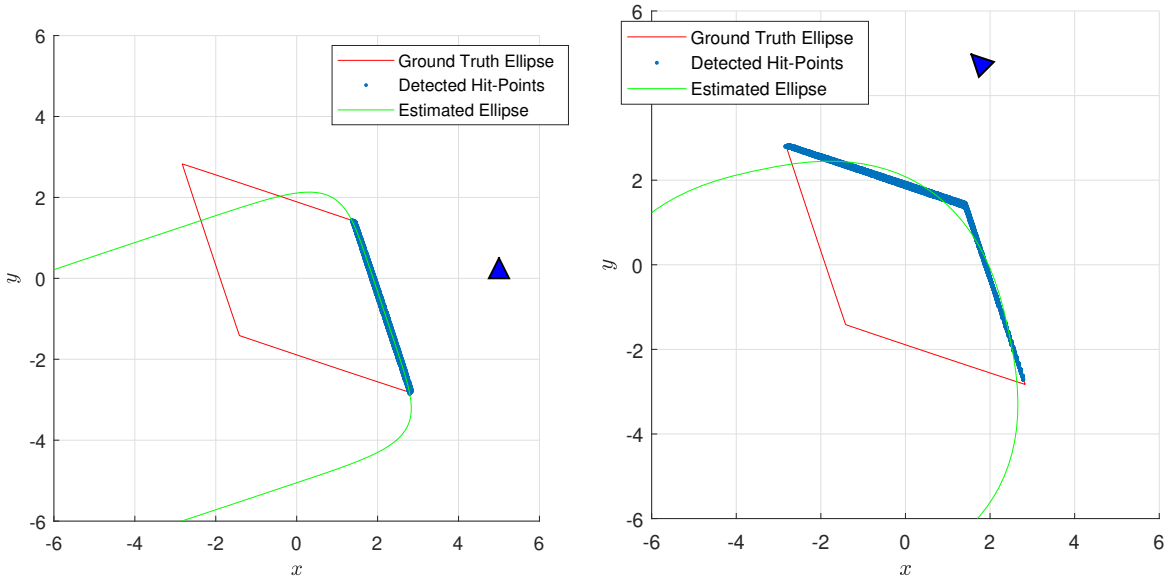
(a) First view

(b) Second view

(c) Third view

(d) Fourth view

Figure 5.3 Sequential scans of a rectangular object from different angles.



(a) First view

(b) Second view

(c) Third view

(d) Fourth view

Figure 5.4 Sequential scans of a diamond-shaped object from different angles.

Using covariance from previous scans as an measure of how much each parameter should be allowed to change without incorporating additional contextual or geometrical information leads to a decay of relevant information. For the case shown in Figure 5.3, not enough importance is given to prior estimates and the system is unable to keep the information it had. For the case shown in Figure 5.4, the opposite is true, too much importance is given to prior estimates and the system is unable to innovate enough to adjust the estimate using the

new information. Section 5.3.4 details why it's impossible to find system parameters that can solve these issues.

Cause for Expecting Information Decay

When working with systems governed by non-linear observation and state propagation models as described in Section 4.4.1, a known challenge is the gradual reduction of the representational accuracy of covariance matrices [64]. The linearization process, a necessary step for applying these non-linear models, inherently approximates the system's behavior. This approximation, represented by the Jacobian matrices in (4.3) and (4.4), is strictly valid only near the point of linearization. The Jacobian matrices are computed using the *autodiff* library [65]. As the state estimate evolves, the fidelity of this linear approximation to the actual non-linear dynamics diminishes.

Differences between these theoretical models and the actual system behavior introduce discrepancies in the estimation. These discrepancies are pronounced in complex real-world systems where modeling the dynamics and observations with complete accuracy is challenging.

The Gaussian distribution assumption for states, measurements, and noises, while simplifying the mathematical formulation, may not adequately represent the statistical nature of non-linear systems. Non-linear systems often exhibit distributions with characteristics that differ significantly from the Gaussian model. This discrepancy results in the covariance matrix's gradual depletion in accurately encapsulating the state's uncertainty, as the system's true state increasingly diverges from these Gaussian-based assumptions.

Furthermore, the actual prediction errors in dynamic environments may not be consistently captured by a static weighting matrix, as this matrix is often formulated based on assumptions that may not hold true in varying operational contexts. These factors - linearization errors, model discrepancies, and the limitations of Gaussian distribution assumptions - all contribute to a gradual reduction in the accuracy of the covariance matrix, presenting significant challenges in maintaining the accuracy and reliability of state estimates as the system evolves over multiple time-steps.

Mitigating Information Decay

In response to the evidence of information decay in covariance, particularly noticeable from scan to scan, it becomes imperative to develop strategies that can effectively mitigate this decay. A novel solution, which aligns with the overarching system's approach of utilizing the structured environment, is proposed. This solution adheres to the objective of retaining the least amount of information necessary from scan to scan, while addressing the inadequacies of solely relying on parameter covariance.

The proposed method employs strategic sampling to preserve geometric information across subsequent scans. This technique focuses on retaining a minimal amount of crucial information. This is done by dividing the contour of the parameterized shape into a pre-defined number s of sectors. This concept is visually demonstrated for shapes like a super-ellipse and a line segment in Figure 5.5. Within each sector, the method selects and retains one hit-point that is closest to the parameterized shape, as determined by a low error evaluation through the cost function. This selection criterion ensures that some of the most representative point in each sector is preserved for future scans, with the restriction of one hit-point per sector.

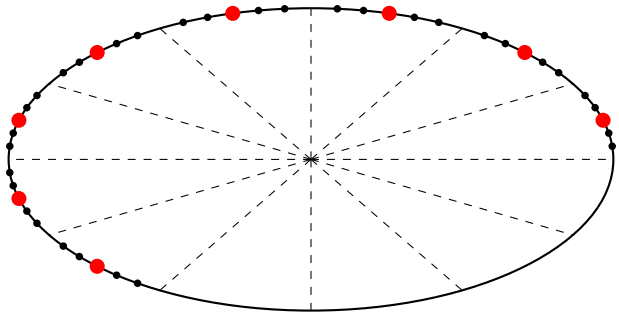
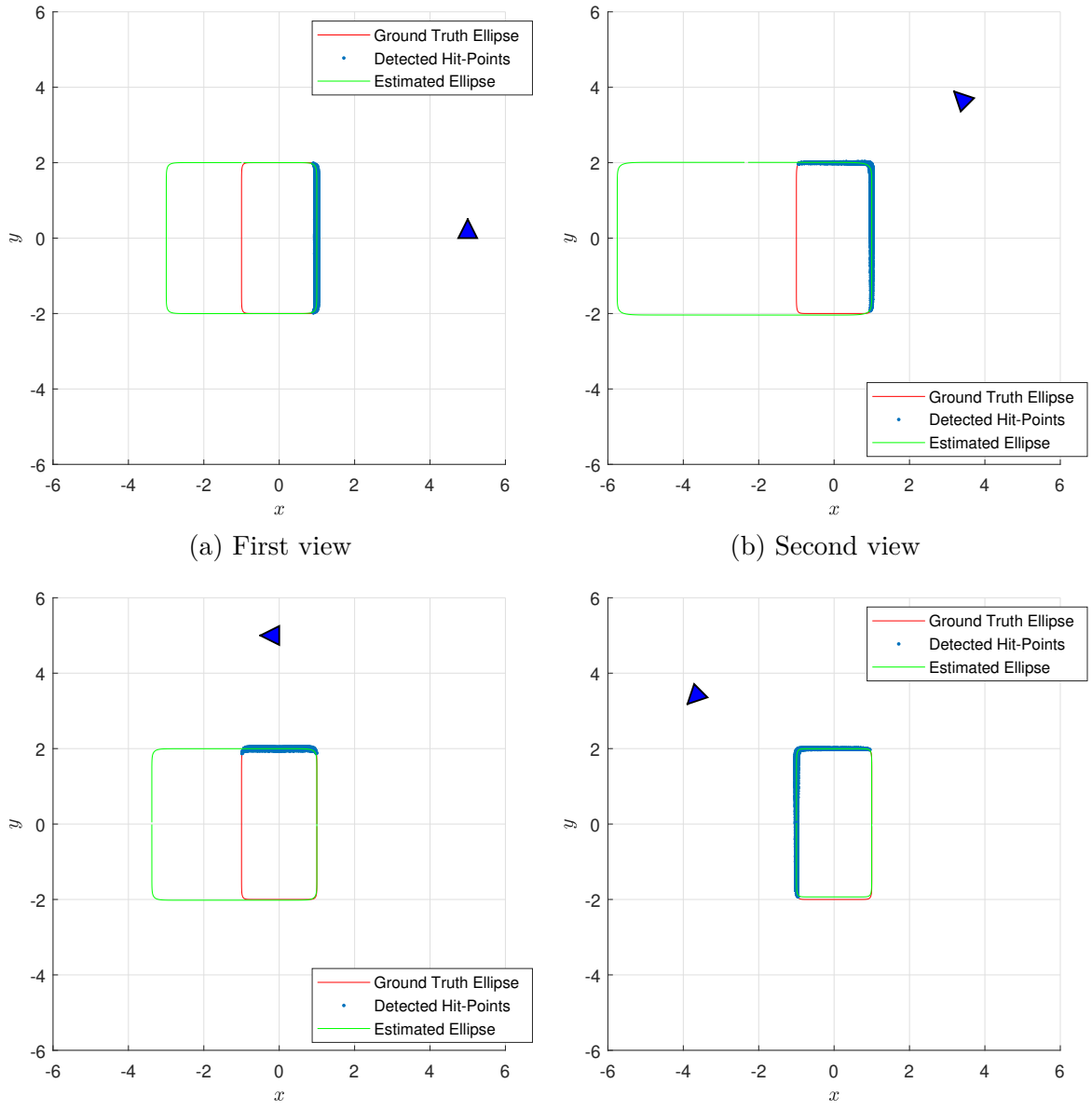


Figure 5.5 Sector visualization, the black dots are the hit-points used to estimate the ellipse in black, the red dots are the sampled hit-points (one per sector with 12 sectors)

Empirical tests of this method, in conjunction with the use of parameter covariance, have shown remarkable success in maintaining geometric information across scans. This is particularly evident in scenarios that previously exhibited significant information decay, as discussed in the preceding section. Figures 5.6 and 5.7 display scenarios analogous to those in Figures 5.3 and 5.4. However, unlike the earlier instances, these scenarios now successfully and accurately map the real shapes over multiple scans, showcasing the effectiveness of this strategic sampling approach in conjunction with parameter covariance for mitigating information decay in this context.



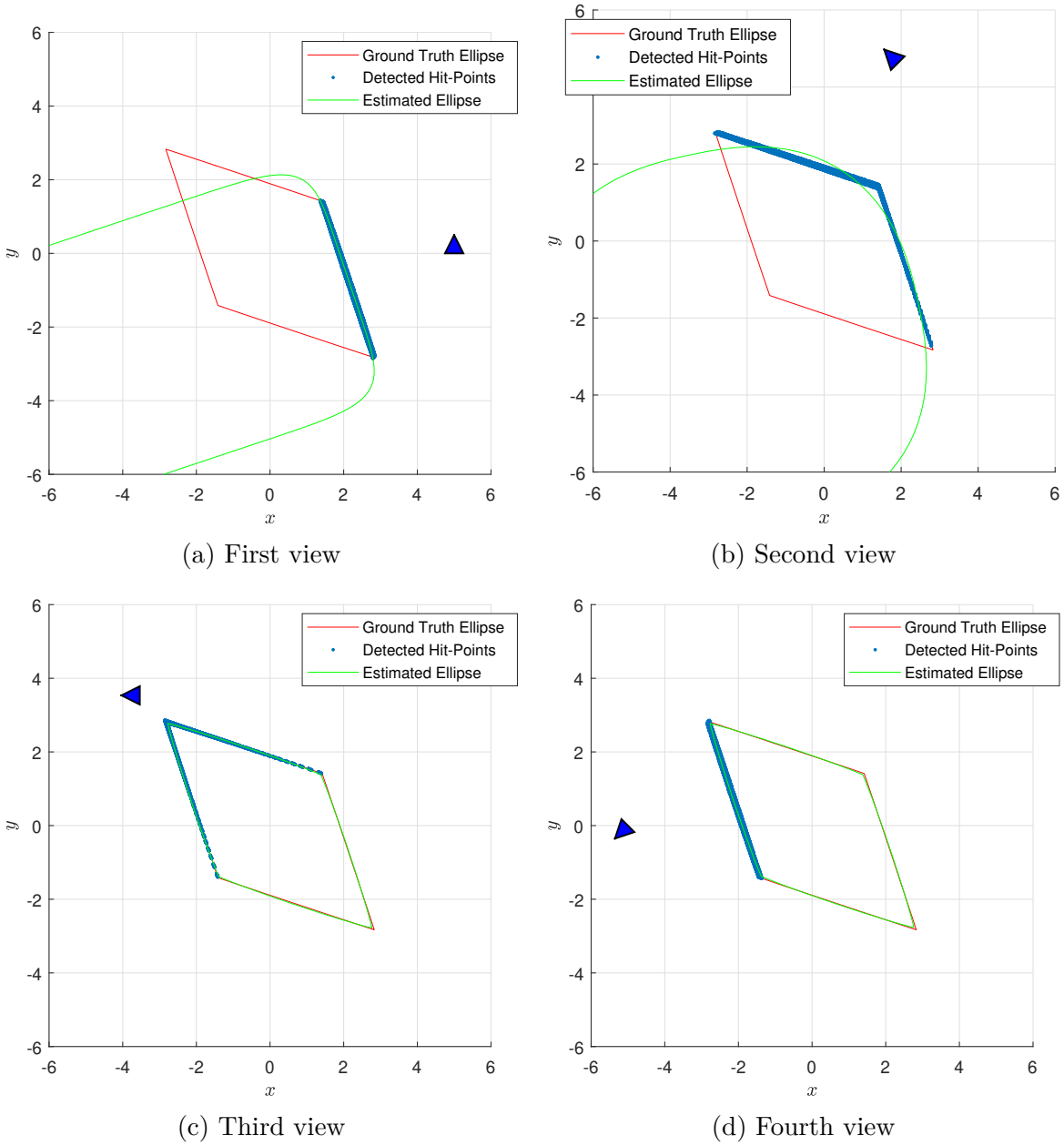
(a) First view

(b) Second view

(c) Third view

(d) Fourth view

Figure 5.6 Sequential scans of a rectangular object from different angles using the information decay mitigation.



(a) First view

(b) Second view

(c) Third view

(d) Fourth view

Figure 5.7 Sequential scans of a diamond-shaped object from different angles using the information decay mitigation.

The integration of this method into the existing equations is trivial. At time-step k, given the previously discussed cost function $\mathcal{L}'_k(\hat{\mathbf{p}}_k, \hat{\mathbf{M}}_{k,o}, \boldsymbol{\lambda})$ from equation (5.13), data retained from previous scans as per Figure 5.5 is introduced as $\tilde{\mathbf{S}}_{k,o} \subset \tilde{\mathbf{M}}_{k,o}$. The modified cost function becomes:

$$\mathcal{L}''_k(\hat{\mathbf{p}}_k, \hat{\mathbf{M}}_{k,o}\boldsymbol{\lambda}) = \mathcal{L}'_k(\hat{\mathbf{p}}_k, \hat{\mathbf{M}}_{k,o}, \boldsymbol{\lambda}) + \left(\mathcal{F}\left(L(\boldsymbol{\Xi}_k, \tilde{\mathbf{S}}_{k,o}), \hat{\mathbf{p}}_k\right) - 1\right)^2 \tag{5.14}$$

the optimization is simply solving:

$$\min_{\hat{\mathbf{p}}_k, \hat{\mathbf{M}}_{k,o}} ||\mathcal{L}''_k(\hat{\mathbf{p}}_k, \hat{\mathbf{M}}_{k,o}, \boldsymbol{\lambda})||^2 \tag{5.15}$$

5.4 Global Architecture

This section details the comprehensive system architecture for the proposed online mapping solution. The global architecture is outlined to depict the data flow between components. This includes previously seen systems in charge of various tasks which together compute a representation of the environment as the robot moves through it.

5.4.1 Main Components

The inner workings of the components presented below have been extensively documented in the preceding sections. We revisit previously investigated methodologies and the associated results to clarify the final workings of the full system. These methods are revisited in the form of building blocks with defined inputs and outputs. The implementation details are kept to a minimum and the concepts are largely taken from preceding sections.

Scan Acquisition and Pre-processing Block

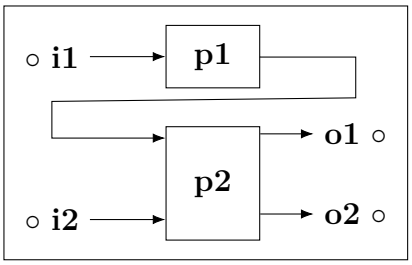
Inputs
i1. New scan **i2**. Map
Processes
p1. Data Transformation to Hit-Points **p2**. Hit-Point Association
Outputs
o1. Associated hit-points (list) **o2**. Non-associated hit-points (stream)
User-defined constants
c1. Mahalanobis distance association threshold, τ_{a}

This block, represented visually in Figure 5.8 has two main operational objectives and corresponding processes. The first, Data Transformation to Hit-Points (**p1**.), serves as the entry point for the system. Incoming LiDAR data, characterized by spatial sequential ordering and time-step separation between scans, is fed into this block. The data units are transformed into hit-points using (3.2), aligning with the system's design to process data one scan at a time.

The second process, Hit-Point Association (**p2**.), is outlined in Section 5.2. Given a map at input **i2**., which already contains post-processed features, association for each hit-point is attempted. This involves computing Mahalanobis distance between the feature and the hit-point. Hit-points with a distance less than the threshold τ_{a} are considered associated with the feature and sent over output **o1**.. Those not meeting this criterion are deemed non-associated and sent over output **o2**.. In cases where the map is initially empty, all hit-points are non-associated. Output **o2**. streams out individual hit-points, while output **o1**. is a list

of associated hit-points.

Figure 5.8 Scan Acquisition and Pre-processing Block



Feature Extraction Block

Inputs
i1. Hit-point
Processes
p1. Model-based Segmentation **p2**. Sampled Measurement Integration
Outputs
o1. Feature list (new features), each feature containing **o1.1** Parameter vector $\mathbf{p}$ **o1.2** Parameter covariance matrix $\mathbf{P}$ **o1.3** Model identifier from which the feature was created M **o1.4** Sampled measurements $\tilde{\mathbf{S}} = \{\tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2, \dots\}$
User-defined Constants
c1. Model list $\mathcal{M} = \{M_1, M_2, \dots\}$, each model containing: **c1.1** Implicit function, $\mathcal{F}_\theta$ **c1.2** Constraints on implicit function parameters, $\mathbf{p}^\theta_{\text{min}}$ and $\mathbf{p}^\theta_{\text{max}}$ **c1.3** Segmentation thresholds τ_{flex} and τ_{strict} **c1.4** Covariance matrices $\mathbf{W}$, $\boldsymbol{\nu}$ and $\mathbf{Q}$

This block, represented visually in 5.9 is in charge of feature extraction, in other words, creating new features from non-associated hit-points received in succession. In **p1**., the dual EKF architecture presented in Section 4.4.3 is applied with a user-defined model list (**c1.**).

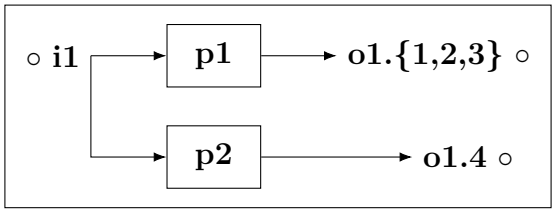


Figure 5.9 Feature Extraction Block

A model is defined as an extension of implicit functions. Due to the effect of many variables on parameterization and segmentation, it is often beneficial for mapping performance to bid multiple "models" of the same implicit function against each other when choosing the best one during segmentation. These models would have different properties despite having the same implicit function. The simplest example involves taking the "ovoid" subset of superellipses as a model by constraining the ϵ parameter between 0 and 1. The "ovoid" model could be evaluated against a "rhomboidal" model, which constrains the ϵ parameter between 1 and 2. These are the simplest examples of model subsets of implicit functions. The subsets are not limited to strictly varying the shape; they can also be characterized by their specific segmentation and parameterization behavior through modifications to their other properties. It is up to the user to create and fine-tune the models to meet specific requirements in processing speed, mapping performance, and environment adaptability.

Notice that it is particularly important that this block is fed the hit-points in a scan-like order, as the algorithm described in Figure 4.17 first groups the incoming hit-points by (very flexible) distance thresholds for initialization. At the output of the algorithm outlined in Section 4.4.3 is therefore a list of newly created features **o1.**, each coming from one of the models in the list **c1.**. Each feature includes an identifier **o1.3** indicating the model from which it was created.

In order to deal with information loss in subsequent scans described in Section 5.3.4 when refining the generated features, an additional step **p2**. is performed after the creation of each feature. The method described in 5.3.4 is applied to add sampled measurements **o1.4** to the features. These are useful in subsequent feature refinement steps.

Feature Refinement Block

Inputs
i1. Associated hit-points (list) **i2**. Existing features (list), each containing **i2.1** Parameter vector $\mathbf{p}$ **i2.2** Parameter covariance matrix $\mathbf{P}$ **i2.3** Model identifier from which the feature was created M **i2.4** Sampled measurements $\tilde{\mathbf{S}} = \{\tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2, \dots\}$ **i3**. Sampled measurements (list)
Processes
p1. Non-linear optimization
Outputs
o1. refined features (list)
User-defined Constants
c1. Non-linear optimization parameters **c2.** refinement criteria thresholds

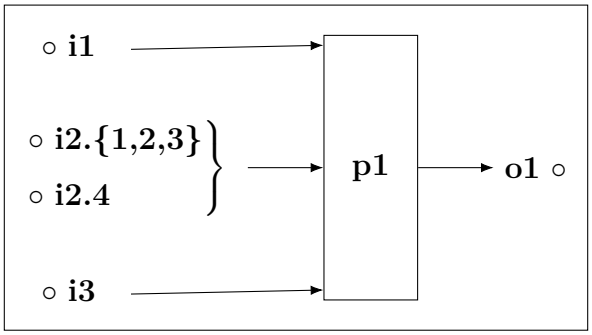


Figure 5.10 Feature refinement Block

This block, represented visually in 5.10 focuses on refining existing features with new information from associated hit-points and sampled measurements. It uses the methods outlined in Section 5.3.3 and Section 5.3.4 to effectively integrate new data while maintaining the integrity of the feature's geometric representation.

Upon receiving associated hit-points (**i1**.) and the corresponding existing features (**i2**.), the block employs non-linear optimization techniques, specifically tailored as per user-defined constants **c1.**, to update each feature's parameters (**i2**.**1**) and covariance matrix (**i2**.**2**). This process incorporates the non-linear constraints of the feature models, ensuring that the updated parameters remain within realistic bounds.

The refinement process is guided by predefined criteria thresholds **c2.**, which determine the degree and manner of refinement for each feature. These criteria are essential for deciding whether a feature should be expanded or refined based on the new data, thereby balancing the trade-off between feature stability and responsiveness to new information.

Additionally, the block integrates the sampled measurements (**i2**.**4**) from the previous scans, as described in Section 5.3.4. This step is crucial for mitigating information decay, as it ensures that the most representative points from each sector of the feature are preserved and considered during the refinement. This approach leads to a set of sampled measurements (**i2**.**4**), enhancing the feature's adaptability to new observations while retaining crucial geometric details from earlier scans.

The output of this block's process **p1**. is a list of refined features (**o1.**), each with updated parameters, covariance, and sampled measurements, ready to be integrated into the system's overall map.

5.4.2 Integration

The integration of the various blocks into a cohesive global architecture is depicted in 5.11. The data flow begins with the LiDAR sensor, which provides raw scan data to the Scan Acquisition and Pre-processing Block. This block transforms the raw data into hit-points and associates them with existing features in the map or classifies them as non-associated.

1. The non-associated hit-points from the Scan Acquisition and Pre-processing Block are then passed to the Feature Extraction Block. Here, new features are extracted from these hit-points, which are subsequently added to the map. This process involves segmenting the hit-points into distinct features and integrating sampled measurements for future reference.

2. Simultaneously, the associated hit-points, along with the current features from the map, are fed into the Feature refinement Block. This block updates the existing features by incorporating the new hit-points, thus refining the map's accuracy. The refinement process considers non-linear constraints and utilizes optimization techniques to update each feature's parameters and covariance matrices.

3. The refined features are then integrated back into the map. This updated map is subsequently used in the next cycle of processing in the association of new hit-points in the Scan Acquisition and Pre-processing Block.

5.5 Performance

5.5.1 Methodology and Metrics

In this chapter, the methodology employed to evaluate the feature refinement processes in both simulated and real-world environments is detailed. The Robotic Operating System (ROS) serves as the backbone of the experimental setup, enabling communication and coordination between different components of the robotics applications. While the main mapping algorithm we compare ourselves to is on ROS 1, our implementation predominantly operates on ROS 2. To streamline the deployment and testing process, each component, whether it be simulation software, any specific algorithm, or ROS component, is encapsulated within its own Docker container. This isolation also contributes to better reproducibility of the experiments.

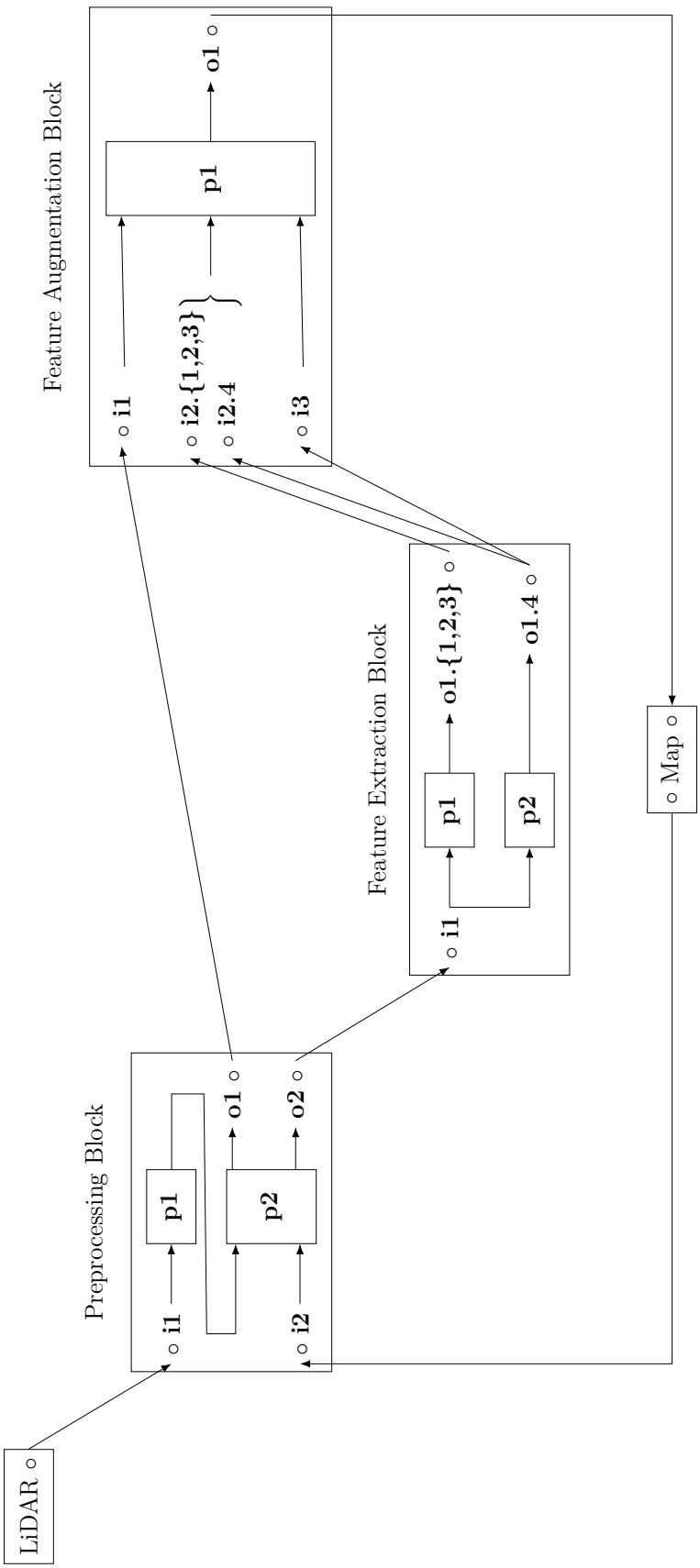


Figure 5.11 Global Architecture

Simulation Setup

Our simulated experiments are run in the simulation software CoppeliaSim, using the *Pioneer P3DX* robot and a LiDAR sensor model called *fastHokuyo*, both of which are default models within the software. The behavior of the fastHokuyo LiDAR is standardized across simulations using a Lua script, details of which are provided in the simulation folder of the GitHub link. The assumption underlying our experiments is that the environment consists entirely of objects that can be accurately represented by super-ellipses. These objects are procedurally generated via a MATLAB script, available at the GitHub link, and then imported into our simulation environments.

The environments for the experiments are publicly accessible through the project's GitHub repository. In the simulated environment, depicted in Figure 5.12, the layout is designed to test the algorithm's ability to construct a map in scenarios where some objects are initially obstructed. Two interior walls obscure two objects, yet these objects are spaced sufficiently apart to facilitate straightforward segmentation. The designated path for the robot forms a cross that intersects the center and extends towards the four corners of the map.

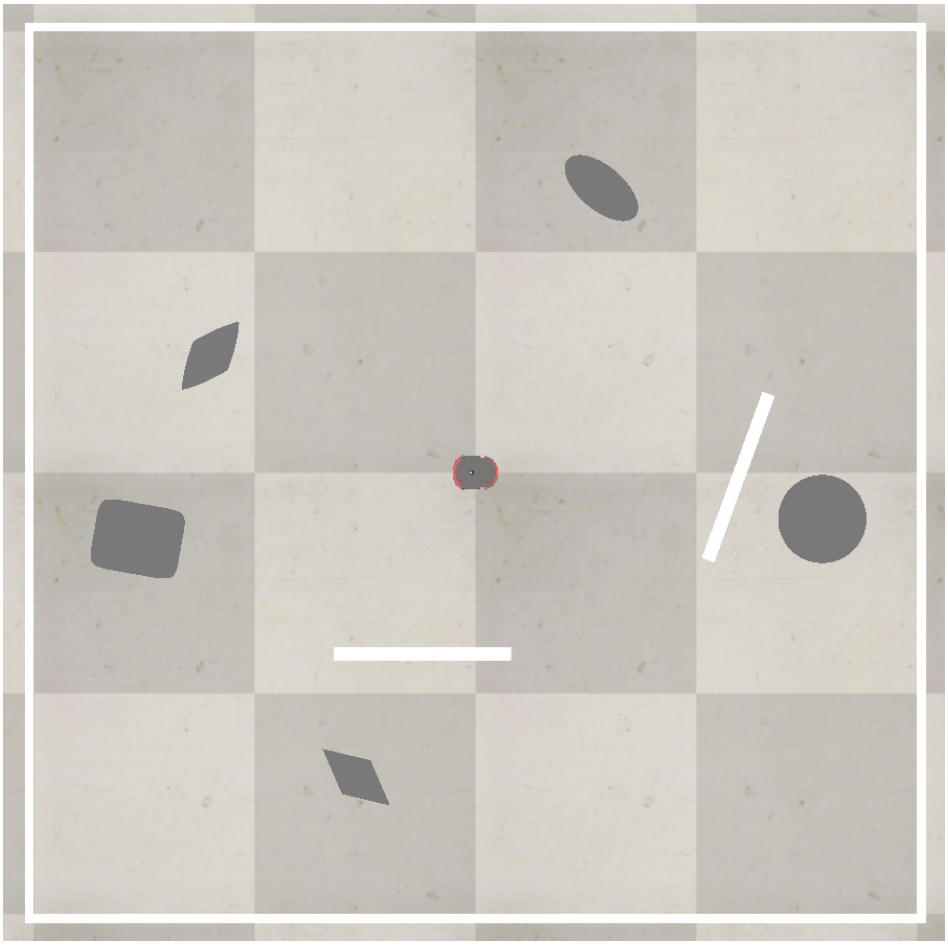


Figure 5.12 Simulated environment

Experimental Setup for Real-world Experiments

For the real-world experiments, a robotic platform based on an iRobot Create 3 with a Hokuyo UST-20LX LiDAR sensor connected to a Raspberry Pi for data acquisition is used. A Vicon motion capture (mocap) system provides ground truth data on the robot's location. This setup is illustrated in Figure 5.13, which shows a photo of the experimental setup.

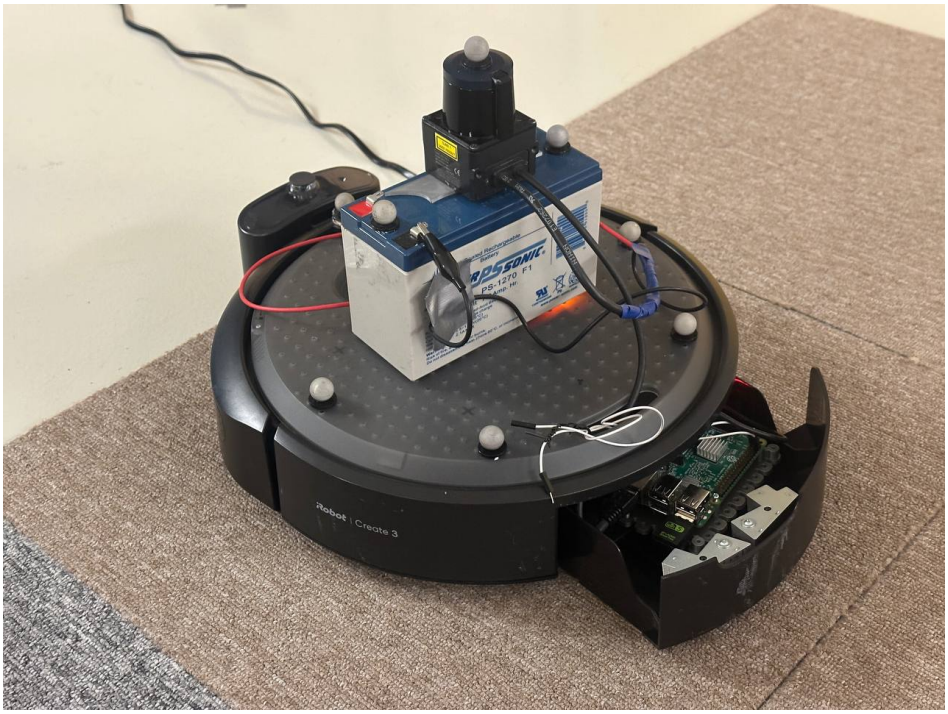


Figure 5.13 Robotic platform setup

The testing environment is enclosed using extruded polystyrene foam walls, replicating interior walls. The objects in the testing area are constructed from combinations of boxes and cardboard, whose shapes can be approximated by various super-ellipses in shape. These objects are placed to challenge both the robot's ability to segment objects that are in close proximity to each other and the robot's capability to detect and segment objects as they become visible in subsequent scans. The robot begins at a distance from the objects, tracing a path towards a farther obstacle, and eventually going to the other side of an interior, which allows it to see the obstructed object. Figure 5.14 show the environment as the robot navigates it.

(a) First view (b) Second view

Figure 5.14 Robot's testing environment

5.5.2 Analysis and Interpretation

To compare the developed mapping algorithm with the state-of-the-art in 2D for LiDAR-based systems, the focus is placed on SLAM algorithms' mapping performance by providing perfect localization. This is achieved for simulations by extracting the exact location using the simulation software. For real-world experiments, the motion capture system is used. The SLAM algorithms behave as mapping algorithms when passed a null (or near null) odometry error covariance matrix as well as perfect odometry information. The goal of this approach is to highlight the strengths and weaknesses of the developed algorithm in mapping in comparison to other algorithms.

A direct comparison is conducted by converting the geometric feature-based map into a grid-based map using the Mahalanobis association scheme presented in Section 5.2.2. Associations are attempted on the center point of each voxel in the 2D grid with every object; if it meets the defined threshold, then the voxel is marked as "full". The ground truths for the simulated environments are derived by directly transforming the maps into grids. For the real environment, ground truth is obtained by physically measuring objects and their positions to approximate them as super-ellipses.

To compare occupancy grids, several commonly used metrics are available. We define the following sets: $\bar{A}$ and $\bar{B}$ represent the free cells according to the algorithm and the ground truth, respectively. Similarly, A represents the occupied cells according to the algorithm, and B represents the occupied cells according to the ground truth. The Intersection over Union

(IoU) metric is defined as the ratio of the intersection of the predicted occupied cells and the ground truth occupied cells to their union. Mathematically, the IoU for each algorithm is calculated as:

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|}$$

Precision is defined as the ratio of true positives (correctly predicted occupied cells) to the total predicted positives (all cells predicted as occupied), and is calculated as:

$$\text{Precision} = \frac{|A \cap B|}{|A|}$$

Recall is the ratio of true positives (correctly predicted occupied cells) to the total actual positives (all cells that are actually occupied), and is given by:

$$\text{Recall} = \frac{|A \cap B|}{|B|}$$

The F1 score, which is the harmonic mean of precision and recall, is calculated as:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Accuracy measures the ratio of correctly predicted instances (both true positives and true negatives) to the total instances, and is expressed as:

$$\text{Accuracy} = \frac{|A \cap B| + |\bar{A} \cap \bar{B}|}{|A \cup B| + |\bar{A} \cup \bar{B}|}$$

Finally, specificity, which is the ratio of true negatives (correctly predicted free cells) to the total actual negatives (all cells that are actually free), is given by:

$$\text{Specificity} = \frac{|\bar{A} \cap \bar{B}|}{|\bar{B}|}$$

Each cell in the grid is categorized into one of four categories:

- **True Positives (TP):** These are cells that both the algorithm and the ground truth agree are occupied. $TP = |A \cap B|$, representing the correctly identified occupied cells.
- **False Positives (FP):** These are the cells that the algorithm incorrectly identifies

as occupied when they are actually unoccupied in the ground truth. $FP = |A \setminus B|$, indicating overestimation by the algorithm.

- **True Negatives (TN):** These are the cells that both the algorithm and the ground truth agree are unoccupied. $TN = |U \setminus (A \cup B)|$, where U is the set of all cells, indicating the correctly identified unoccupied cells.
- **False Negatives (FN):** These are the cells that the algorithm incorrectly identifies as unoccupied when they are occupied in the ground truth. $FN = |B \setminus A|$, representing underestimation by the algorithm.

Together, these measurements provide a comprehensive view of the algorithm's performance in replicating the actual environment's layout in the occupancy grid.

5.5.3 Results

Figure 5.15 illustrates the positions from which the robot conducted its scans in the simulated environment, along with the aggregate of detected hit-points from all scans and provides a view of the implicit function-based ground truth for the simulated setup.

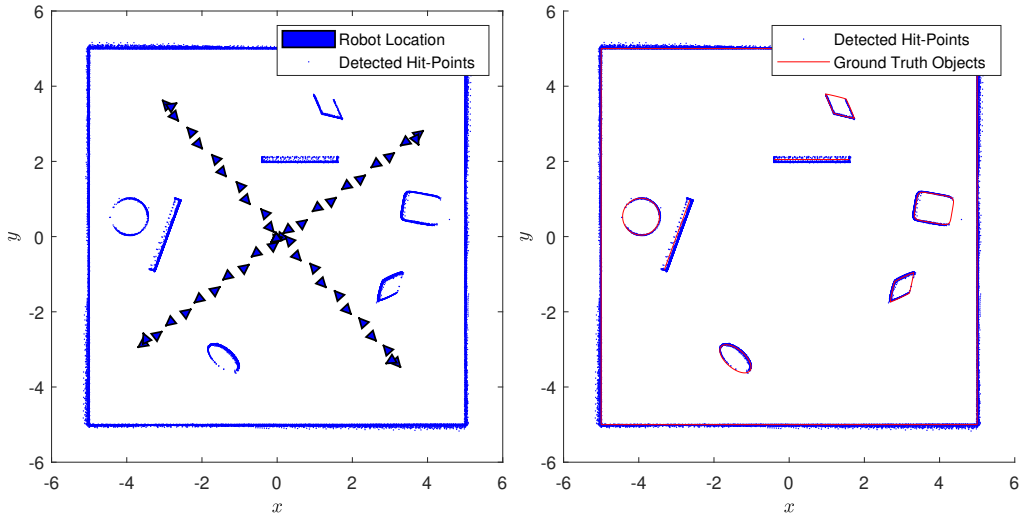


Figure 5.15 (left) Scans extracted in simulated environment, (right) Implicit function-based ground truth in the simulated environment

Figure 5.16 shows this ground truth converted into an occupancy grid and the mapping results from our algorithm using the implicit function approach for the simulated data.

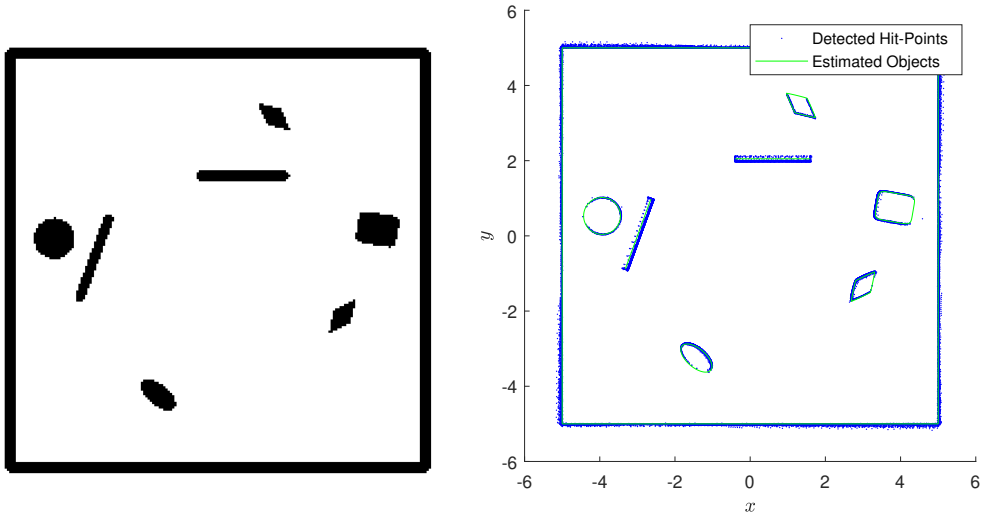


Figure 5.16 (left) Implicit function ground truth converted into occupancy grid for the simulated environment, (right) Our algorithm's implicit function results in the simulated environment (black: occupied, white: unoccupied)

Finally, Figure 5.17 offers a side-by-side comparison of our algorithm's results, transformed into an occupancy grid, against the occupancy grid results from the GMapping algorithm [66] for the simulated environment.

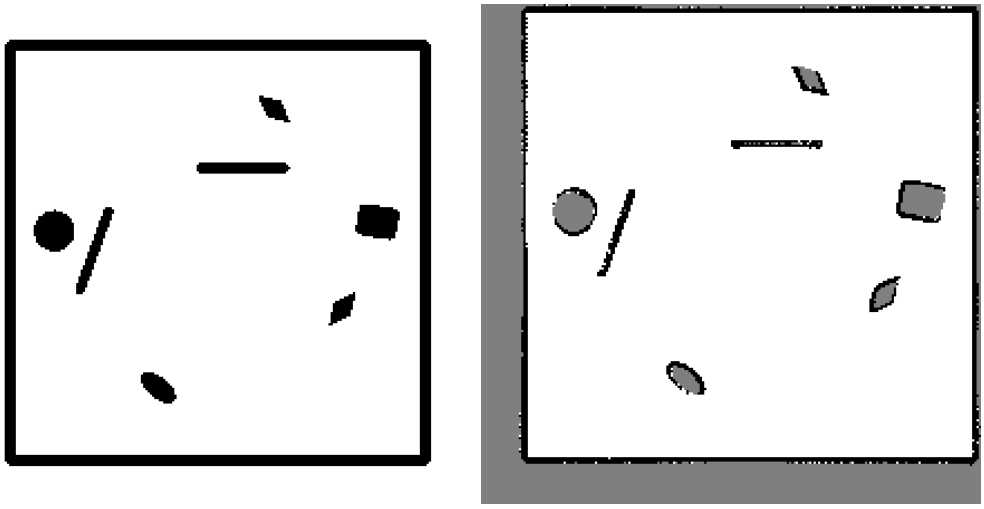


Figure 5.17 (left) Our algorithm's results, (right) GMapping's results in the simulated environment (black: occupied, white: unoccupied, gray: unknown)

Figure 5.18 illustrates the positions from which the robot conducted its scans, along with the

aggregate of detected hit-points from all scans and provides a view of the implicit function-based ground truth.

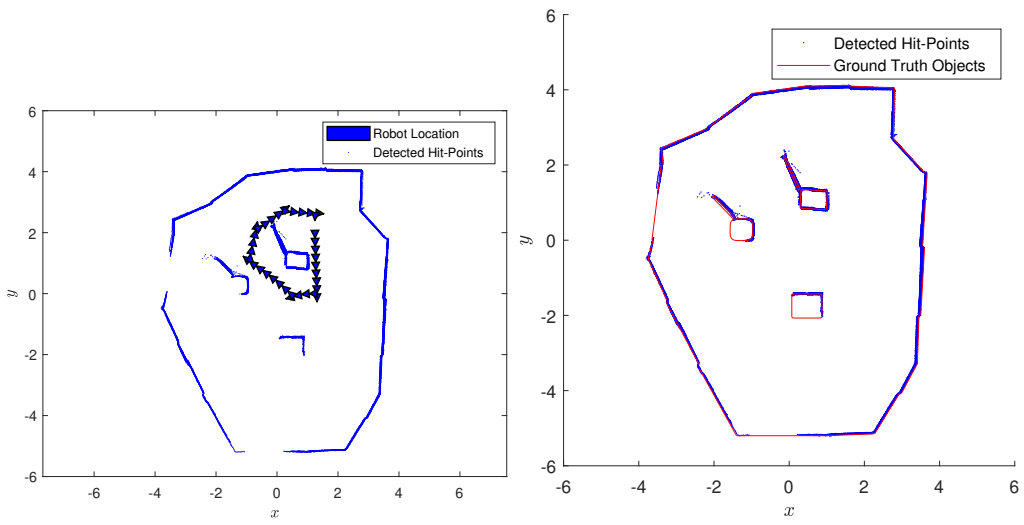


Figure 5.18 (left) Scans extracted in the real environment, (right) Implicit function-based ground truth

Figure 5.19 shows this ground truth converted into an occupancy grid and displays the mapping results from our algorithm in the implicit function approach.

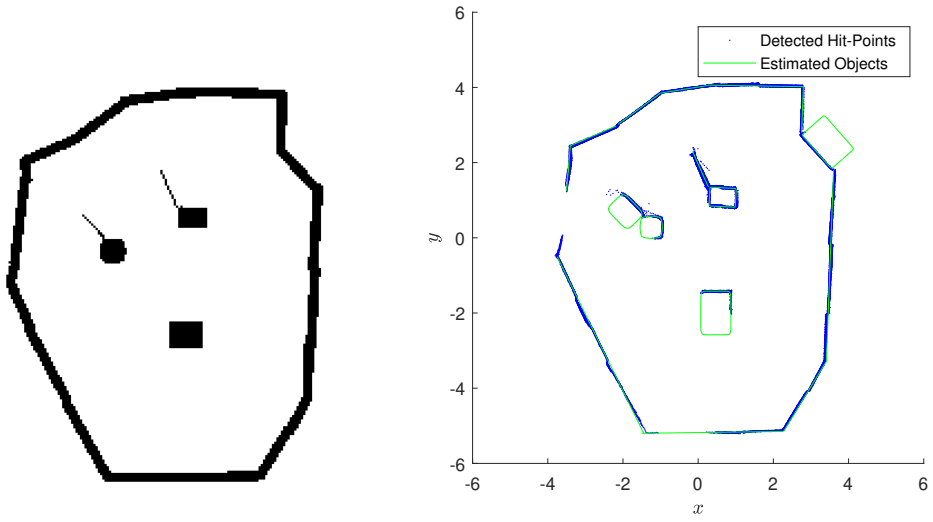


Figure 5.19 (left) Implicit function ground truth converted into occupancy grid, (right) Our algorithm's implicit function results in the real environment (black: occupied, white: unoccupied)

Finally, Figure 5.20 offers a side-by-side comparison of our algorithm's results, transformed into an occupancy grid, against the occupancy grid results from the GMapping algorithm.

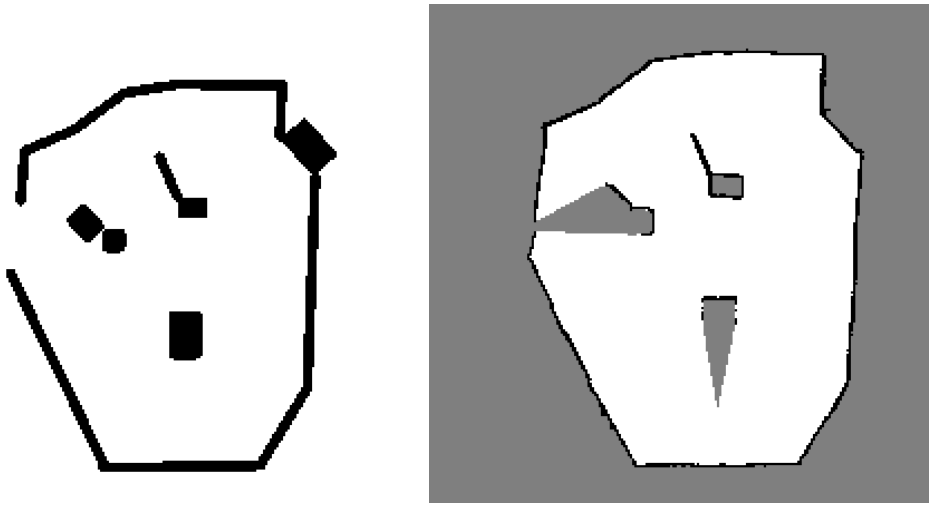


Figure 5.20 (left) Our algorithm's results, (right) GMapping's results in the real environment (black: occupied, white: unoccupied, gray: unknown)

The results are presented in Table 5.1.

Metric	**Real Environment**		**Simulated Environment**	
	Our Algorithm	**GMapping**	**Our Algorithm**	**GMapping**
True Positives	45828	19340	44502	36506
False Positives	537	1063	356	1883
True Negatives	2773	1062	4619	2210
False Negatives	1038	87	699	396
Precision	0.9884	0.9479	0.9921	0.9509
F1 Score	0.9831	0.9711	0.9883	0.9697
Recall	0.9779	0.9955	0.9845	0.9893
Accuracy	0.9686	0.9466	0.9790	0.9444
Specificity	0.8378	0.4998	0.9284	0.5399
IoU Score	0.9668	0.9439	0.9768	0.9412

Table 5.1 Comparison of various metrics between our algorithm and GMapping in real and simulated environments

In the real environment, our algorithm occasionally missed some lines in the walls due to insufficient points for initialization, as shown in Figures 5.18 and 5.19. Additionally, some walls were estimated as rectangles because the scan vaguely curved at the edge, giving them

more depth than present in the ground truth. This discrepancy arose because the backside of these walls was not visited, resulting in overestimations, while GMapping marked these areas as unknown due to lack of data. Similarly, the backside of the bottom box, which was not visited, led to an overestimation of its size in our algorithm's results, whereas GMapping left it as unknown, as seen in Figure 5.20. Overall, GMapping had significantly more unknowns due to its grid-based approach without environmental assumptions, while our algorithm's assumptions enabled it to estimate implicit functions, sometimes leading to overestimations and missing smaller objects due to the need for sufficient initialization points.

The metrics in Table 5.1 further support these observations. In the real environment, our algorithm shows higher True Positives (45828 vs. 19340) and lower False Positives (537 vs. 1063) compared to GMapping, indicating more accurate identification of occupied spaces. Precision (0.9884 vs. 0.9479) and F1 Score (0.9831 vs. 0.9711) are consistently higher for our algorithm, showing better performance in correctly identifying true positives while minimizing false positives. Recall is slightly lower for our algorithm (0.9779 vs. 0.9955) in the real environment but still very high, suggesting a slight trade-off in sensitivity for improved precision. Accuracy (0.9686 vs. 0.9466) and Specificity (0.8378 vs. 0.4998) are notably higher for our algorithm, reflecting its overall better performance in correctly identifying both occupied and unoccupied spaces. The IoU Score (0.9668 vs. 0.9439) is also higher for our algorithm, indicating a more accurate overlap between the predicted and actual occupied spaces. These metrics underline the strengths and weaknesses of our approach, emphasizing its ability to make more accurate overall estimates with some limitations in specific scenarios.

In the simulated environment, both our algorithm and GMapping produced nearly perfect results because the environment was extensively covered and the walls were perfectly straight, with no noise at the edges, preventing misidentification of walls as rectangular super-ellipses, as illustrated in Figures 5.15, 5.16, and 5.17. The metrics in Table 5.1 show that our algorithm maintains higher True Positives (44502 vs. 36506) and lower False Positives (356 vs. 1883) compared to GMapping in the simulated environment as well. Precision (0.9921 vs. 0.9509) and F1 Score (0.9883 vs. 0.9697) remain higher for our algorithm. Recall (0.9845 vs. 0.9893) and Accuracy (0.9790 vs. 0.9444) also indicate strong performance, with Specificity (0.9284 vs. 0.5399) and IoU Score (0.9768 vs. 0.9412) being better for our algorithm.

5.5.4 Efficiency

Implicit function-based maps represent the environment through continuous geometric functions, defined implicitly, rather than explicitly storing discrete cells like in gridmaps. This

representation keeps in memory only the function parameters and uncertainty information from scan to scan, significantly reducing memory usage compared to gridmaps, which store vast amounts of pixel data. Computational efficiency is also enhanced, as associations and updates involve comparing with a number of functions rather than the entire grid, avoiding the costly operation of evaluating every pixel. Although it was not quantitatively measured, it is evident that our implicit function-based implementation is more efficient than the compared grid-based algorithm.

5.6 Conclusion

The principal goal of this chapter was to refine features over multiple scans, leveraging association algorithms and optimization techniques. The primary focus was on associating new hit-points with existing features to maintain map integrity and improve features. The Mahalanobis distance proved instrumental in establishing associations. Feature refinement was first addressed using an IEKF approach, then using a non-linear least-squares approach.

A significant challenge identified was the phenomenon of information decay in the covariance matrices over multiple scans. The impact of this decay on mapping performance was highlighted using empirical evidence. To address this, a novel mitigation strategy involving strategic sampling within defined sectors was proposed. This approach retained critical geometric information across scans, preserving the most representative hit-points and significantly reducing information loss. The integration of sampled measurements into the optimization process ensured that the refinement algorithms could adapt to new observations while retaining crucial details from previous scans.

The global architecture for the proposed multi-scan feature refinement system was detailed. Performance evaluation was conducted through simulations and real-world experiments. The metrics-based analysis provided a quantitative measure of the system's performance, comparing it to GMapping in several key aspects. The precision, F1 score, accuracy, and specificity metrics consistently showed higher values for the proposed algorithm, reflecting its ability to accurately identify and represent features in both simulated and real-world environments. The results demonstrated that while the algorithm occasionally overestimated feature sizes due to the assumptions made about the environment, its overall performance in accurately mapping and filling in blanks with minimal exploration was better.

CHAPTER 6 CONCLUSION

6.1 Summary of Work

In this thesis, a comprehensive approach to robotic mapping that exploits the structured nature of built environments has been presented. The research has focused on the development of a mapping algorithm that leverages geometric regularities. The core contributions of this work are centered around the development of a model-aware multi-model geometric feature extraction framework, which addresses the shortcomings of existing low-dimensional feature-based SLAM methods.

A significant innovation introduced in this work is a framework capable of generalizing to any geometric model and handling multiple models simultaneously, thereby overcoming the limitations of previous methods that were restricted to specific models or could not manage more than two models concurrently. The proposed approach instantiates an adaptation of the dual-EKF architecture from the SEGMENTS algorithm [15] for each implicit model used and employs a FSM with conditions based on the Mahalanobis distance to effectively switch between models. This has been tested with line segments and super-ellipses as feature types.

An online mapping algorithm that incrementally updates the map at each time-step, estimating object boundaries, associating hit-points with objects, and refining parameter estimates for each object has been developed. The proposed architecture for the online mapping system has been designed with modularity in mind, facilitating the coordination of various functional blocks through a system of inputs and outputs and encapsulated data structures. The functional blocks include pre-processing of raw data, feature extraction, and feature updating.

Extensive Monte-Carlo simulations and real-world experiments have been conducted throughout this thesis to evaluate the performance of the algorithms. The results have shown that the system is capable of accurately segmenting single-scan data and refining features over multiple scans. The challenge of information decay has been addressed by proposing a novel mitigation strategy based on strategic sampling within defined sectors, which preserves critical geometric information across scans.

6.2 Limitations

While the research objectives have been fulfilled, there are limitations that must be acknowledged. The performance of the system is contingent upon the accuracy of the initial assumptions regarding the environment's structure. Environments that deviate from these assumptions may result in sub-optimal mapping outcomes. Additionally, the reliance on the Mahalanobis distance for model switching in the FSM may lead to occasional misclassifications from which the system is unable to recover, particularly in environments with high levels of clutter or ambiguous geometries. Finally, the method depends on accurate fine-tuning for every environment, depending on the scale of the obstacles contained in it. User-set parameters that work well in an environment with small objects will not work well in an environment with larger objects. This is a limitation inherent to the method that cannot be removed without modifying the architecture significantly.

6.3 Future Research

The work presented in this thesis opens several avenues for future research. Firstly, extending the system to handle more complex environments, such as those with non-convex shapes or multiple levels, would be a natural next step. Another immediate direction is the extension of the framework to handle dynamic environments. Incorporating techniques for detecting and tracking moving objects could significantly enhance the utility of the mapping system in real-world scenarios. Additionally, leveraging the mapping system for SLAM could provide significant advancements in real-time localization in structured environments. Finally, integrating the mapping algorithm with higher-level robotic functions, such as planning and decision-making, could enable robots to perform complex tasks in indoor environments using this framework.

REFERENCES

[1] L. Pedraza, G. Dissanayake, J. Valls Miro, D. Rodríguez-Losada, and F. Matia, "BS-SLAM: Shaping the World," Jun. 2007.

[2] O. Alexandrov, "Signed distance function visualization," Wikimedia Commons, 2007, accessed: 2024-06-24. [Online]. Available: https://commons.wikimedia.org/wiki/File:Signed_distance1.png

[3] Hellingspaul, "Various superquadratics to show the effects of the powers in the general equation," Wikimedia Commons, 2021, accessed: 2024-06-24. [Online]. Available: https://commons.wikimedia.org/wiki/File:Superkwadrieken8.jpg

[4] W. Commons, "File:superellipse.svg — wikimedia commons, the free media repository," 2022, [Online; accessed 17-June-2024]. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:Superellipse.svg&oldid=656421344

[5] M. Shafiqul Islam, M. Tarequl Islam, and M. G. G. Faruque, "A Survey on LiDAR-Based SLAM Technique for an Autonomous Model Using Particle Filters," in *Soft Computing for Security Applications*, ser. Advances in Intelligent Systems and Computing, G. Ranganathan, X. Fernando, F. Shi, and Y. El Allioui, Eds. Singapore: Springer, 2022, pp. 227–240.

[6] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *Trans. Rob.*, vol. 32, no. 6, p. 1309–1332, dec 2016. [Online]. Available: https://doi.org/10.1109/TRO.2016.2624754

[7] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto, "Voxgraph: Globally Consistent, Volumetric Mapping Using Signed Distance Function Submaps," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 227–234, 2020.

[8] R. P. I. I. P. Laboratory and D. Meagher, *Octree Encoding: a New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer*, 1980. [Online]. Available: https://books.google.ca/books?id=CgRPOAAACAAJ

[9] H. Fuchs, Z. M. Kedem, and B. F. Naylor, "On visible surface generation by a priori tree structures," *SIGGRAPH Comput. Graph.*, vol. 14, no. 3, p. 124–133, jul 1980. [Online]. Available: https://doi.org/10.1145/965105.807481

[10] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 4758–4765, iSSN: 2153-0866.

[11] K. Honda, K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Generalized LOAM: LiDAR Odometry Estimation With Trainable Local Geometric Features," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 459–66, 2022.

[12] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," Jul. 2020.

[13] koide3, "hdl_graph_slam," Sep. 2023, original-date: 2018-01-01T07:35:43Z. [Online]. Available: https://github.com/koide3/hdl_graph_slam

[14] M. Adams and A. Kerstens, "Tracking naturally occurring indoor features in 2-D and 3-D with lidar range/amplitude data," *International Journal of Robotics Research*, vol. 17, no. 9, pp. 907–923, 1998, place: United States INIS Reference Number: 30006784.

[15] S. Roumeliotis and G. Bekey, "SEGMENTS: a layered, dual-Kalman filter algorithm for indoor feature extraction," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, vol. 1, Oct. 2000, pp. 454–461 vol.1.

[16] S. Zhang, L. Xie, and M. Adams, "Feature extraction for outdoor mobile robot navigation based on a modified Gauss–Newton optimization approach," *Robotics and Autonomous Systems*, vol. 54, pp. 277–287, Apr. 2006.

[17] S. Zhang, M. Adams, F. Tang, and L. Xie, "Geometrical Feature Extraction Using 2D Range Scanner," Jul. 2003, pp. 901–905.

[18] M. R. B. Clarke, "Pattern Classification and Scene Analysis," *Journal of the Royal Statistical Society: Series A (General)*, vol. 137, no. 3, pp. 442–443, 1974, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.2307/2344977. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.2307/2344977

[19] G. A. Borges and M.-J. Aldon, "Line Extraction in 2D Range Images for Mobile Robotics," *Journal of Intelligent and Robotic Systems*, vol. 40, no. 3, pp. 267–297, Jul. 2004. [Online]. Available: https://doi.org/10.1023/B:JINT.0000038945.55712.65

[20] P. Nunez, R. Vazquez-Martin, J. del Toro, A. Bandera, and F. Sandoval, "Feature extraction from laser scan data based on curvature estimation for mobile robotics," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 1167–1172, iSSN: 1050-4729. [Online]. Available: https://ieeexplore.ieee.org/document/1641867

[21] M. Adams, T. Fan, W. S. Wijesoma, and C. Sok, "Convergent Smoothing and Segmentation of Noisy Range Data in Multiscale Space," *IEEE Transactions on Robotics*, vol. 24, no. 3, pp. 746–753, Jun. 2008, conference Name: IEEE Transactions on Robotics. [Online]. Available: https://ieeexplore.ieee.org/document/4493418

[22] L. Pedraza, D. Rodriguez-Losada, F. Matia, G. Dissanayake, and J. Valls Miro, "Extending the Limits of Feature-Based SLAM With B-Splines," *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 353–366, Apr. 2009, conference Name: IEEE Transactions on Robotics.

[23] Rui-Jun Yan, Jing Wu, Ji Yeong Lee, and Chang-Soo Han, "Representation of 3D Environment Map Using B-Spline Surface with Two Mutually Perpendicular LRFs," *Mathematical Problems in Engineering*, vol. 2015, p. 690310 (14 pp.), 2015, place: USA Publisher: Hindawi Publishing Corporation.

[24] R. T. Rodrigues, N. Tsiogkas, A. Pascoal, and A. P. Aguiar, "Online Range-Based SLAM Using B-Spline Surfaces," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1958–1965, Apr. 2021, conference Name: IEEE Robotics and Automation Letters.

[25] A. N. Ravari and H. D. Taghirad, "NURBS-based representation of urban environments for mobile robots," in *2016 4th International Conference on Robotics and Mechatronics (ICROM)*, Oct. 2016, pp. 20–25.

[26] R. O. Castle, D. J. Gawley, G. Klein, and D. W. Murray, "Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, Apr. 2007, pp. 4102–4107, iSSN: 1050-4729.

[27] M. Hosseinzadeh, Y. Latif, T. Pham, N. Suenderhauf, and I. Reid, "Structure Aware SLAM using Quadrics and Planes," Nov. 2018. [Online]. Available: http://arxiv.org/abs/1804.09111

[28] A. Ravankar, A. Ravankar, T. Emaru, and Y. KOBAYASHI, "Line Segment Extraction and Polyline Mapping for Mobile Robots in Indoor Structured Environments Using

Range Sensors," *SICE Journal of Control, Measurement, and System Integration*, vol. 13, pp. 138–147, May 2020.

[29] J.-D. Fossel, K. Tuyls, and J. Sturm, "2D-SDF-SLAM: A signed distance function based SLAM frontend for laser scanners," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 1949–1955.

[30] A. Millane, H. Oleynikova, J. Nieto, R. Siegwart, and C. Cadena, "Free-Space Features: Global Localization in 2D Laser SLAM Using Distance Function Maps," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 1271–1277, iSSN: 2153-0866.

[31] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart, "Signed Distance Fields: A Natural Representation for Both Mapping and Planning." ETH Zurich, 2016, p. 6 p., artwork Size: 6 p. Medium: application/pdf. [Online]. Available: http://hdl.handle.net/20.500.11850/128029

[32] K. Daun, S. Kohlbrecher, J. Sturm, and O. von Stryk, "Large Scale 2D Laser SLAM using Truncated Signed Distance Functions," in *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Sep. 2019, pp. 222–228, iSSN: 2475-8426.

[33] K. Daun, M. Schnaubelt, S. Kohlbrecher, and O. von Stryk, "HectorGrapher: Continuous-time Lidar SLAM with Multi-resolution Signed Distance Function Registration for Challenging Terrain," in *2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Oct. 2021, pp. 152–159, iSSN: 2475-8426.

[34] I. Vizzo, T. Guadagnino, J. Behley, and C. Stachniss, "VDBFusion: Flexible and Efficient TSDF Integration of Range Sensor Data," *Sensors*, vol. 22, no. 3, p. 1296 (24 pp.), 2022, place: Switzerland Publisher: MDPI.

[35] M. M. Johari, C. Carta, and F. Fleuret, "ESLAM: Efficient Dense SLAM System Based on Hybrid Representation of Signed Distance Fields," Apr. 2023, arXiv:2211.11704 [cs]. [Online]. Available: http://arxiv.org/abs/2211.11704

[36] J. Arukgoda, R. Ranasinghe, and G. Dissanayake, "Robot Localisation in 3D Environments Using Sparse Range Measurements," in *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Jul. 2019, pp. 551–558, iSSN: 2159-6255.

[37] J. Vandorpe, H. Van Brussel, and H. Xu, "Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2D range finder,"

in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, Apr. 1996, pp. 901–908 vol.1, iSSN: 1050-4729. [Online]. Available: https://ieeexplore.ieee.org/document/503887

[38] R. Pascoal, V. Santos, C. Premebida, and U. Nunes, "Simultaneous Segmentation and Superquadrics Fitting in Laser-Range Data," *IEEE Transactions on Vehicular Technology*, vol. 99, May 2014.

[39] Jiaheng Zhao, Liang Zhao, Shoudong Huang, and Yue Wang, "2D Laser SLAM With General Features Represented by Implicit Functions," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4329–36, Jul. 2020, place: USA Publisher: IEEE.

[40] Y. Meng and B. Zhou, "Ellipsoid SLAM with Novel Object Initialization," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, Aug. 2022, pp. 1333–1338, iSSN: 2161-8089.

[41] N. Vaskevicius, K. Pathak, R. Pascanu, and A. Birk, "Extraction of quadrics from noisy point-clouds using a sensor noise model," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 3466–3471, iSSN: 1050-4729.

[42] Y. Lu and D. Song, "Visual Navigation Using Heterogeneous Landmarks and Unsupervised Geometric Constraints," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 736–749, Jun. 2015, conference Name: IEEE Transactions on Robotics.

[43] M. Liu, S. Huang, and G. Dissanayake, "Feature based SLAM using laser sensor data with maximized information usage," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1811–1816, iSSN: 1050-4729. [Online]. Available: https://ieeexplore.ieee.org/document/5980504

[44] T. Bailey, "Mobile Robot Localisation and Mapping in Extensive Outdoor Environments," 2002. [Online]. Available: https://www.semanticscholar.org/paper/Mobile-Robot-Localisation-and-Mapping-in-Extensive-Bailey/b9fc38a6ce41c8709377f6a5974a66b59de9487c

[45] Y. Bar-Shalom, X. Li, and T. Kirubarajan, "Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software." Wiley, Jan. 2002, edition: 1. [Online]. Available: https://onlinelibrary.wiley.com/doi/book/10.1002/0471221279

[46] S. Zhang, L. Xie, and M. Adams, "An Efficient Data Association Approach to Simultaneous Localization and Map Building," *I. J. Robotic Res.*, vol. 24, pp. 49–60, Jan. 2005.

[47] R. Vazquez-Martin, P. Nunez, J. del Toro, A. Bandera, and F. Sandoval, "Simultaneous mobile robot localization and mapping using an adaptive curvature-based environment description," in *MELECON 2008 - The 14th IEEE Mediterranean Electrotechnical Conference*, May 2008, pp. 343–349, iSSN: 2158-8481. [Online]. Available: https://ieeexplore.ieee.org/document/4618458

[48] J. Zhao, S. Huang, L. Zhao, Y. Chen, and X. Luo, "Conic Feature Based Simultaneous Localization and Mapping in Open Environment via 2D Lidar," *IEEE Access*, vol. 7, pp. 173 703–173 718, 2019, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/document/8917627

[49] J. Zhao, L. Zhao, S. Huang, and Y. Wang, "2D Laser SLAM With General Features Represented by Implicit Functions," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4329–4336, Jul. 2020, conference Name: IEEE Robotics and Automation Letters. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9099049?casa_token=tKkfQq0Ot_UAAAAA:zvLoNO3BK1lwQUnUtHs7ch9vx1dOEAJA1T2_H0wYo91W5g5g0c8PwHl5OdjenCVaFxCbLb2B8A

[50] J. Zhao, T. Li, T. Yang, L. Zhao, and S. Huang, "2D Laser SLAM With Closed Shape Features: Fourier Series Parameterization and Submap Joining," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1527–1534, Apr. 2021, conference Name: IEEE Robotics and Automation Letters. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9351608?casa_token=nNq56Dc_dvYAAAAA:BGcuChYniqou1O7NgpiYhsWCSIGF3NexQZ_YLIZ3HJm0y2-mx6WIgBrgUi0lt9Y9aJTGpaL8uw

[51] S. Zhang, J. Gong, and K. K. Lee, "A novel mobile robot localization approach based on a model switching feature extraction," in *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, Jul. 2012, pp. 1279–1284, iSSN: 2158-2297. [Online]. Available: https://ieeexplore.ieee.org/document/6360919

[52] M. Liu, X. Lei, S. Zhang, and B. Mu, "Natural landmark extraction in 2D laser data based on local curvature scale for mobile robot navigation," in *2010 IEEE International Conference on Robotics and Biomimetics*, Dec. 2010, pp. 525–530. [Online]. Available: https://ieeexplore.ieee.org/document/5723381

[53] P. Rosin, "Fitting superellipses," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 726–732, Aug. 2000.

[54] A. Jaklič, A. Leonardis, and F. Solina, *Segmentation and Recovery of Superquadrics*, 1st ed., ser. Computational Imaging and Vision. Springer Dordrecht, 2000, vol. 20.

[55] S. Chaudonneret, "Segmentation, localisation et cartographie avec primitives géométriques 2D," Master's thesis, Polytechnique Montréal, Dec. 2021. [Online]. Available: https://publications.polymtl.ca/9969/

[56] A. Venet, "Localisation et cartographie avec superquadriques comme primitives géométriques," Master's thesis, Polytechnique Montréal, Dec. 2019. [Online]. Available: https://publications.polymtl.ca/4175/

[57] C. Kanzow, N. Yamashita, and M. Fukushima, "Levenberg–Marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints," *Journal of Computational and Applied Mathematics*, vol. 172, no. 2, pp. 375–397, Dec. 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377042704001256

[58] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres solver." [Online]. Available: http://ceres-solver.org

[59] H. P. Gavin, "The levenberg-marquardt method for nonlinear least squares curve-fitting problems," 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:5708656

[60] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37–52, Aug. 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0169743987800849

[61] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[62] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, "The Mahalanobis distance," *Chemometrics and Intelligent Laboratory Systems*, vol. 50, no. 1, pp. 1–18, Jan. 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169743999000477

[63] F. C. Monteiro and A. Campilho, "Performance evaluation of image segmentation," *Image Analysis and Recognition*, pp. 248–259, 2006, accepted: 2010-02-11T12:23:40Z ISBN: 9783540448914 Publisher: Springer. [Online]. Available: https://bibliotecadigital.ipb.pt/handle/10198/1841

[64] T. Dang, "An Iterative Parameter Estimation Method for Observation Models with Nonlinear Constraints," *Metrology and Measurement Systems*, 2008. [Online]. Available: https://www.semanticscholar.org/paper/An-Iterative-Parameter-Estimation-Method-for-Models-Dang/7ab01a8c43fbb8f433b7f66f639b396d5a6261dd

[65] A. Leal, "autodiff," https://github.com/autodiff/autodiff, 2024, fix compilation warning: mismatch between integer types.

[66] G. Grisetti, C. Stachniss, and W. Burgard, "Gmapping," https://github.com/OpenSLAM-org/openslam_gmapping, accessed: 2024-06-23.