UNIVERSITY OF CALIFORNIA SAN DIEGO

DKF-SLAM: Distributed Kalman Filtering for Multi-Robot SLAM

A thesis submitted in partial satisfaction of the
requirements for the degree Master of Science

in

Electrical Engineering (Intelligent Systems, Robotics, and Control)

by

Sriram Shreedharan

Committee in charge:

Professor Nikolay A. Atanasov, Chair
Professor Dinesh Bharadia
Professor Florian Meyer

2023

The thesis of Sriram Shreedharan is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am extremely grateful to have worked with Professor Nikolay Atanasov during my master's, whose constant support and guidance made this thesis possible.

I would like to thank the committee members, Professor Dinesh Bharadia and Professor Florian Meyer, for their expertise and feedback on my research.

I also want to thank my group members from the Existential Robotics Lab, Hanwen Cao, Shrey Kansal, Shubham Kumar, and Kishore Nukala, for their invaluable insights throughout the research.

All the Chapters, in part, are currently being prepared for submission for publication of the material. Cao, Hanwen; Shreedharan, Sriram; Kansal, Shrey; Kumar, Shubham; Nukala, Kishore; Atanasov, Nikolay. The thesis author was one of the researchers/authors of this paper.

Section 3.3 in Chapter 3 is co-authored with Hanwen Cao. The thesis author was the primary author of this chapter.

ABSTRACT OF THE THESIS

DKF-SLAM: Distributed Kalman Filtering for Multi-Robot SLAM

by

Sriram Shreedharan

Master of Science in Electrical Engineering (Intelligent Systems, Robotics, and Control)

University of California San Diego, 2023

Professor Nikolay A. Atanasov, Chair

The simultaneous Localization and Mapping (SLAM) problem for single robot systems has been a topic of intense research for many years. However, in scenarios where a single robot is not sufficient to explore and reconstruct a large or complex environment, multi-robot SLAM can prove to be a promising solution. By working and communicating together, multiple robots can cover larger areas and improve the accuracy of localization and mapping.

A traditional approach for multi-robot SLAM includes using a central server, which communicates with each robot to exchange information and updates. However, this approach is not ideal for real-time scenarios, as it can cause delays and render the computation power of each agent futile. Moreover, if the central server fails, the whole system will break down.

The main aim of this thesis is to propose a distributed filtering algorithm to build an accurate sparse global map by leveraging the computation power of multiple robots and minimizing communication across them. The approach involves each robot estimating its state locally and sharing its observations with its neighbors. The robots must agree upon an estimate of the common observations, which is achieved through consensus optimization. The algorithm is inspired by the distributed stochastic mirror descent approach to solve a constrained variational inference problem that can be decomposed. The optimization algorithm ensures consistency among agents and their measurements and is systematically evaluated in both simulation and real-time datasets.

Chapter 1

Introduction

1.1 Perception for Robots

All living things are equipped with some form of perceptual system to understand and interact with the world surrounding them. One of the common perceptual systems is the visual system, which enables people to navigate their environment. With the advent of imaging sensors in the 20^{th} century, robots could finally see and interpret the world around them, giving rise to the fascinating field of Computer Vision. Over the past few decades, this technology has captured the attention of many researchers and enthusiasts alike, paving the way for exciting advancements in robotics and artificial intelligence.

The imaging sensor allows for the capture of 3D world points as 2D image points using a camera observation model. While it is easy to solve for the 2D image points given the 3D world points and camera location, the inverse problem of finding the camera trajectory when it moves through space and takes multiple images is more complex due to random sensor noise. This noise prevents an accurate estimate of the camera position. Additionally, knowing only the camera trajectory does not provide information about the environment or the robot's location within it. This is where localization comes in. However, a map is required to localize oneself in an environment, as we can say where we are only with reference to the map. For instance, imagine a completely dark and empty room; without any reference, it is impossible to say the location of where we are in the room. Assume there is a small lamp in the room. Then we can localize

ourselves with respect to the lamp. The lamp would serve as a map (or landmark). However, in real-life conditions, when a robot is exploring an unknown environment, it will not know if there will be objects or what kind of objects there will be, so it must create a map of the environment while keeping track of its location in the map. This is known as the simultaneous localization and mapping problem, and accurate localization and mapping is crucial for the robots to navigate and explore the environment safely.

1.2 Simultaneous Localization and Mapping

In recent years, the Simultaneous Localization and Mapping (SLAM) problem has gained much attention due to its wide range of applications, such as indoor robots and autonomous vehicles. The main objective of this problem is to estimate the robot's pose in an environment while simultaneously constructing a map of the surroundings. This might seem like a chicken and egg problem, as we need the map to locate the robot, and we need the robot's location to create the map. To solve this, we leverage various sensors that can track the robot's motion and obtain information about the environment. When the robot is equipped with only visual sensors such as a camera, this approach is called Visual SLAM. Visual-Inertial SLAM is another approach that combines visual sensors with inertial sensors, such as IMU, which provides information about the robot's motion. The data from the sensors is combined to estimate the robot and map state, and this can be done in two ways: smoothing or filtering. Smoothing involves estimating the entire trajectory and map state by combining all control inputs and observations. This is done by formulating the SLAM problem as a non-linear least squares problem and minimizing the error using iterative non-linear optimization techniques such as Levenberg-Marquardt or Gauss-Newton. On the other hand, filtering is a recursive technique for real-time systems where the current state is estimated based on the previous states and current observations. This thesis will focus on the filtering approach rather than smoothing.

1.2.1 Bayes Filter

Bayes filter is a general probabilistic inference technique commonly used in the SLAM paradigm for estimating the robot and map states by combining information about the control inputs and observations made by the robot of the environment. This approach models the robot states and the map as a Markov process where each state is only dependent on the previous state and the control input at that state, as shown in Fig. 1.1.

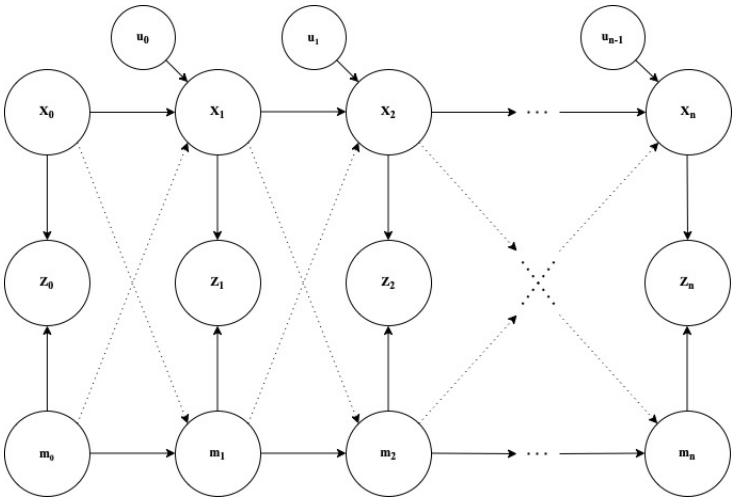


Figure 1.1. Markov Process Representation of the SLAM problem.

Markov Assumptions [2]:

1. The robot state x_{t+1} at time $t+1$ only depends on the previous state x_t and control input u_t at time t. It is independent of the history of states and inputs from time $0,...,t-1$
2. The map state m_{t+1} at time $t+1$ only depends on the previous map state m_t at time t.
3. The observations z_t at time t only depend on the robot state x_t and map state m_t at time t.
4. The robot state and map state might also affect each other's motion.

Given a motion model and observation model of the robot,

$$\begin{aligned} x_{t+1} &= f(x_t, u_t, w_t) \\ z_t &= h(x_t, m_t, v_t), \end{aligned} \tag{1.1}$$

where x_t and x_{t+1} are the robot states at time t and $t+1$ respectively, u_t is the control input, w_t is the motion noise, m_t is the map state, and v_t is the observation noise at time t.

The Bayes filter is a recursive estimation approach where the posterior (or updated) probability density function (pdf) of a state at time $t+1$ is computed using the posterior pdf at time t. The pdf is determined by combining the Markov assumptions with the Bayes rule.

Marginal Distribution: $p(x) = \int p(x|y)dy.$

Joint Distribution: $p(x,y) = p(y|x)p(x).$

Bayes Rule: $p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\int p(y|x)p(x)dx}.$

The posterior pdf is computed in two steps. First, the control input is incorporated in the previous state which gives the predicted pdf $p(x_{t+1}|z_{0:t}, u_{0:t})$, where $(z_{0:t}, u_{0:t})$ denotes $(z_0, z_1, ..., z_t, u_0, u_1, ..., u_t)$. Then the measurements or observations at that state is used to get the updated pdf $p(x_{t+1}|z_{0:t+1}, u_{0:t})$.

The posterior pdf (updated pdf) at time $t+1$ can be factorized as follows [2]:

$$
\begin{aligned}
& p(x_{t+1}|z_{0:t+1}, u_{0:t}) \\
&= \frac{1}{p(z_{t+1}|z_{0:t}, u_{0:t})} p(z_{t+1}|x_{t+1}, z_{0;t}, u_{0:t})\ p(x_{t+1}|z_{0:t}, u_{0:t}) && \text{(Bayes Rule)} \\
&= \frac{1}{p(z_{t+1}|z_{0:t}, u_{0:t})} p(z_{t+1}|x_{t+1})\ p(x_{t+1}|z_{0:t}, u_{0:t}) && \text{(Markov Assumption)} \\
&= \frac{1}{p(z_{t+1}|z_{0:t}, u_{0:t})} p(z_{t+1}|x_{t+1}) \int p(x_{t+1}, x_t|z_{0:t}, u_{0:t}) dx_t && \text{(Marginal Dist.)} \\
&= \frac{1}{p(z_{t+1}|z_{0:t}, u_{0:t})} p(z_{t+1}|x_{t+1}) \int p(x_{t+1}|z_{0:t}, u_{0:t}, x_t)\ p(x_t|z_{0:t}, u_{0:t}) dx_t && \text{(Joint Dist.)} \\
&= \frac{1}{p(z_{t+1}|z_{0:t}, u_{0:t})} p(z_{t+1}|x_{t+1}) \underbrace{\int p(x_{t+1}|x_t, u_t)\ \underbrace{p(x_t|z_{0:t}, u_{0:t-1})}_{\text{updated pdf at time } t} dx_t}_{\text{predicted pdf } p(x_{t+1}|z_{0:t}, u_{0:t})} && \text{(Markov Assumption)}
\end{aligned}
\tag{1.2}
$$

A special case of the Bayesian Recursive Estimation approach is the Kalman filter, where the pdf of the robot and map states are assumed to be Gaussian distributed random variables, and

the motion and observation models are assumed to be linear.

1.3 Towards Multi-agent Systems

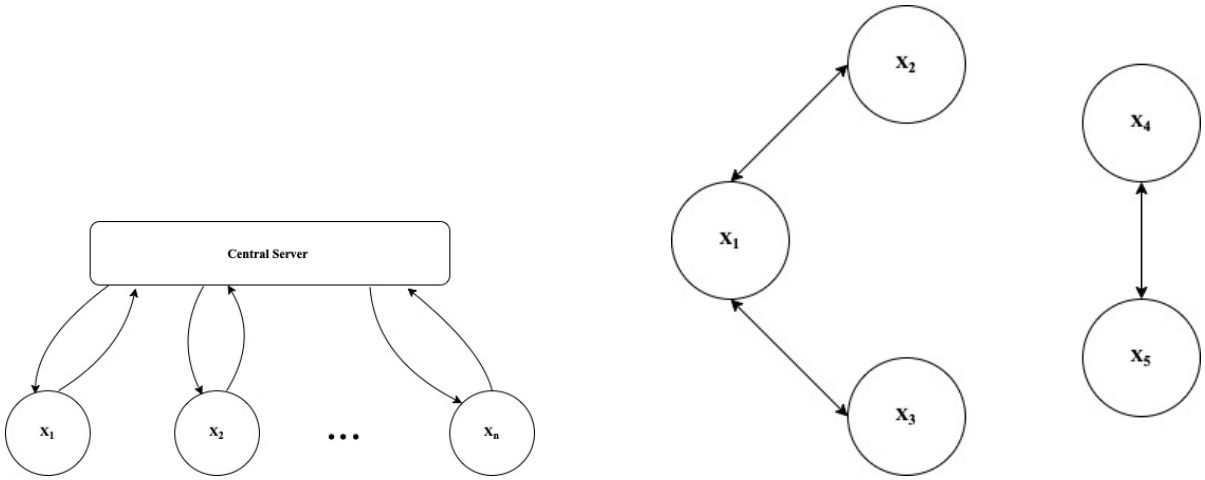
When a single-agent system can no longer solve or takes too much time to solve a problem or perform a task, multiple agents can be used to solve the problem. By leveraging the computation power from multiple agents, the problem can be solved in a distributed manner where each agent solves a smaller sub-problem. On top of this, multi-agent systems are fault-tolerant. If one agent fails, the task can still be divided among all the other agents. These advantages also apply to multi-robot systems where multiple robots can achieve certain objectives more efficiently than a single-robot system. This includes faster and more accurate mapping of a large area while localizing the robot in the environment.

There are mainly two approaches for multi-robot systems:

Centralized: In this approach, the robots communicate their states and observations to a central server or node, as shown in Fig. 1.2a, which builds a global map and sends commands to the robots. Due to the heavy computational load, this method is not feasible for large-scale real-time systems.

Decentralized: Instead, another approach is the decentralized / distributed approach, as shown in Fig. 1.2b, where the robots maintain their own local trajectories and local map while sharing data with their neighbors when they are in communication range. The information from these neighboring robots helps to locally optimize their state and the map of the environment.

This thesis introduces a distributed Kalman filter for multi-robot systems, by extending the single-robot Kalman filter with a consensus on the common landmarks observed across all robots. The algorithm is simple and efficient and builds an accurate sparse global map by sharing the information of the landmark states maintained by each robot with its neighbors.



(a) Centralized **(b)** Decentralized

Figure 1.2. Multi-Robot Approaches

1.4 Chapter Organization

The remainder of the thesis is organized as follows. Chapter 2 discusses the related work in single and multi-robot SLAM and the background in distributed consensus optimization. Chapter 3 presents the distributed Kalman filter algorithm. Chapter 4 discusses the application of the algorithm to multi-robot SLAM and evaluates its performance on a simulation and real-time dataset. Chapter 5 concludes this thesis and discusses future work.

All the Chapters, in part, are currently being prepared for submission for publication of the material. Cao, Hanwen; Shreedharan, Sriram; Kansal, Shrey; Kumar, Shubham; Nukala, Kishore; Atanasov, Nikolay. The thesis author was one of the researchers/authors of this paper.

Chapter 2

Related Work and Background

Single Robot SLAM

Single-robot SLAM has been extensively studied in recent years, and the most commonly used approaches are Visual and Visual-Inertial SLAM. ORB SLAM [28], [29] and OpenVINS [20] are popular real-time SLAM systems for visual and visual-inertial setup. SLAM is also mainly categorized into two based on the type of map generation: sparse SLAM and dense SLAM. Sparse maps are commonly represented by point clouds and landmarks, while dense maps are represented by occupancy grid, signed-distance field, and mesh.

Popular algorithms for sparse SLAM include the Kalman filter for linear motion and observation models, the extended Kalman filter (EKF), and the unscented Kalman filter (UKF) for non-linear motion and observation models, which models the robot state and landmarks as Gaussian random variables [5]. The Rao-Blackwellized Particle Filter [14] uses particles to approximate the robot state, while the landmarks are considered to be Gaussian distributed. There is also Factor Graph SLAM [13], which represents the robot states and landmarks as variables, and the pose constraints and observations are added as factors to the graph. Then a non-linear solver such as Levenberg-Marquardt is used to solve the non-linear least squares problem over all the states, which makes this a smoothing approach.

Some algorithms for Dense SLAM include FAST SLAM [27], where an occupancy map is created, and the particle filter is leveraged for accurate localization of the robot state. Kinect Fusion [31] and Voxblox [32], [37] use truncated signed distance field functions to create a

continuous or dense representation of the observed environment.

Multi-Robot SLAM

Extending single-robot SLAM to multiple robots has been a key challenge in this area and has gained less popularity than single-robot SLAM. The need for multi-robot comes from the shortcomings of single-robot SLAM when exploring large spaces. The exploration time can be significantly reduced by employing multiple robots in the environment. On top of this, multi-robot localization and mapping can help improve the overall localization accuracy of the robots and the mapping of the environment. However, these advantages also pose some challenges that need to be solved, such as,

1. How do these multiple robots need to explore the environment? If multiple robots explore the same space, the exploration time will increase.

2. What kind of information do the robots need to share with the others? The whole trajectory and map cannot be shared, as communicating such massive data in real-time is not possible.

3. What kind of communication system should be set up between the robots? Communication with a central server from all the robots is not possible for real-time systems; therefore, a decentralized approach would be better.

4. How to perform data association between robots? When multiple robots observe the same region of the environment, then the common features must be assigned the same global id across robots.

These challenges give rise to different areas of focus within the multi-robot or distributed SLAM problem. The papers [9], [10] deal with optimizing multi-robot exploration. The key problem is to find appropriate target points for individual robots so they can explore different regions of the environment simultaneously. This is done by calculating a common cost combining the reachability of the target point and the utility of the target point. A robot is assigned to a

target point based on least cost, and once a target point is assigned to a robot, the utility of the target point is reduced for the other robots. In this way, different target points are assigned for individual robots to explore, leading to faster exploration.

Another paper [17] proposes a collaborative method for improving only the localization accuracy by updating the robot's beliefs when one robot detects another. The idea here is that when one robot can find the relative pose of the other robot with respect to itself, then both robots can synchronize their beliefs and improve localization accuracy. Moreover, some robots can be equipped with high-cost (high accuracy) sensors and others with low-cost sensors. When the information is shared across multiple robots, the high-accuracy sensor data can be leveraged, thus improving the overall performance with fewer high-cost sensors.

In the papers [12], [15], [22], the authors propose different methods to merge the maps from multiple robots. [12] focuses on map matching based on landmarks from different sensors on multiple robots. A topologically correct single map is generated based on the commonly observed features by the individual robots. The algorithm finds a relative transformation from one robot to another by matching the landmarks observed from the environment and takes the transform that gives the highest matches. This transform is then used to merge the map. In [15], the relative positions between the robots are assumed to be known, and the feature estimates from all the robots are combined along with the robot's states as one state and updated as a whole. [22] uses 3D line features for matching across multiple robots. Each robot runs its own EKF SLAM and extracts line-based features from the environment. These features are then shared and matched when the robots are in communication range, and an initial guess of the relative pose is generated, which is used to do a rendezvous event to visually confirm the relative pose. Once a good estimate of the relative pose is determined, ICP is used to perform map merging and generate a global map.

An early popular work in multi-robot SLAM was by [16]. A real-time system was created for efficient exploration and mapping even when the initial robot poses were unknown. The data from the individual robots were combined into shared maps to coordinate exploration more

efficiently, and when combining the individual maps, the relative poses between the robots were consistently verified by the rendezvous method. The system was robust and produced identical maps on every run when tested on real-world conditions. However, this method fails to scale for a larger number of robots, as communicating and maintaining a shared map between many robots is not possible. Recently, Kimera-Multi [41] and DOOR-SLAM [25] were developed, which consider the states of multiple robots as variables of a factor graph. Each robot maintains individual factor graphs, and when they communicate, the relative pose is estimated along with loop closures, and the robot trajectories are updated in a distributed manner. Based on this globally optimized trajectory, the local maps maintained by the robots are updated, thus improving the accuracy of the trajectory estimate and map. These systems, too, however, fail to address the scalability to a larger number of robots.

Proposed Work

This thesis proposes an algorithm to improve localization and mapping accuracy by leveraging the common map or environment observations across the robots and imposing a consensus constraint on the same. Usually, this method has been used only for static networks [4], [35], [36] where the agents collectively try to minimize a common function, as discussed in the following sections, and this thesis aims to adapt this idea to a SLAM setting by deriving a distributed Kalman filter for multiple robots. Even though a centralized Kalman filter would work well for multiple robots, it is computationally expensive and may not be suitable for real-time applications. The aim of the distributed Kalman filter is for each robot to solve for its own local poses and map while also communicating its observations to its neighbors. By adding the consensus constraint, the robots agree upon the estimate of the common observations and a sparse global map is created with minimum communication while optimizing the robot's local poses. Furthermore, the distributed Kalman filter is fault-tolerant as the robots could still map the environment even if one of the robots failed. Overall, our algorithm improves the accuracy, scalability, and robustness while minimizing the computation and communication overheads.

2.1 Distributed Consensus Optimization

Given a connected graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ representing the agents and edges $\mathcal{E}$ representing the communication between the agents, in a distributed optimization problem, instead of minimizing a global function **F(x)** centrally, the task is split into a sum of local functions $\mathbf{F(x)} := \sum_{i=1}^{n} f_i(x_i)$ for each agent who collaboratively works to minimize the global function by exchanging information only with their neighbors $\mathbb{N}_i$. In this way, the computation is distributed between multiple agents leading to faster convergence. In the case of SLAM, each robot can maintain a pdf of their own poses and map, but all robots will have to agree upon a common pdf of the map. This imposes a consensus constraint on the distributed optimization problem leading to a constrained optimization problem: $\min_{x_i} \sum_{i=1}^{n} f_i(x_i)$ such that $x_i = x_j,\ \forall i, j \in$ **A** where **A** is the adjacency matrix. The distributed consensus optimization problem can be solved in multiple ways, like subgradient descent, double decomposition, and ADMM [21].

2.2 Optimization Methods

2.2.1 Distributed Subgradient

From [30], the optimization step in the distributed sub-gradient method is as follows:

$$x_i(k+1) = \sum_{j \in N_i} a_{ij}(k) x_j(k) - \alpha_i(k) \widetilde{\nabla} f_i(x_i(k)) \tag{2.1}$$

where $a_{ij}(k)$ are the weights corresponding to each neighbor, $\alpha_i(k)$ is the step size and $\widetilde{\nabla} f_i(x_i(k))$ is the subgradient of f_i at $x_i^{(k)}$.

The consensus optimization problem can be reformulated as follows:

$$\begin{aligned} &\min_{x_i} \sum_{i=1}^{n} f_i(x_i), \\ &\text{subject to} \sum_{j \in N_i} \frac{\mathbf{A}_{ij}}{2} \|x_j - x_i\|_2^2 \leq 0, \\ &\mathbf{A}_{ij} > 0,\ \forall j \in \mathbb{N}_i. \end{aligned} \tag{2.2}$$

The distributed subgradient method for this consensus optimization now follows the same update rule as shown in Eq. 2.1 but can be written in two steps : consensus step where the agent mixes its own information with its neighbors, and a local optimization step in the subgradient direction.

The consensus step is given as,

$$w_i(k+1) = \sum_{j \in N_i} \mathbf{A}_{ij} x_j(k). \tag{2.3}$$

The local gradient step is given by,

$$x_i(k+1) = w_i(k+1) - \alpha_i(k) \widetilde{\nabla} f_i(w_i(k+1)). \tag{2.4}$$

This method converges to the global optimum [30].

2.2.2 Decomposition

In optimization, when there's a separable problem

$$\begin{aligned} &\min f(x) = f_1(x_1) + f_2(x_2), \\ &\text{subject to } x_1 \in \mathbb{C},\ x_2 \in \mathbb{C}. \end{aligned} \tag{2.5}$$

x_1 and x_2 can be solved separately in parallel. Decomposition method is splitting the given problem into sub problems that can be solved independently of each other.

Primal Decomposition

The direct method of decomposition is Primal Decomposition [8] where the primal variables are manipulated. Consider the minimization problem,

$$\min f(x_1, x_2, y) = f_1(x_1, y) + f_2(x_2, y), \tag{2.6}$$

where y is called the coupling or complicating variable (primal variable) and when it's fixed, x_1 and x_2 can be solved separately by solving the two sub problems:

$$\begin{aligned} \phi_1(y) &= \min_{x_1} f_1(x_1, y) \\ \phi_2(y) &= \min_{x_2} f_2(x_2, y) \end{aligned} \tag{2.7}$$

$$\phi(y) = \phi_1(y) + \phi_2(y) \tag{2.8}$$

This is called the master problem, and the original optimization problem is rewritten in terms of this dual problem,

$$\min f(x_1, x_2, y) = \max_{y} \phi(y) \tag{2.9}$$

The master problem can then be solved using subgradient method or other convex optimization methods.

Dual Decomposition

The dual decomposition method solves the dual problem instead of the primal problem. The dual problem is considered because it is either unconstrained or has simple constraints [8].

Let's consider an equality constrained optimization problem,

$$\begin{aligned} &\min f(x), \\ &\text{subject to } Ax = b. \end{aligned} \tag{2.10}$$

The Lagrangian of the function is given by,

$$L(x,\lambda) = f(x) + \lambda^T (Ax - b). \tag{2.11}$$

Then the dual function and dual problem is given by,

$$\begin{aligned} g(\lambda) &= \inf_x L(x,\lambda), \\ &\max_\lambda g(\lambda). \end{aligned} \tag{2.12}$$

Finally, x can be determined by,

$$x^* = \operatorname*{argmin}_x L(x,\lambda^*). \tag{2.13}$$

Now say the optimization function is separable, then we can introduce local coupling variables $y1,\ y2$ and the problem can be reformulated as,

$$\begin{aligned} &\min f(x_1,x_2,y_1,y_2) = f_1(x_1,y_1) + f_2(x_2,y_2), \\ &\text{subject to } y_1 = y_2, \end{aligned} \tag{2.14}$$

where $y_1 = y_2$ is the consensus constraint.

The Lagrangian can now be written as,

$$L(x_1,x_2,y_1,y_2,\lambda) = f_1(x_1,y_1) + f_2(x_2,y_2) + \lambda^T (y_1 - y_2). \tag{2.15}$$

This leads to the dual problem as the function is separable,

$$\begin{aligned} g_1(\lambda) &= \inf_{x_1,y_1} f_1(x_1,y_1) + \lambda^T y_1, \\ g_2(\lambda) &= \inf_{x_2,y_2} f_2(x_2,y_2) + \lambda^T y_2, \\ \max\, g(\lambda) &= g_1(\lambda) + g_2(\lambda). \end{aligned} \tag{2.16}$$

These dual subproblems can now be solved in parallel.

2.2.3 Alternating Direction Method of Multipliers

To robustify ascent of dual problem, the augmented lagrangian is considered which converges faster [8].

The objective function is rewritten as,

$$\begin{aligned} &\min\, f(x) + g(z), \\ &\text{subject to } Ax + Bz = 0. \end{aligned} \tag{2.17}$$

The augmented Lagrangian is given by,

$$L_\rho(x,z,\lambda) = f(x) + g(z) + \lambda^T(Ax+Bz) + \frac{\rho}{2}\|Ax+Bz\|_2^2. \tag{2.18}$$

The ADMM update is then performed as follows,

$$\begin{aligned} x^{k+1} &= \operatorname*{argmin}_x L_\rho(x,z^k,\lambda^k) \quad \text{(Fix } z \text{ and } \lambda \text{ and minimize } x\text{)}, \\ z^{k+1} &= \operatorname*{argmin}_x L_\rho(x^{k+1},z,\lambda^k) \quad \text{(Fix } x \text{ and } \lambda \text{ and minimize } z\text{)}, \\ \lambda^{k+1} &= \lambda^k + \rho(Ax^{k+1} + Bz^{k+1}) \quad \text{(Fix } x \text{ and } z \text{ and update dual variable } \lambda\text{)}. \end{aligned} \tag{2.19}$$

2.3 Gaussian Variational Inference

Variational Inference is a method used to approximate the posterior probability distribution of an unknown variable given its prior and a set of measurements. If the approximation is determined using only Gaussian distribution, it is known as Gaussian Variational Inference [6]. One of the most common variational inference problems is minimizing the Kullback-Leibler (KL) divergence between the approximated and true posterior distributions. Let the true posterior be denoted by $p(x|z)$ where x is the unknown state we are trying to estimate and z is the measurement. Let $q(x) = \mathcal{N}(\mu, \Sigma)$ be the approximation of the posterior distribution.

The KL divergence between the true posterior and approximation is given by,

$$\begin{aligned} \mathrm{KL}(q||p) &= \int_{-\infty}^{\infty} q(x) \ln \frac{p(x|z)}{q(x)} dx \\ &= \mathbb{E}_q[\log(q(x)) - \log(p(x|z))], \end{aligned} \tag{2.20}$$

where $\mathbb{E}_q$ is the expectation over the probability distribution of q.

$$\mathrm{KL}(q||p) = \mathbb{E}_q[-\log(p(x,z))] - \underbrace{\frac{1}{2}\log((2\pi \exp)^N |\Sigma|)}_{\text{Gaussian entropy}} + \underbrace{\log(p(z))}_{\text{constant}}. \tag{2.21}$$

From this, the function to be minimized is given by,

$$V(q) = \mathbb{E}_q[\phi(x)] + \frac{1}{2}\log(|\Sigma|^{-1}), \tag{2.22}$$

where $\phi(x) = -\log(p(x,z))$.

It can also be seen that $V(q)$ is the negative of the Evidence Lower Bound (ELBO) [23] and by minimizing $V(q)$, ELBO is maximized.

All the Chapters, in part, are currently being prepared for submission for publication of the material. Cao, Hanwen; Shreedharan, Sriram; Kansal, Shrey; Kumar, Shubham; Nukala, Kishore; Atanasov, Nikolay. The thesis author was one of the researchers/authors of this paper.

Chapter 3

Methodology

3.1 Overview

This chapter focuses on deriving a distributed Kalman filter using consensus optimization and is inspired from [4], [35], [36]. The overall idea is to form a centralized objective that can be decomposed and solved by individual robots. Further, to enable consensus among the robots about the common landmark states $\mathbf{y}$, we consider the KL divergence between the robot's distribution and its neighbor's distribution. The derivation is mainly divided into two sections. In the first section, the posterior pdf parameters are approximated by considering the robots to be static, and only the common landmark states are observed. The second section presents the distributed filtering algorithm by considering the joint probability of both the robots and the common landmark states along with the consensus constraint.

3.2 Problem Statement

Consider the simultaneous localization and mapping problem where each robot's motion is described by the linear motion model given by,

$$\mathbf{x}_{i,t+1} = \mathbf{A}_i\, \mathbf{x}_{i,t} + \mathbf{B}_i\, \mathbf{u}_{i,t} + \mathbf{w}_{i,t} \qquad \mathbf{w}_i, t \sim \mathcal{N}(\mathbf{0}, \mathbf{W}_i), \tag{3.1}$$

where $\mathbf{x}_{i,t}$ and $\mathbf{x}_{i,t+1}$ are the states of the robot i at time t and $t+1$ respectively, $\mathbf{u}_{i,t}$ is the control input for robot i at time t, and $\mathbf{w}_{i,t}$ is the zero-mean Gaussian motion noise of robot i with covariance $\mathbf{W}_i$ at time t.

The robot obtains observations of the common landmarks $\mathbf{y}$ at each time step through the following observation model,

$$\mathbf{z}_{i,t} = \begin{bmatrix} \mathbf{C}_{i,y} & \mathbf{C}_{i,x} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{x}_{i,t} \end{bmatrix} + \mathbf{v}_{i,t} \qquad \mathbf{v}_i, t \sim \mathcal{N}(\mathbf{0}, \mathbf{V}_i), \tag{3.2}$$

where $\mathbf{z}_{i,t}$ is the observation obtained by the robot i at time t, and $\mathbf{v}_{i,t}$ is the zero-mean Gaussian measurement noise of robot i with covariance $\mathbf{V}_i$ at time t.

The aim is to estimate the states of the individual robots $\mathbf{x}_i$ while also collectively estimating the common landmark states $\mathbf{y}$, in other words, estimate the parameters that maximize the joint probability of the robot and landmark states given the control inputs and measurements $\underset{\mathbf{x}_i, \mathbf{y}}{\max} p(\mathbf{x}_{i,0:t}, \mathbf{y} | \mathbf{u}_{0:t-1}, \mathbf{z}_{0:t})$. The communication between the robots is modeled as a graph $\mathcal{G}$ with the robots as the nodes $\mathcal{V}$ and the connection between them as edges $\mathcal{E}$. The graph is represented as an adjacency matrix $\mathbf{A}$. If nodes i and j are connected, that is, if robots i and j can communicate, then $\mathbf{A}_{ij} > 0$, else $\mathbf{A}_{ij} = 0$.

3.3 Mapping only

First, let us ignore the motion model and consider only the observation model for the common landmark states $\mathbf{y}$. The observation model is now defined as,

$$\mathbf{z} = \mathbf{C}\,\mathbf{y} + \mathbf{v} \qquad \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{V}). \tag{3.3}$$

Now, the problem is reduced to estimating only the common parameter $\mathbf{y}$ based on the robot's own observations and the estimates from its neighbors.

3.3.1 Single-Robot Case

Before moving onto multiple robots, the objective function of a single robot is considered which is the same as discussed in Section 2.3.

$$\begin{aligned} &\min_{q} V(q) \\ &= \min_{q} \mathbb{E}_q[-\log p(\mathbf{y},\mathbf{z})] + \frac{1}{2}\log(|\mathbf{\Sigma}^{-1}|). \end{aligned} \tag{3.4}$$

The posterior distribution of $\mathbf{y}$ conditioned on observation $\mathbf{z}$ is given by Bayes rule,

$$p(\mathbf{y}|\mathbf{z}) = \frac{p(\mathbf{z}|\mathbf{y})p(\mathbf{y})}{p(\mathbf{z})} \propto p(\mathbf{z}|\mathbf{y})p(\mathbf{y}). \tag{3.5}$$

Taking the logarithm of Eq. 3.5,

$$\begin{aligned} \log p(\mathbf{y}|\mathbf{z}) \propto &-\frac{1}{2}(\mathbf{z}-\mathbf{C}\mathbf{y})^\top \mathbf{V}^{-1}(\mathbf{z}-\mathbf{C}\mathbf{y}) \\ &-\frac{1}{2}(\mathbf{y}-\check{\boldsymbol{\mu}})^\top \check{\mathbf{\Sigma}}^{-1}(\mathbf{y}-\check{\boldsymbol{\mu}}), \end{aligned} \tag{3.6}$$

where $\check{\boldsymbol{\mu}}$ and $\check{\mathbf{\Sigma}}$ are the mean and covariance of the prior distribution $p(\mathbf{y})$.

Stochastic Mirror Descent (SMD) Formulation

The SMD algorithm is a generalization of the Stochastic Gradient Descent (SMD) where the Bregman divergence operator is introduced for iterative minimization of a convex optimization problem [8], [35].

Considering an optimization problem given by,

$$\min_{\mathbf{x}} \mathbb{E}[f(\mathbf{x};\mathbf{y})] \approx \frac{1}{T}\sum_{t=1}^{T} f(\mathbf{x};\mathbf{y}_t). \tag{3.7}$$

The gradient ∇f is computed at time t and the iterative optimization is performed as follows,

$$\mathbf{x}_{t+1} = \underset{\mathbf{x}}{\operatorname{argmin}} \left\{ \langle \alpha_t \nabla f(\mathbf{x}_t,\mathbf{y}_t), \mathbf{x} \rangle + D_\psi(\mathbf{x},\mathbf{x}_t) \right\}, \tag{3.8}$$

where $\langle(.,.)\rangle$ is the inner product and D_ψ is a Bregman divergence between $\mathbf{x}$ and $\mathbf{x}_t$.

The objective function in Eq. 3.4 is a stochastic optimization problem and can be rewritten in terms of the SMD algorithm with constant step size $\alpha = 1$.

$$\begin{aligned} q^* &\in \operatorname*{argmin}_q V(q) \\ &= \operatorname*{argmin}_q \mathbb{E}_q[-\log p(\mathbf{y},\mathbf{z})] + \frac{1}{2}\log(|\mathbf{\Sigma}^{-1}|) \\ &= \operatorname*{argmin}_q \mathbb{E}_q[-\log p(\mathbf{z}|\mathbf{y}) - \log p(\mathbf{y})] + \mathbb{E}_q[\log q(\mathbf{y})] \\ &= \operatorname*{argmin}_q \mathbb{E}_q[-\log p(\mathbf{z}|\mathbf{y})] + \mathbb{E}_q[\log q(\mathbf{y})] - \mathbb{E}_q[\log p(\mathbf{y})] \\ &= \operatorname*{argmin}_q \mathbb{E}_q[-\log p(\mathbf{z}|\mathbf{y})] + \mathrm{KL}(q(\mathbf{y})||p(\mathbf{y})). \end{aligned} \tag{3.9}$$

This minimization problem is similar to the SMD algorithm in Eq. 3.8 and is defined over the space of probability density functions. Therefore the standard inner product is the Expectation $\langle f_1, f_2\rangle = \int f_1 f_2$ and Bregman divergence reduces to KL divergence. This optimization problem has a closed form solution given by [36],

$$q^* \propto p(\mathbf{z}|\mathbf{y})\ p(\mathbf{y}). \tag{3.10}$$

Proof [36]. This is an equality constrained problem with the constraint being $\int q = 1$. Relaxing the constraint with Lagrangian multiplier, leads to,

$$L(q,\lambda) = \mathbb{E}_q[-\log p(\mathbf{z}|\mathbf{y})] + \mathrm{KL}(q(\mathbf{y})||p(\mathbf{y})) + \lambda(\mathbb{E}_{\mathbb{1}}[q] - 1). \tag{3.11}$$

The derivative of the right hand side with respect to q is given by,

$$\frac{\partial L}{\partial q} = -\log p(\mathbf{z}|\mathbf{y}) + (1 + \log q(y) - \log p(\mathbf{y})) + \lambda. \tag{3.12}$$

Setting this equal to 0,

$$\begin{aligned} \log q(\mathbf{y}) &= \log p(\mathbf{z}|\mathbf{y}) + \log p(\mathbf{y}) - 1 - \lambda, \\ q(\mathbf{y}) &= \exp(-1 - \lambda + \log p(\mathbf{z}|\mathbf{y}))\ p(\mathbf{y}). \end{aligned} \tag{3.13}$$

The constraint $\int q = 1$ can be used to solve for λ.

$$\begin{aligned} &\exp(-1-\lambda) \underbrace{\int \exp(\log p(\mathbf{z}|\mathbf{y}))\ p(\mathbf{y})}_{\Gamma} = 1, \\ &\exp(-1-\lambda) = \frac{1}{\Gamma}. \end{aligned} \tag{3.14}$$

Substituting Eq. 3.14 in Eq. 3.13 leads to,

$$q(\mathbf{y}) = \frac{1}{\Gamma}\ p(\mathbf{z}|\mathbf{y})\ p(\mathbf{y}). \tag{3.15}$$

□

In this case, the posterior distribution is approximated by a Gaussian and the corresponding optimal parameters is given by the following equations,

$$\log q(\mathbf{y}) \propto \log p(\mathbf{z}|\mathbf{y}) + \log p(\mathbf{y}). \tag{3.16}$$

$$\begin{aligned} \log q(\mathbf{y}) &\propto -\frac{1}{2}(\mathbf{y}-\mu^*)^T \mathbf{\Sigma}^{*^{-1}} (\mathbf{y}-\mu^*) \\ &\propto -\frac{1}{2}(\mathbf{y}^T \mathbf{\Sigma}^{*^{-1}} \mathbf{y}) + (\mathbf{y}^T \mathbf{\Sigma}^{*^{-1}} \boldsymbol{\mu}^*) - \frac{1}{2}(\boldsymbol{\mu}^{*^T} \mathbf{\Sigma}^{*^{-1}} \boldsymbol{\mu}^*). \end{aligned} \tag{3.17}$$

$$
\begin{aligned}
\log p(\mathbf{z}|\mathbf{y}) + \log p(\mathbf{y}) &\propto -\frac{1}{2}(\mathbf{z}-\mathbf{C}\mathbf{y})^T\mathbf{V}^{-1}(\mathbf{z}-\mathbf{C}\mathbf{y}) \\
&\quad -\frac{1}{2}(\mathbf{y}-\check{\boldsymbol{\mu}})^T\check{\boldsymbol{\Sigma}}^{-1}(\mathbf{y}-\check{\boldsymbol{\mu}}) \\
&\propto -\frac{1}{2}(\mathbf{z}^T\mathbf{V}^{-1}\mathbf{z}) + (\mathbf{y}^T\mathbf{C}^T\mathbf{V}^{-1}\mathbf{z}) - \frac{1}{2}(\mathbf{y}^T\mathbf{C}^T\mathbf{V}^{-1}\mathbf{C}\mathbf{y}) \\
&\quad -\frac{1}{2}(\mathbf{y}^T\check{\boldsymbol{\Sigma}}^{-1}\mathbf{y}) + (\mathbf{y}^T\check{\boldsymbol{\Sigma}}^{-1}\check{\boldsymbol{\mu}}) - \frac{1}{2}(\check{\boldsymbol{\mu}}^T\check{\boldsymbol{\Sigma}}^{-1}\check{\boldsymbol{\mu}}) \\
&\propto -\frac{1}{2}\mathbf{y}^T(\check{\boldsymbol{\Sigma}}^{-1} + \mathbf{C}^T\mathbf{V}^{-1}\mathbf{C})\mathbf{y} + \mathbf{y}^T(\check{\boldsymbol{\Sigma}}^{-1}\check{\boldsymbol{\mu}} + \mathbf{C}^T\mathbf{V}^{-1}\mathbf{z}) \\
&\quad -\frac{1}{2}(\mathbf{z}^T\mathbf{V}^{-1}\mathbf{z} + \check{\boldsymbol{\mu}}^T\check{\boldsymbol{\Sigma}}^{-1}\check{\boldsymbol{\mu}}).
\end{aligned}
\tag{3.18}
$$

Comparing Eq. 3.17 and 3.18, we get,

$$
\begin{aligned}
\boldsymbol{\Sigma}^{*^{-1}} &= \check{\boldsymbol{\Sigma}}^{-1} + \mathbf{C}^\top\mathbf{V}^{-1}\mathbf{C}, \\
\boldsymbol{\Sigma}^{*^{-1}}\boldsymbol{\mu}^* &= \check{\boldsymbol{\Sigma}}^{-1}\check{\boldsymbol{\mu}} + \mathbf{C}^\top\mathbf{V}^{-1}\mathbf{z},
\end{aligned}
\tag{3.19}
$$

which then leads to the Kalman filter update equation [5].

Alternative Formulation

The posterior distribution parameters can also be approximated using the method described in [6]. Let $\phi(\mathbf{y}) = -\log p(\mathbf{y},\mathbf{z})$, the derivatives of the objective function with respect to the Gaussian parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ is determined using Stein's lemma [40] and from [34]. Stein's lemma says that, if $\mathbf{y}$ is a normally distributed random variable, and f is a differentiable function, then we have,

$$
\mathbb{E}_q[(\mathbf{y}-\boldsymbol{\mu})f(\mathbf{y})] = \boldsymbol{\Sigma}\mathbb{E}_q[\frac{\partial f(\mathbf{y})}{\partial \mathbf{y}^\top}].
\tag{3.20}
$$

This leads to the following set of derivatives,

$$\frac{\partial V(q)}{\partial \boldsymbol{\mu}^\top} = \mathbb{E}_q[\frac{\partial \boldsymbol{\phi}(\mathbf{y})}{\partial \mathbf{y}^T}] = \boldsymbol{\Sigma}^{-1}\mathbb{E}_q[(\mathbf{y}-\boldsymbol{\mu})\boldsymbol{\phi}(\mathbf{y})], \tag{3.21a}$$

$$\frac{\partial^2 V(q)}{\partial \boldsymbol{\mu}^\top \partial \boldsymbol{\mu}} = \boldsymbol{\Sigma}^{-1}\mathbb{E}_q[(\mathbf{y}-\boldsymbol{\mu})(\mathbf{y}-\boldsymbol{\mu})^\top \boldsymbol{\phi}(\mathbf{y})]\boldsymbol{\Sigma}^{-1} \tag{3.21b}$$

$$-\boldsymbol{\Sigma}^{-1}\mathbb{E}_q[\boldsymbol{\phi}(\mathbf{y})], \tag{3.21b}$$

$$\frac{\partial V(q)}{\partial \boldsymbol{\Sigma}^{-1}} = -\frac{1}{2}\mathbb{E}_q[(\mathbf{y}-\boldsymbol{\mu})(\mathbf{y}-\boldsymbol{\mu})^\top \boldsymbol{\phi}(\mathbf{y})] \tag{3.21c}$$

$$+\frac{1}{2}\boldsymbol{\Sigma}\mathbb{E}_q[\boldsymbol{\phi}(\mathbf{y})] + \frac{1}{2}\Sigma. \tag{3.21c}$$

Comparing Eq. 3.21b and 3.21c,

$$\frac{\partial^2 V(q)}{\partial \boldsymbol{\mu}^\top \partial \boldsymbol{\mu}} = \boldsymbol{\Sigma}^{-1} - 2\boldsymbol{\Sigma}^{-1}\frac{\partial V(q)}{\partial \boldsymbol{\Sigma}^{-1}}\boldsymbol{\Sigma}^{-1}. \tag{3.22}$$

Using Eq. 3.20 along with the fact,

$$p(\mathbf{y},\mathbf{z}) = p(\mathbf{y}|\mathbf{z})p(\mathbf{z}) \propto p(\mathbf{y}|\mathbf{z}), \tag{3.23}$$

we get,

$$\begin{aligned}\frac{\partial V(q)}{\partial \boldsymbol{\mu}^\top} &= \mathbb{E}_q[\frac{\partial \boldsymbol{\phi}(\mathbf{y})}{\partial \mathbf{y}^\top}] \\ &= \mathbb{E}_q[\check{\boldsymbol{\Sigma}}^{-1}(\mathbf{y}-\check{\boldsymbol{\mu}}) - \mathbf{C}^\top\mathbf{V}^{-1}(\mathbf{z}-\mathbf{C}\mathbf{y})] \\ &= \check{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{\mu}-\check{\boldsymbol{\mu}}) - \mathbf{C}^\top\mathbf{V}^{-1}(\mathbf{z}-\mathbf{C}\boldsymbol{\mu}).\end{aligned} \tag{3.24}$$

The second derivative with respect to μ is then given by,

$$\frac{\partial^2 V(q)}{\partial \boldsymbol{\mu}^\top \partial \boldsymbol{\mu}} = \check{\boldsymbol{\Sigma}}^{-1} + \mathbf{C}^\top\mathbf{V}^{-1}\mathbf{C}. \tag{3.25}$$

By setting $\frac{\partial V(q)}{\partial \Sigma^{-1}} = 0$ and using Eq. 3.22,

$$\boldsymbol{\Sigma}^{-1} = \frac{\partial^2 V(q)}{\partial \boldsymbol{\mu}^\top \partial \boldsymbol{\mu}} = \check{\boldsymbol{\Sigma}}^{-1} + \mathbf{C}^\top \mathbf{V}^{-1} \mathbf{C}. \tag{3.26}$$

Substitute the above equation to Eq. 3.24 and setting $\frac{\partial V(q)}{\partial \boldsymbol{\mu}^\top} = 0$,

$$\boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} = \check{\boldsymbol{\Sigma}}^{-1} \check{\boldsymbol{\mu}} + \mathbf{C}^\top \mathbf{V}^{-1} \mathbf{z}. \tag{3.27}$$

This is the same as the optimal Gaussian parameters obtained through the SMD formulation as shown in Eq. 3.19.

3.3.2 Multi-robot case

In the case of multiple robots, each robot maintains an individual pdf of the common landmark $\mathbf{y}$, denoted by $q_i(\mathbf{y})$. The consensus constraint is then imposed on these distributions, saying that the distribution maintained by each robot for the common state needs to be the same, that is, the KL divergence between the distributions should be 0.

Distributed SMD Formulation

The observations made by the robots are assumed to be independent and thus the centralized objective $V(q)$ in Eq. 3.9 can be decomposed as a separable problem [36],

$$V(q) = \sum_{i \in \mathcal{V}} V_i(q_i). \tag{3.28}$$

The robot's individually minimize their own local objective while also maintaining consensus of the common landmarks across all the robots $p_i(\mathbf{y}) = p_j(\mathbf{y})$. The consensus constraint is added into the objective as minimizing the KL-divergence between robot i's distribution and

the prior distribution of its neighbor's $\mathbb{N}_i$.

$$\begin{aligned} q_i^* &\in \underset{q_i}{\operatorname{argmin}}\, V_i(q_i) \\ &= \underset{q_i}{\operatorname{argmin}}\, \mathbb{E}_{q_i}[-\log p_i(\mathbf{z}_i|\mathbf{y})] + \sum_{j\in\mathbb{N}_i} \mathbf{A}_{ij}\mathrm{KL}(q_i(\mathbf{y})||p_j(\mathbf{y})). \end{aligned} \tag{3.29}$$

The closed form solution to this is given by [36],

$$q_i^* \propto p_i(\mathbf{z}_i|\mathbf{y}) \prod_{j\in\mathbb{N}_i} p_j^{\mathbf{A}_{ij}}(\mathbf{y}). \tag{3.30}$$

Proof [36]. Similar to the single robot case, this is again an equality constrained problem with the constraint being $\int q_i = 1$. Relaxing the constraint with Lagrangian multiplier, leads to,

$$L(q_i, \lambda) = \mathbb{E}_{q_i}[-\log p_i(\mathbf{z}_i|\mathbf{y})] + \sum_{j\in\mathbb{N}_i} \mathbf{A}_{ij}\mathrm{KL}(q_i(\mathbf{y})||p_j(\mathbf{y})) + \lambda(\mathbb{E}_{\mathbb{1}}[q_i] - 1). \tag{3.31}$$

The derivative of the right hand side with respect to q_i is given by,

$$\frac{\partial L}{\partial q_i} = -\log p_i(\mathbf{z}_i|\mathbf{y}) + (1 + \log q_i(\mathbf{y}) - \sum_{j\in\mathbb{N}_i} \mathbf{A}_{ij}\log p_j(\mathbf{y})) + \lambda. \tag{3.32}$$

Setting this equal to 0,

$$\begin{aligned} \log q_i(\mathbf{y}) &= \log p_i(\mathbf{z}_i|\mathbf{y}) + \sum_{j\in\mathbb{N}_i} \mathbf{A}_{ij}\log p_j(\mathbf{y}) - 1 - \lambda, \\ q_i(\mathbf{y}) &= \exp(-1 - \lambda + \log p_i(\mathbf{z}_i|\mathbf{y}_i)) \prod_{j\in\mathbb{N}_i} p_j^{\mathbf{A}_{ij}}(\mathbf{y}). \end{aligned} \tag{3.33}$$

The constraint $\int q_i = 1$ can be used to solve for λ.

$$\begin{aligned} \exp(-1-\lambda) \underbrace{\int \exp(\log p_i(\mathbf{z}_i|\mathbf{y})) \prod_{j\in\mathbb{N}_i} p_j^{\mathbf{A}_{ij}}(\mathbf{y})}_{\Gamma_i} &= 1, \\ \exp(-1-\lambda) &= \frac{1}{\Gamma_i}. \end{aligned} \tag{3.34}$$

Substituting Eq. 3.34 in Eq. 3.33 leads to,

$$q_i(\mathbf{y}) = \frac{1}{\Gamma_i}\, p_i(\mathbf{z}_i|\mathbf{y}) \prod_{j\in\mathbb{N}_i} p_j^{\mathbf{A}_{ij}}(\mathbf{y}). \tag{3.35}$$

□

Define $\bar{q} = \prod_{j\in\mathbb{N}_i} p_j^{\mathbf{A}_{ij}}(\mathbf{y})$ with the corresponding Gaussian parameters mean $\bar{\boldsymbol{\mu}}$ and covariance $\bar{\boldsymbol{\Sigma}}$. Given a set of random variables with Gaussian distribution denoted by $p_j = \mathcal{G}(\boldsymbol{\nu}_j, \boldsymbol{\Omega}_j)$, where $\boldsymbol{\nu}_j$ is the information vector and $\boldsymbol{\Omega}_j$ is the information matrix, the weighted geometric mean of the individual pdfs $\prod_j p_j^{\kappa_j}$ (κ_j is the corresponding weight) is proportional to the pdf with distribution $\mathcal{G}(\sum_j \kappa_j \boldsymbol{\nu}_j, \sum_j \kappa_j \boldsymbol{\Omega}_j)$ [3].

$$\prod_j p_j^{\kappa_j} \sim \mathcal{G}(\sum_j \kappa_j \boldsymbol{\nu}_j, \sum_j \kappa_j \boldsymbol{\Omega}_j) \tag{3.36}$$

This leads to the following equations for $\bar{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{\Sigma}}$,

$$\begin{aligned} \bar{\boldsymbol{\Sigma}}^{-1} &= \sum_{j\in\mathbb{N}_i} \mathbf{A}_{ij} \check{\boldsymbol{\Sigma}}_j^{-1}, \\ \bar{\boldsymbol{\Sigma}}^{-1} \bar{\boldsymbol{\mu}} &= \sum_{j\in\mathbb{N}_i} \mathbf{A}_{ij} \check{\boldsymbol{\Sigma}}_j^{-1} \check{\boldsymbol{\mu}}_j. \end{aligned} \tag{3.37}$$

Similar to the single robot case, the posterior distribution is approximated by a Gaussian

and the corresponding optimal parameters is given by,

$$\begin{aligned} \boldsymbol{\Sigma}_i^{*^{-1}} &= \bar{\boldsymbol{\Sigma}}^{-1} + \mathbf{C}_i^\top \mathbf{V}_i^{-1} \mathbf{C}_i, \\ \boldsymbol{\Sigma}_i^{*^{-1}} \boldsymbol{\mu}_i^* &= \bar{\boldsymbol{\Sigma}}^{-1} \bar{\boldsymbol{\mu}} + \mathbf{C}_i^\top \mathbf{V}_i^{-1} \mathbf{z}_i. \end{aligned} \tag{3.38}$$

By iteratively averaging the information mean and information matrix, consensus is achieved across all robots with respect to the pdf of the common landmark states $\mathbf{y}$ [36].

Alternate Formulation

Again, the posterior distribution parameters can be similarly derived as in the single robot case. Consider the following constrained optimization problem,

$$\begin{aligned} &\min_{q_i} \sum_{i=1}^{n} V_i(q_i), \\ &\text{subject to} \sum_{j \in N_i} \mathbf{A}_{ij} \mathrm{KL}(q_i(\mathbf{y}) || q_j(\mathbf{y})) = 0, \\ &\mathbf{A}_{ij} > 0, \ \forall j \in \mathbb{N}_i. \end{aligned} \tag{3.39}$$

The objective function similar to the SMD algorithm in Eq. 3.9 is considered with $\alpha = 1$,

$$\min_q \mathbb{E}_q[-\log p(\mathbf{z}|\mathbf{y})] + \alpha \mathrm{KL}(q(\mathbf{y}) || p(\mathbf{y})). \tag{3.40}$$

Using Eq. 3.40 in Eq. 3.39 and relaxing the constraint with Lagrangian multiplier, the cost function for each individual robot is the dual problem given by,

$$\begin{aligned} &V_i(q_i) \\ &= \mathbb{E}_{q_i}[-\log p(\mathbf{z}_i|\mathbf{y})] + \alpha \mathrm{KL}(q_i(\mathbf{y}) || p(\mathbf{y})) + \lambda \sum_{j \in \mathbb{N}_i} \mathbf{A}_{ij} \mathrm{KL}(q_i(\mathbf{y}) || q_j(\mathbf{y})), \end{aligned} \tag{3.41}$$

which can be rewritten in the following form,

$$
\begin{aligned}
& V_i(q_i) \\
&= \mathbb{E}_{q_i}[-\log p(\mathbf{z}_i|\mathbf{y})] + (\alpha+\lambda-\lambda\mathbf{A}_{ii}-1)\mathbb{E}_{q_i}[\log q_i(\mathbf{y})] \\
&\quad + \mathrm{KL}(q_i(\mathbf{y})||p^{\alpha}(\mathbf{y}) \prod_{j\in\mathbb{N}_i\setminus\{i\}} q_j^{\lambda\mathbf{A}_{ij}}(\mathbf{y})).
\end{aligned} \tag{3.42}
$$

Define $\bar{q}(\mathbf{y}) = p^{\alpha}(\mathbf{y})\prod_{j\in\mathbb{N}_i\setminus\{i\}} q_j^{\lambda\mathbf{A}_{ij}}(\mathbf{y})$ with mean and covariance $\bar{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{\Sigma}}$. Using Eq. 3.36, it leads to the following equations for $\bar{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{\Sigma}}$,

$$
\begin{aligned}
\bar{\boldsymbol{\Sigma}}^{-1} &= \alpha\check{\boldsymbol{\Sigma}}^{-1} + \lambda \sum_{j\in\mathbb{N}_i\setminus\{i\}} \mathbf{A}_{ij}\boldsymbol{\Sigma}_j^{-1}, \\
\bar{\boldsymbol{\Sigma}}^{-1}\bar{\boldsymbol{\mu}} &= \alpha\check{\boldsymbol{\Sigma}}^{-1}\check{\boldsymbol{\mu}} + \lambda \sum_{j\in\mathbb{N}_i\setminus\{i\}} \mathbf{A}_{ij}\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j.
\end{aligned} \tag{3.43}
$$

Let $\phi_i(\mathbf{y}_i) = -\log p(\mathbf{z}_i|\mathbf{y})$, the derivatives of $V_i(q_i)$ are computed again similar to Eq. 3.21 using Stein's lemma from Eq. 3.20 and [34],

$$
\begin{aligned}
\frac{\partial V_i(q_i)}{\partial \boldsymbol{\mu}_i^{\top}} &= \boldsymbol{\Sigma}_i^{-1}\mathbb{E}_{q_i}[(\mathbf{y}-\boldsymbol{\mu}_i)\phi_i(\mathbf{y})] && \text{(3.44a)} \\
&\quad - \boldsymbol{\Sigma}_i^{-1}\mathbb{E}_{q_i}[(\mathbf{y}-\boldsymbol{\mu}_i)\log\bar{q}(\mathbf{y})], && \text{(3.44a)} \\
\frac{\partial^2 V_i(q_i)}{\partial \boldsymbol{\mu}_i^{\top}\partial\boldsymbol{\mu}_i} &= \boldsymbol{\Sigma}_i^{-1}\mathbb{E}_{q_i}[(\mathbf{y}-\boldsymbol{\mu}_i)(\mathbf{y}-\boldsymbol{\mu}_i)^{\top}\phi_i(\mathbf{y})]\boldsymbol{\Sigma}_i^{-1} && \text{(3.44b)} \\
&\quad - \boldsymbol{\Sigma}_i^{-1}\mathbb{E}_{q_i}[(\mathbf{y}-\boldsymbol{\mu}_i)(\mathbf{y}-\boldsymbol{\mu}_i)^{\top}\log\bar{q}(\mathbf{y})]\boldsymbol{\Sigma}_i^{-1} && \text{(3.44b)} \\
&\quad - \boldsymbol{\Sigma}_i^{-1}\mathbb{E}_{q_i}[\phi_i(\mathbf{y})] + \boldsymbol{\Sigma}_i^{-1}\mathbb{E}_{q_i}[\log\bar{q}(\mathbf{y})], && \text{(3.44b)} \\
\frac{\partial V_i(q_i)}{\partial \boldsymbol{\Sigma}_i^{-1}} &= -\frac{1}{2}\mathbb{E}_{q_i}[(\mathbf{y}-\boldsymbol{\mu}_i)(\mathbf{y}-\boldsymbol{\mu}_i)^{\top}\phi_i(\mathbf{y})] && \text{(3.44c)} \\
&\quad + \frac{1}{2}\boldsymbol{\Sigma}_i\mathbb{E}_{q_i}[\boldsymbol{\psi}(\mathbf{y})] + \frac{1}{2}(\alpha+\lambda-\lambda\mathbf{A}_{ii})\boldsymbol{\Sigma}_i && \text{(3.44c)} \\
&\quad + \frac{1}{2}\mathbb{E}_{q_i}[(\mathbf{y}-\boldsymbol{\mu}_i)(\mathbf{y}-\boldsymbol{\mu}_i)^{\top}\log\bar{q}(\mathbf{y})] && \text{(3.44c)} \\
&\quad - \frac{1}{2}\boldsymbol{\Sigma}_i\mathbb{E}_{q_i}[\log\bar{q}(\mathbf{y})]. && \text{(3.44c)}
\end{aligned}
$$

Comparing Eq. 3.44b and Eq. 3.44c,

$$\frac{\partial^2 V_i(q_i)}{\partial \boldsymbol{\mu}_i^\top \partial \boldsymbol{\mu}_i} = (\alpha + \lambda - \lambda A_{ii})\boldsymbol{\Sigma}_i^{-1} - 2\boldsymbol{\Sigma}_i^{-1}\frac{\partial V_i(q_i)}{\partial \boldsymbol{\Sigma}_i^{-1}}\boldsymbol{\Sigma}_i^{-1}. \tag{3.45}$$

Applying Stein's lemma from Eq. 3.20 to Eq. 3.44a, we get,

$$\begin{aligned}\frac{\partial V_i(q_i)}{\partial \boldsymbol{\mu}_i^\top} &= \mathbb{E}_{q_i}[\frac{\partial \phi_i(\mathbf{y})}{\partial \mathbf{y}^\top}] - \mathbb{E}_{q_i}[\frac{\partial \log \bar{q}(\mathbf{y})}{\partial \mathbf{y}^\top}] \\ &= \mathbb{E}_{q_i}[-\mathbf{C}_i^\top \mathbf{V}_i^{-1}(\mathbf{z}_i - \mathbf{C}_i\mathbf{y})] + \mathbb{E}_{q_i}[\bar{\boldsymbol{\Sigma}}^{-1}(\mathbf{y} - \bar{\boldsymbol{\mu}})] \\ &= -\mathbf{C}_i^\top \mathbf{V}_i^{-1}(\mathbf{z}_i - \mathbf{C}_i\boldsymbol{\mu}_i) + \bar{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{\mu}_i - \bar{\boldsymbol{\mu}}).\end{aligned} \tag{3.46}$$

The second derivative is then given by,

$$\frac{\partial^2 V_i(q_i)}{\partial \boldsymbol{\mu}_i^\top \partial \boldsymbol{\mu}_i} = \bar{\boldsymbol{\Sigma}}^{-1} + \mathbf{C}_i^\top \mathbf{V}_i^{-1}\mathbf{C}_i. \tag{3.47}$$

By setting $\frac{\partial V_i(q_i)}{\partial \boldsymbol{\Sigma}_i^{-1}} = 0$ and using Eq. 3.45, we get,

$$\boldsymbol{\Sigma}_i^{-1} = \frac{1}{\alpha + \lambda - \lambda A_{ii}}(\bar{\boldsymbol{\Sigma}}^{-1} + \mathbf{C}_i^T \mathbf{V}_i^{-1}\mathbf{C}_i). \tag{3.48}$$

Setting $\frac{\partial V_i(q_i)}{\partial \boldsymbol{\mu}_i} = 0$ and substituting $\boldsymbol{\Sigma}_i^{-1}$ into Eq. 3.46, the information vector is given by,

$$\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i = \frac{1}{\alpha + \lambda - \lambda \mathbf{A}_{ii}}(\bar{\boldsymbol{\Sigma}}^{-1}\bar{\boldsymbol{\mu}} + \mathbf{C}_i^T \mathbf{V}_i^{-1}\mathbf{z}_i). \tag{3.49}$$

If we set $\alpha = \mathbf{A}_{ii}$ and $\lambda = 1$, it is similar to the optimal parameters derived through SMD formulation as shown in Eq. 3.38.

3.4 Distributed Kalman Filter

This section presents the distributed Kalman filter algorithm which is the main focus of this thesis. By extending the objective function in the previous section to take into account

the robot's motion, the joint probability of both robot and landmark states $q(\mathbf{x},\mathbf{y})$ is considered instead of only the common landmarks $q(\mathbf{y})$, and the minimization function can be rewritten as,

$$\min_{q_{t+1}} V(q_{t+1}) = \min_{q_{t+1}} \mathrm{KL}(q_{t+1}(\mathbf{x}_{t+1},\mathbf{y})||p(\mathbf{x}_{t+1},\mathbf{y}|\mathbf{z}_{0:t+1},\mathbf{u}_{0:t})), \tag{3.50}$$

where $q_{t+1}(\mathbf{x}_{t+1},\mathbf{y})$ is the approximate posterior distribution, and $p_{t+1}(\mathbf{x}_{t+1},\mathbf{y}|\mathbf{z}_{0:t+1},\mathbf{u}_{0:t})$ is the true posterior distribution.

This equation can then be rewritten again similar to the SMD formulation in the previous section using Markov assumptions and Bayes rule.

$$\begin{aligned}
V(q_{t+1}) &= \mathrm{KL}\left(q_{t+1}(\mathbf{x}_{t+1},\mathbf{y})||p_{t+1}(\mathbf{x}_{t+1},\mathbf{y}|\mathbf{z}_{0:t+1},\mathbf{u}_{0:t})\right) \\
&= \mathbb{E}_{q_{t+1}}\left[\log q_{t+1}(\mathbf{x}_{t+1},\mathbf{y})\right] - \mathbb{E}_{q_{t+1}}\left[\log p_{t+1}(\mathbf{x}_{t+1},\mathbf{y}|\mathbf{z}_{0:t+1},\mathbf{u}_{0:t})\right] \\
&= \mathbb{E}_{q_{t+1}}\left[\log q_{t+1}(\mathbf{x}_{t+1}|\mathbf{y}) + \log q_{t+1}(\mathbf{y})\right] \\
&\quad - \mathbb{E}_{q_{t+1}}\left[\log p_{t+1}(\mathbf{z}_{t+1}|\mathbf{x}_{t+1},\mathbf{y},\mathbf{z}_{0:t},\mathbf{u}_{0:t}) + \log p_{t+1}(\mathbf{x}_{t+1},\mathbf{y}|\mathbf{z}_{0:t},\mathbf{u}_{0:t})\right] \\
&= \mathbb{E}_{q_{t+1}}\left[\log q_{t+1}(\mathbf{x}_{t+1}|\mathbf{y})\right] + \mathbb{E}_{q_{t+1}}\left[\log q_{t+1}(\mathbf{y})\right] \\
&\quad - \mathbb{E}_{q_{t+1}}\left[\log p_{t+1}(\mathbf{z}_{t+1}|\mathbf{x}_{t+1},\mathbf{y})\right] - \mathbb{E}_{q_{t+1}}\left[-\log p_{t+1}(\mathbf{x}_{t+1}|\mathbf{y},\mathbf{z}_{0:t},\mathbf{u}_{0:t})\right] \\
&\quad - \mathbb{E}_{q_{t+1}}\left[\log p_{t+1}(\mathbf{y})\right] \\
&= \mathbb{E}_{q_{t+1}}\left[-\log p_{t+1}(\mathbf{z}_{t+1}|\mathbf{x}_{t+1},\mathbf{y})\right] \\
&\quad + \mathbb{E}_{q_{t+1}}\left[\log q_{t+1}(\mathbf{x}_{t+1}|\mathbf{y}) - \log p_{t+1}(\mathbf{x}_{t+1}|\mathbf{y},\mathbf{z}_{0:t},\mathbf{u}_{0:t})\right] \\
&\quad + \mathbb{E}_{q_{t+1}}\left[\log q_{t+1}(\mathbf{y}) - \log p_{t+1}(\mathbf{y})\right].
\end{aligned} \tag{3.51}$$

This problem can be decomposed like in Eq. 3.29 and adding the consensus constraint only to the marginal distribution of the landmarks, it leads to the following minimization problem

that is solved by each individual robot,

$$\begin{aligned} q_{i,t+1} \in \underset{q_{i,t+1}}{\operatorname{argmin}}\ & \mathbb{E}_{q_{i,t+1}}[-\log p_{i,t+1}(\mathbf{z}_{i,t+1}|\mathbf{x}_{i,t+1},\mathbf{y})] \\ & + \mathbb{E}_{q_{i,t+1}}\left[\log q_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y}) - \log p_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y},\mathbf{z}_{i,0:t},\mathbf{u}_{i,0:t})\right] \\ & + \sum_{j\in\mathbb{N}_{t+1}} \mathbf{A}_{ij,t+1}\mathbb{E}_{q_{i,t+1}}\left[\log q_{i,t+1}(\mathbf{y}) - \log p_{j,t+1}(\mathbf{y})\right]. \end{aligned} \tag{3.52}$$

This minimization problem has a closed form solution given by,

$$q_{i,t+1}(\mathbf{x}_{i,t+1},\mathbf{y}) \propto p_{i,t+1}(\mathbf{z}_{i,t+1}|\mathbf{x}_{i,t+1},\mathbf{y})\ p_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y},\mathbf{z}_{i,0:t},\mathbf{u}_{i,0:t}) \prod_{j\in\mathbb{N}_{i,t+1}} p_{j,t+1}^{\mathbf{A}_{ij,t+1}}(\mathbf{y}). \tag{3.53}$$

Proof. Relaxing the equality constraint $\int q_{i,t+1} = 1$ with Lagrangian multiplier, leads to,

$$\begin{aligned} L(q_{i,t+1},\lambda) = & \mathbb{E}_{q_{i,t+1}}[-\log p_{i,t+1}(\mathbf{z}_{i,t+1}|\mathbf{x}_{i,t+1},\mathbf{y})] \\ & + \mathbb{E}_{q_{i,t+1}}\left[\log q_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y}) - \log p_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y},\mathbf{z}_{i,0:t},\mathbf{u}_{i,0:t})\right] \\ & + \sum_{j\in\mathbb{N}_{t+1}} \mathbf{A}_{ij,t+1}\mathbb{E}_{q_{i,t+1}}\left[\log q_{i,t+1}(\mathbf{y}) - \log p_{j,t+1}(\mathbf{y})\right] + \lambda(\mathbb{E}_{\mathbb{1}}[q_{i,t+1}] - 1). \end{aligned} \tag{3.54}$$

The derivative of the right hand side with respect to $q_{i,t+1}$ is given by,

$$\begin{aligned} \frac{\partial L}{\partial q_{i,t+1}} = & -\log p_{i,t+1}(\mathbf{z}_{i,t+1}|\mathbf{x}_{i,t+1},\mathbf{y}) \\ & + \left(\frac{\partial}{\partial q_{i,t+1}}\mathbb{E}_{q_{i,t+1}}[\log q_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y})] - \log p_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y},\mathbf{z}_{i,0:t},\mathbf{u}_{i,0:t})\right) \\ & + \left(\frac{\partial}{\partial q_{i,t+1}}\mathbb{E}_{q_{i,t+1}}[\log q_{i,t+1}(\mathbf{y})] - \sum_{j\in\mathbb{N}_{i,t+1}} \mathbf{A}_{ij,t+1}\log p_{j,t+1}(\mathbf{y})\right) + \lambda. \end{aligned} \tag{3.55}$$

Setting this equal to 0,

$$\begin{aligned}
\frac{\partial}{\partial q_{i,t+1}}\mathbb{E}_{q_{i,t+1}}[\log q_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y})+\log q_{i,t+1}(\mathbf{y})] &= \log p_{i,t+1}(\mathbf{z}_{i,t+1}|\mathbf{x}_{i,t+1},\mathbf{y}) \\
&\quad + \log p_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y},\mathbf{z}_{i,0:t},\mathbf{u}_{i,0:t}) \\
&\quad + \sum_{j\in\mathbb{N}_{i,t+1}} \mathbf{A}_{ij,t+1}\log p_{j,t+1}(\mathbf{y}) - \lambda, \\
\frac{\partial}{\partial q_{i,t+1}}\mathbb{E}_{q_{i,t+1}}[\log q_{i,t+1}(\mathbf{x}_{i,t+1},\mathbf{y})] &= \log p_{i,t+1}(\mathbf{z}_{i,t+1}|\mathbf{x}_{i,t+1},\mathbf{y}) \\
&\quad + \log p_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y},\mathbf{z}_{i,0:t},\mathbf{u}_{i,0:t}) \\
&\quad + \sum_{j\in\mathbb{N}_{i,t+1}} \mathbf{A}_{ij,t+1}\log p_{j,t+1}(\mathbf{y}) - \lambda, \\
1+\log q_{i,t+1}(\mathbf{x}_{i,t+1},\mathbf{y}) &= \log p_{i,t+1}(\mathbf{z}_{i,t+1}|\mathbf{x}_{i,t+1},\mathbf{y}) \\
&\quad + \log p_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y},\mathbf{z}_{i,0:t},\mathbf{u}_{i,0:t}) \\
&\quad + \sum_{j\in\mathbb{N}_{i,t+1}} \mathbf{A}_{ij,t+1}\log p_{j,t+1}(\mathbf{y}) - \lambda, \\
q_{i,t+1}(\mathbf{x}_{i,t+1},\mathbf{y}) &= \exp(-1-\lambda+\log p_{i,t+1}(\mathbf{z}_{i,t+1}|\mathbf{x}_{i,t+1},\mathbf{y}) \\
&\quad + \log p_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y},\mathbf{z}_{i,0:t},\mathbf{u}_{i,0:t})) \\
&\quad \prod_{j\in\mathbb{N}_{i,t+1}} p_{j,t+1}^{\mathbf{A}_{ij,t+1}}(\mathbf{y}).
\end{aligned} \tag{3.56}$$

The constraint $\int q_{i,t+1} = 1$ can be used to solve for λ.

$$\begin{aligned}
&\exp(-1-\lambda)\times \\
&\underbrace{\int \exp(\log p_{i,t+1}(\mathbf{z}_{i,t+1}|\mathbf{x}_{i,t+1},\mathbf{y})+\log p_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y},\mathbf{z}_{i,0:t},\mathbf{u}_{i,0:t}))\prod_{j\in\mathbb{N}_{i,t+1}} p_{j,t+1}^{\mathbf{A}_{ij,t+1}}(\mathbf{y})}_{\Gamma_{i,t+1}} = 1, \\
&\exp(-1-\lambda) = \frac{1}{\Gamma_{i,t+1}}.
\end{aligned} \tag{3.57}$$

Substituting Eq. 3.57 in Eq. 3.56 leads to,

$$q_{i,t+1}(\mathbf{x}_{i,t+1},\mathbf{y}) = \frac{1}{\Gamma_{i,t+1}}\; p_{i,t+1}(\mathbf{z}_{i,t+1}|\mathbf{x}_{i,t+1},\mathbf{y})\; p_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y},\mathbf{z}_{i,0:t},\mathbf{u}_{i,0:t}) \prod_{j\in\mathbb{N}_{i,t+1}} p_{j,t+1}^{\mathbf{A}_{ij,t+1}}(\mathbf{y}). \tag{3.58}$$

□

The prior distribution $p_{t+1}(\mathbf{y})$ at time $t+1$ is same as the posterior distribution $q_t(\mathbf{y})$ at time t, giving the following closed-form solution,

$$q_{i,t+1}(\mathbf{x}_{i,t+1},\mathbf{y}) \propto \overbrace{\underbrace{p_{i,t+1}(\mathbf{z}_{i,t+1}|\mathbf{x}_{i,t+1},\mathbf{y})\; p_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y},\mathbf{z}_{i,0:t},\mathbf{u}_{i,0:t})}_{\text{Predict}} \underbrace{\prod_{j\in\mathbb{N}_{i,t}} q_{j,t}^{\mathbf{A}_{ij,t}}(\mathbf{y})}_{\text{Average}}}^{\text{Update}}, \tag{3.59}$$

which is similar to the Bayes filter shown in Eq. 1.2 but with an additional averaging component.

The distributed Kalman filter algorithm is presented as a recursive estimation algorithm with three steps: Average, Predict and Update. Consider the joint distribution of the robot and landmark states at time t to be Gaussian distributed and is given by,

$$\begin{bmatrix}\mathbf{y}\\ \mathbf{x}_{i,t}\end{bmatrix} = \mathcal{G}\Big(\begin{bmatrix}\boldsymbol{\omega}_{i,t}\\ \boldsymbol{\nu}_{i,t}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Omega}_{i,t} & \mathbf{H}_{i,t}\\ \mathbf{H}_{i,t}^T & \mathbf{N}_{i,t}\end{bmatrix}\Big), \tag{3.60}$$

where $\mathcal{G}(\boldsymbol{\omega},\boldsymbol{\Omega})$ denote a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ parameterized with information mean $\boldsymbol{\omega} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ and information matrix $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$.

The marginal and conditional distributions from Eq. 3.60 are given by,

$$\begin{aligned}\mathbf{y} &\sim \mathcal{G}(\boldsymbol{\omega}_{i,t} - \mathbf{H}_{i,t}\mathbf{N}_{i,t}^{-1}\boldsymbol{\nu}_{i,t},\; \boldsymbol{\Omega}_{i,t} - \mathbf{H}_{i,t}\mathbf{N}_{i,t}^{-1}\mathbf{H}_{i,t}^T),\\ \mathbf{x}_{i,t}|\mathbf{y} &\sim \mathcal{G}(\boldsymbol{\nu}_{i,t} - \mathbf{H}_{i,t}^T\mathbf{y}_t,\; \mathbf{N}_{i,t}).\end{aligned} \tag{3.61}$$

Average: Define $\bar{q}_{i,t}(\mathbf{y}) = \prod_{j \in \mathbb{N}_{i,t}} q_{j,t}^{\mathbf{A}_{ij,t}}(\mathbf{y})$ with information mean $\bar{\boldsymbol{\nu}}_{i,t}$ and information matrix $\bar{\boldsymbol{\Omega}}_{i,t}$. Using Eq. 3.36 and Eq. 3.61, we get,

$$\begin{aligned} \bar{\boldsymbol{\omega}}_{i,t} &= \sum_{j \in \mathbb{N}_{i,t}} \mathbf{A}_{ij,t}(\boldsymbol{\omega}_{j,t} - \mathbf{H}_{j,t}\mathbf{N}_{j,t}^{-1}\boldsymbol{\nu}_{j,t}), \\ \bar{\boldsymbol{\Omega}}_{i,t} &= \sum_{j \in \mathbb{N}_{i,t}} \mathbf{A}_{ij,t}(\boldsymbol{\Omega}_{j,t} - \mathbf{H}_{j,t}\mathbf{N}_{j,t}^{-1}\mathbf{H}_{j,t}^{T}). \end{aligned} \tag{3.62}$$

Predict: Define $q_{i,t+1}^{+}(\mathbf{x}_{i,t+1}, \mathbf{y}) = p_{i,t+1}(\mathbf{x}_{i,t+1}|\mathbf{y}, \mathbf{z}_{0:t}, \mathbf{u}_{0:t})\ \bar{q}_{i,t}(\mathbf{y})$. Using Eq. 3.61 and the motion model in Eq. 3.1, the joint information mean and information matrix is given by,

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\Omega}_{i,t+1}^{+} & \mathbf{H}_{i,t+1}^{+} \\ \mathbf{H}_{i,t+1}^{+^T} & \mathbf{N}_{i,t+1}^{+} \end{bmatrix} &= \left(\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_i \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{\Omega}}_{i,t} + \mathbf{H}_{i,t}\mathbf{N}_{i,t}^{-1}\mathbf{H}_{i,t}^{T} & \mathbf{H}_{i,t} \\ \mathbf{H}_{i,t}^{T} & \mathbf{N}_{i,t} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_i \end{bmatrix}^{T} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_i \end{bmatrix} \right)^{-1}, \\ \begin{bmatrix} \boldsymbol{\omega}_{i,t+1}^{+} \\ \boldsymbol{\nu}_{i,t+1}^{+} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{\Omega}_{i,t+1}^{+} & \mathbf{H}_{i,t+1}^{+} \\ \mathbf{H}_{i,t+1}^{+^T} & \mathbf{N}_{i,t+1}^{+} \end{bmatrix} \left(\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_i \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{\Omega}}_{i,t} + \mathbf{H}_{i,t}\mathbf{N}_{i,t}^{-1}\mathbf{H}_{i,t}^{T} & \mathbf{H}_{i,t} \\ \mathbf{H}_{i,t}^{T} & \mathbf{N}_{i,t} \end{bmatrix}^{-1} \begin{bmatrix} \bar{\boldsymbol{\omega}}_{i,t} + \mathbf{H}_{i,t}\mathbf{N}_{i,t}^{-1}\boldsymbol{\nu}_{i,t} \\ \boldsymbol{\nu}_{i,t} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_i\mathbf{u}_{i,t} \end{bmatrix} \right). \end{aligned} \tag{3.63}$$

Update: Define $q_{i,t+1}(\mathbf{x}_{i,t+1}, \mathbf{y}) = p_{i,t+1}(\mathbf{z}_{i,t+1}|\mathbf{x}_{i,t+1}\mathbf{y})\ q_{i,t+1}^{+}(\mathbf{x}_{i,t+1}, \mathbf{y})$. The joint information mean and information matrix based on the observation model in Eq. 3.2 is given by,

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\Omega}_{i,t+1} & \mathbf{H}_{i,t+1} \\ \mathbf{H}_{i,t+1}^{T} & \mathbf{N}_{i,t+1} \end{bmatrix}, &= \begin{bmatrix} \boldsymbol{\Omega}_{i,t+1}^{+} & \mathbf{H}_{i,t+1}^{+} \\ \mathbf{H}_{i,t+1}^{+^T} & \mathbf{N}_{i,t+1}^{+} \end{bmatrix} + \begin{bmatrix} \mathbf{C}_{i,y}^{T} \\ \mathbf{C}_{i,x}^{T} \end{bmatrix} \mathbf{V}_i^{-1} \begin{bmatrix} \mathbf{C}_{i,y} & \mathbf{C}_{i,x} \end{bmatrix}, \\ \begin{bmatrix} \boldsymbol{\omega}_{i,t+1} \\ \boldsymbol{\nu}_{i,t+1} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{\omega}_{i,t+1}^{+} \\ \boldsymbol{\nu}_{i,t+1}^{+} \end{bmatrix} + \begin{bmatrix} \mathbf{C}_{i,y}^{T} \\ \mathbf{C}_{i,x}^{T} \end{bmatrix} \mathbf{V}_i^{-1}\mathbf{z}_{i,t+1}. \end{aligned} \tag{3.64}$$

The distributed Kalman filter is described in Alg. 1.

Algorithm 1. Distributed Kalman Filter

Require: Prior pdf parameters $(\boldsymbol{\omega}_{i,t}, \boldsymbol{\nu}_{i,t}, \boldsymbol{\Omega}_{i,t}, \mathbf{H}_{i,t}, \mathbf{N}_{i,t})$, messages $(\boldsymbol{\omega}_{j,t}, \boldsymbol{\nu}_{j,t}, \boldsymbol{\Omega}_{j,t}, \mathbf{H}_{j,t}, \mathbf{N}_{j,t})$, control input $\mathbf{u}_{i,t}$, and measurement $\mathbf{z}_{i,t+1}$.

For each robot i,
Average:

$$\bar{\boldsymbol{\omega}}_{i,t} = \sum_{j \in \mathbb{N}_{i,t}} \mathbf{A}_{ij,t}(\boldsymbol{\omega}_{j,t} - \mathbf{H}_{j,t}\mathbf{N}_{j,t}^{-1}\boldsymbol{\nu}_{j,t})$$
$$\bar{\boldsymbol{\Omega}}_{i,t} = \sum_{j \in \mathbb{N}_{i,t}} \mathbf{A}_{ij,t}(\boldsymbol{\Omega}_{j,t} - \mathbf{H}_{j,t}\mathbf{N}_{j,t}^{-1}\mathbf{H}_{j,t}^{T})$$

Predict:

$$\begin{bmatrix} \boldsymbol{\Omega}_{i,t+1}^{+} & \mathbf{H}_{i,t+1}^{+} \\ \mathbf{H}_{i,t+1}^{+^T} & \mathbf{N}_{i,t+1}^{+} \end{bmatrix} = \left(\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_i \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{\Omega}}_{i,t} + \mathbf{H}_{i,t}\mathbf{N}_{i,t}^{-1}\mathbf{H}_{i,t}^{T} & \mathbf{H}_{i,t} \\ \mathbf{H}_{i,t}^{T} & \mathbf{N}_{i,t} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_i \end{bmatrix}^{T} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_i \end{bmatrix} \right)^{-1}$$
$$\begin{bmatrix} \boldsymbol{\omega}_{i,t+1}^{+} \\ \boldsymbol{\nu}_{i,t+1}^{+} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Omega}_{i,t+1}^{+} & \mathbf{H}_{i,t+1}^{+} \\ \mathbf{H}_{i,t+1}^{+^T} & \mathbf{N}_{i,t+1}^{+} \end{bmatrix} \left(\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_i \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{\Omega}}_{i,t} + \mathbf{H}_{i,t}\mathbf{N}_{i,t}^{-1}\mathbf{H}_{i,t}^{T} & \mathbf{H}_{i,t} \\ \mathbf{H}_{i,t}^{T} & \mathbf{N}_{i,t} \end{bmatrix}^{-1} \begin{bmatrix} \bar{\boldsymbol{\omega}}_{i,t} + \mathbf{H}_{i,t}\mathbf{N}_{i,t}^{-1}\boldsymbol{\nu}_{i,t} \\ \boldsymbol{\nu}_{i,t} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_i\mathbf{u}_{i,t} \end{bmatrix} \right)$$

Update:

$$\begin{bmatrix} \boldsymbol{\Omega}_{i,t+1} & \mathbf{H}_{i,t+1} \\ \mathbf{H}_{i,t+1}^{T} & \mathbf{N}_{i,t+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Omega}_{i,t+1}^{+} & \mathbf{H}_{i,t+1}^{+} \\ \mathbf{H}_{i,t+1}^{+^T} & \mathbf{N}_{i,t+1}^{+} \end{bmatrix} + \begin{bmatrix} \mathbf{C}_{i,y}^{T} \\ \mathbf{C}_{i,x}^{T} \end{bmatrix} \mathbf{V}_i^{-1} \begin{bmatrix} \mathbf{C}_{i,y} & \mathbf{C}_{i,x} \end{bmatrix}$$
$$\begin{bmatrix} \boldsymbol{\omega}_{i,t+1} \\ \boldsymbol{\nu}_{i,t+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega}_{i,t+1}^{+} \\ \boldsymbol{\nu}_{i,t+1}^{+} \end{bmatrix} + \begin{bmatrix} \mathbf{C}_{i,y}^{T} \\ \mathbf{C}_{i,x}^{T} \end{bmatrix} \mathbf{V}_i^{-1}\mathbf{z}_{i,t+1}$$

All the Chapters, in part, are currently being prepared for submission for publication of the material. Cao, Hanwen; Shreedharan, Sriram; Kansal, Shrey; Kumar, Shubham; Nukala, Kishore; Atanasov, Nikolay. The thesis author was one of the researchers/authors of this paper.

Section 3.3 in Chapter 3 is co-authored with Hanwen Cao. The thesis author was the primary author of this chapter.

Chapter 4

Experiment and Results

4.1 Application to Multi-Robot SLAM

We apply the distributed Kalman filter algorithm proposed in the previous Chapter to a Multi-Robot Visual SLAM problem. For practical purposes, the robot's motion model and the camera observation model is not linear. Therefore, we linearize the non-linear models and follow the update step similar to the extended Kalman filter [5].

Assumptions:

1. The observed images are undistorted and rectified, and the camera intrinsic parameters, that is, the calibration matrix is known which is given by,

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.1}$$

   where (f_x, f_y) is the focal length of the camera in pixels, (c_x, c_y) is the optical center of the camera in pixels, and s is the skew.

2. The data association across robots is known with the common landmarks sharing the same global ids.

3. The initial poses of all the robots with respect to a common world frame is known.

4. The communication graph is fully-connected; that is, each robot can communicate with all the other robots at every time step.

To simultaneously create a sparse map of the environment and accurately localize the robot, features can be extracted from the images and tracked through time. With only the control input, and a single observation of a feature, the estimated state of the robot and the landmark will be noisy, and over time, the noise accumulates and leads to a large error. To correct this error, we can use the observations from the environment tracked through multiple frames and combine all the information to accurately estimate the states using filtering methods.

The 3D coordinates of the landmarks are initialized using triangulation when they are first observed. The depth of a landmark is computed as follows,

$$\begin{aligned} disp &= u_l - u_r \\ d &= \frac{f_x\, b}{disp} \end{aligned} \tag{4.2}$$

where $disp$ is the disparity, u_l and u_r are the features in the x-axis from the left and right images, d is the depth of the landmark, and b is the baseline of the stereo camera.

Once we have the depth, we can compute the 3D coordinates of the landmark in the camera frame, which can then be transformed to the world frame using the estimated pose of the robot at that time (assuming Identity transformation between the camera and the robot frame).

$$\begin{aligned} \begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{bmatrix} &= d * K^{-1} \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} \\ \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} &= {}_wT_{cam} \begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \\ 1 \end{bmatrix} \end{aligned} \tag{4.3}$$

where $(X_{cam}, Y_{cam}, Z_{cam})$ and (X_w, Y_w, Z_w) are the coordinates of the landmark in camera frame and world frame respectively, and ${}_wT_{cam}$ is the transformation from camera to world frame.

4.1.1 Feature Tracking

We use the Oriented FAST and rotated BRIEF [39] feature detection algorithm to obtain unique features from the images. This method is considered as it is faster than other state-of-the-art feature detection algorithms like SIFT (Scale-Invariant Feature Transform) [26] and SURF (Speeded-Up Robust Features) [7] while matching the performance of these algorithms.

Once we obtain the features, we utilize the feature matching algorithm from OpenCV with outlier rejection to accurately track the observed ORB features over time.

ORB Feature Detection Algorithm

ORB is built on top of FAST (Features from Accelerated Segment Test) [38] keypoint detector and BRIEF (Binary Robust Independent Elementary Features) [11] feature descriptor. FAST algorithm works by taking a pixel p and comparing the brightness of its 16 surrounding pixels within a small circle. If 8 pixels are brighter or darker than p, it is considered a keypoint. However, these keypoints do not have any orientation assigned and are therefore not rotation invariant. To solve this issue, the ORB feature detection algorithm takes a patch around the pixel and calculates its orientation using its moments, introducing rotation invariance. Additionally, the ORB features are detected from multi-scale image pyramids, meaning different scales of the same image are considered for keypoint detection, thus making the algorithm partially scale invariant.

The next step is to compute the keypoint descriptors using the BRIEF algorithm. BRIEF compares a random pair of points from the neighborhood around the keypoint. The first point is sampled from a Gaussian distribution centered around the mean point with some standard deviation σ, while the second point is sampled from a Gaussian distribution centered around the first point with standard deviation $\frac{\sigma}{2}$. If the first pixel is brighter than the second pixel, then a value of 1 is assigned, else, it is set to 0. This process is repeated 128 times for each keypoint to get a 128-bit descriptor. The issue with the BRIEF algorithm is that it is not rotation invariant, so ORB uses rBRIEF (Rotation-aware BRIEF), which steers the binary test points based on the

patch orientation θ and computes the descriptor from the new set of points. Let (x_i, y_i) denote the test points and say there are n test points, then,

$$\mathbf{S} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{bmatrix}$$
$$\mathbf{S}_\theta = \mathbf{R}_\theta \mathbf{S} \tag{4.4}$$

Feature matching

First, the ORB features are computed and matched between the stereo pairs of two consecutive frames. Then, the matched features in the left images of the two frames are matched again to ensure consistency. A brute force matcher is used to match the keypoints with similar descriptors. Essentially, every descriptor in one frame is compared with all the descriptors in the other frame and the descriptors that are almost the same (nearest neighbor) are matched. Additionally, the fundamental matrix is computed between the two frames using the detected set of matches, and RANSAC is utilized to remove outliers, thus retaining only the good matches.

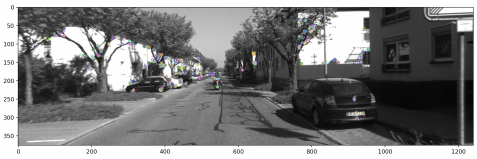
An example feature matching between the first and second frame of sequence 00 from the KITTI dataset using ORB feature detection and matching from OpenCV [33] is shown in Fig. 4.1

4.2 Evaluation

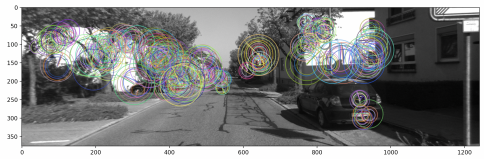
The algorithm has been tested on a simulation data and real-time data from KITTI. The results on both datasets show that the distributed Kalman filter has a smaller estimation error than each robot individually running the Kalman filter.

4.2.1 Metrics

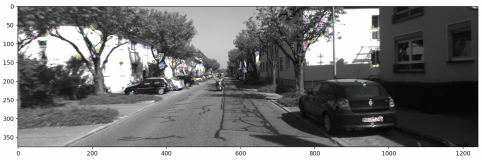
We use two different metrics to evaluate the trajectory estimates: the absolute translation error t_{abs} given in meters and the relative translation error t_{rel} given in %. The errors are calculated



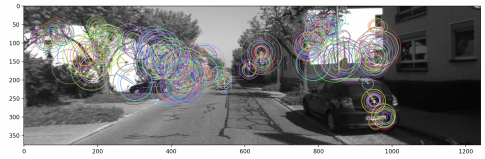
(a) ORB Keypoints - Image 1



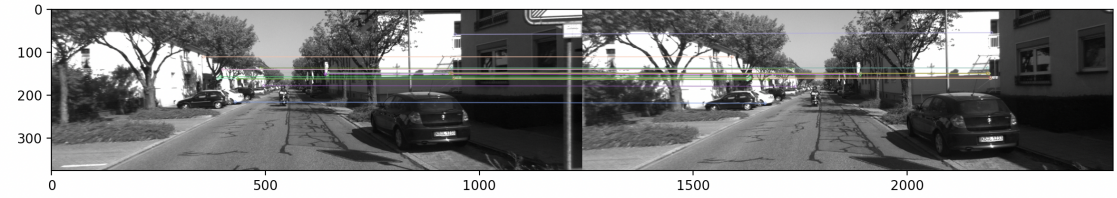
(b) Keypoints Size



(c) ORB Keypoints - Image 2



(d) Keypoints Size



(e) Feature Matches between Image 1 and 2

Figure 4.1. ORB Feature Matching

using the tool rpg trajectory evaluation [42]. The absolute translation error is determined directly by computing the root-mean-squared error (rmse) between the estimated state and the groundtruth at each time step. To compute the relative translation error, the trajectory is divided into equal lengths, and the absolute translation error is calculated for each sub-trajectory. This error is then expressed as a percentage of the sub-trajectory length, and the final relative translation error is calculated by averaging the individual sub-trajectory errors.

4.2.2 Simulation Data

In the simulation environment, we consider 3 robots and 600 common landmarks. We generated an 8-shape trajectory for each robot consisting of 400 time steps. The control inputs and measurements with some random Gaussian noise added are available at each time step. The control inputs are pose transformations for the robots from time t to $t+1$, and the measurements are rectified stereo observations with data association obtained by each robot. The robots ground truth trajectories and landmarks in 2D are shown in Fig. 4.2.

To analyze the performance of our algorithm, we first run the Kalman filter algorithm individually for each robot to obtain a baseline error, and the results are shown in Fig. 4.3. We do not consider any communication between the robots in this case, and each robot estimates its own state and map. Next, we implemented the proposed distributed Kalman filter algorithm on the dataset, and the results are shown in Fig. 4.4. In this case, each robot can communicate with all the other robots and shares the pdf parameters (information mean and information vector) of the common landmarks. The comparison of the overall translation error and error in each axis (x, y, z) across time is shown in Fig. 4.5 and 4.6, where the orange line indicates the error using the Kalman filter and the blue line indicates the error using the distributed Kalman filter algorithm. Table 4.1 compares the algorithms based on the metrics described in the previous section. It is clear from the figures and the table that our algorithm outperforms the individual Kalman filter.

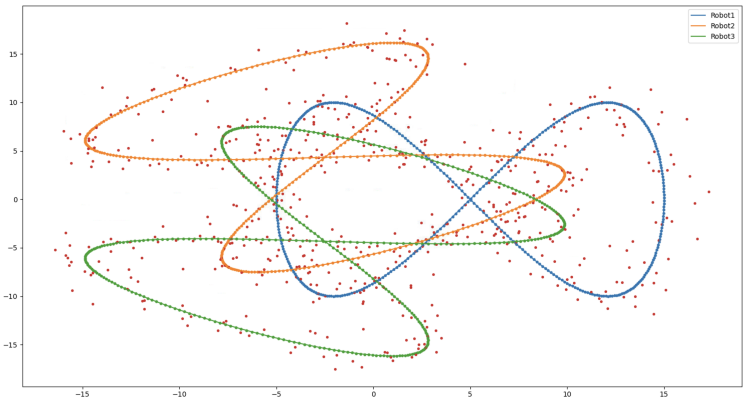
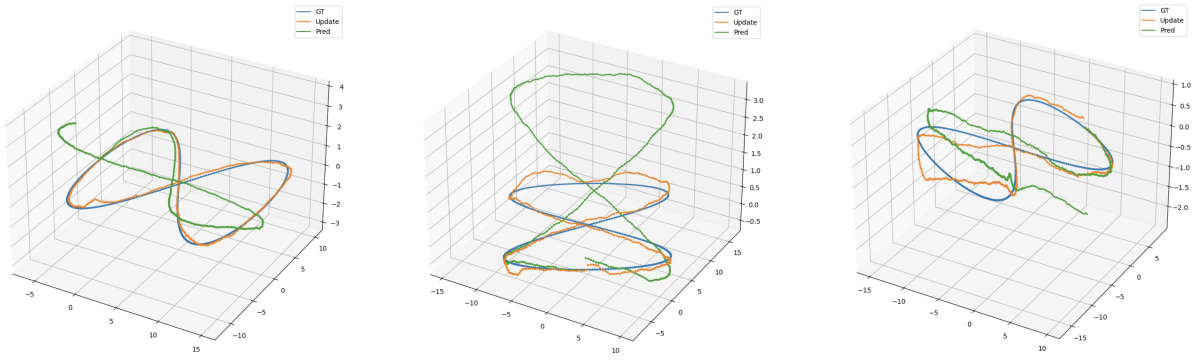


Figure 4.2. Simulation Environment of 3 Robots and Landmarks in 2D

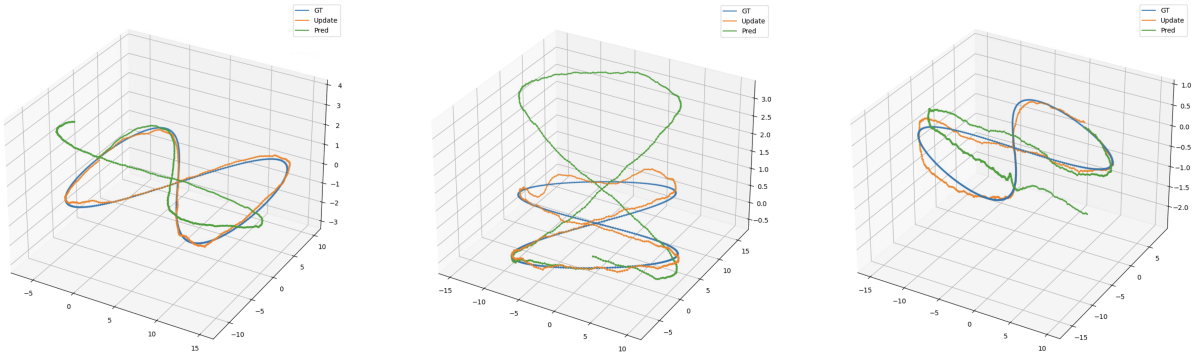
Table 4.1. Error Comparison in Simulation Data

	Distributed Kalman Filter		Kalman Filter	
Robot	t_{abs} (m)	t_{rel} (%)	t_{abs} (m)	t_{rel} (%)
1	**0.2289**	**4.6610**	0.3531	5.3456
2	**0.1919**	**5.4560**	0.2922	6.8593
3	**0.3732**	**5.8495**	0.4595	6.2197
Mean	**0.2646**	**5.3222**	0.3683	6.1415



(a) Robot-1 **(b)** Robot-2 **(c)** Robot-3

Figure 4.3. Kalman Filter in Simulation Data



(a) Robot-1 **(b)** Robot-2 **(c)** Robot-3

Figure 4.4. Distributed Kalman Filter in Simulation Data

4.2.3 KITTI Data

The algorithm has also been tested with Semantic KITTI dataset [1], [18] sequence 00, where the instance segmentation and labels of around 176 objects in the images have been provided. The centroid of the bounding box of these objects are considered as landmarks in the environment, and since the instance labels are given, the data association is known across all frames. However, the number of objects in the Semantic KITTI dataset is very few, so to improve the localization accuracy, ORB features are extracted from the images, and features that appear in continuous frames are tracked. These ORB features have only short-term data association and are local to each robot. Once the feature track is lost, the feature will be initialized as a new one when observed again. On the other hand, the semantic landmarks have long-term data

Figure 4.5. Translation Error Comparison across Time in Simulation Data

association and share the same feature id across robots. Therefore, only the semantic feature observations are shared across all robots.

The sequence 00 has a total of 4540 frames (stereo images) which is split into 3: robot1 is from frame 0 to 2500, robot2 is from frame 1300 to 3800, and robot3 is from frame 2000 to 4540. The ground truth trajectories of the robots and the landmarks are shown in 2D in Fig. 4.7. The initial pose predictions (control inputs) for the robot states are obtained from libviso [19], [24], which uses the stereo images to perform feature detection and matching, and computes the relative pose between the camera frames. This pose estimation is, however, noisy, and we require a filter for an accurate state estimate.

Only Semantic Landmarks

In this section, we will consider only the semantic landmarks for localization and mapping. Similar to the simulation dataset, we run the Kalman filter algorithm individually for each robot to obtain a baseline error, and the results are shown in Fig. 4.8. Again, we do not consider any communication between the robots, and each robot estimates its own state and map. Next, the

Figure 4.6. xyz Drift Comparison across Time in Simulation Data

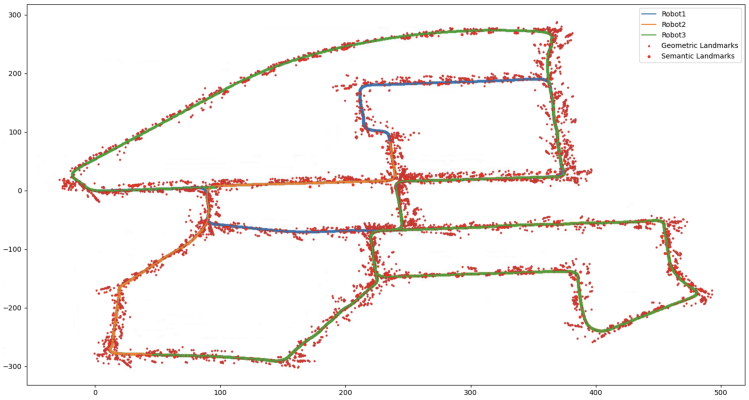
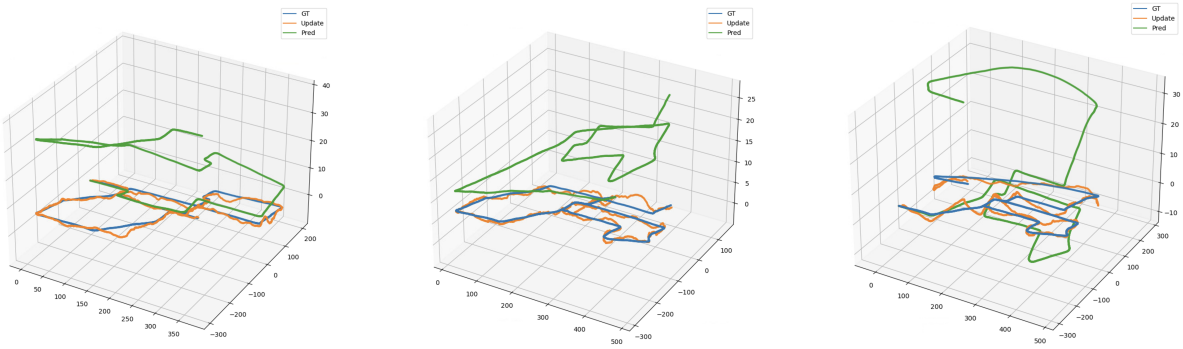


Figure 4.7. KITTI Sequence 00 split for 3 Robots with Semantic Landmarks and Geometric Landmarks in 2D

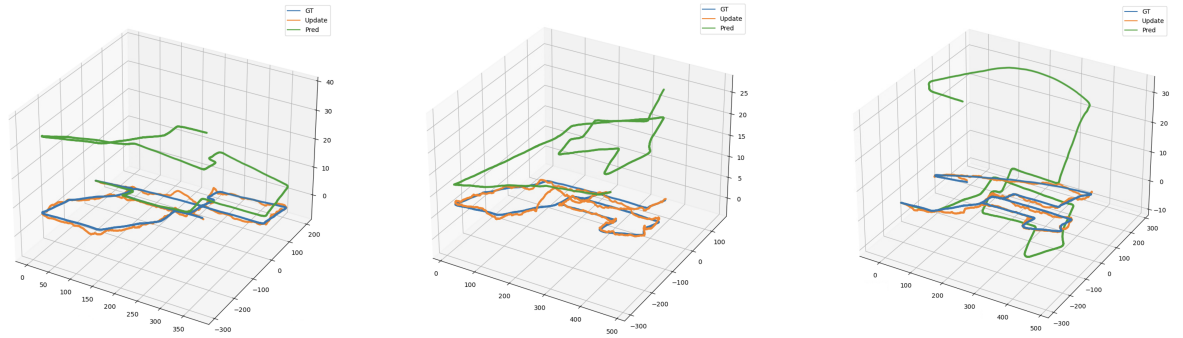
proposed distributed Kalman filter algorithm is implemented on the dataset, and the results are shown in Fig. 4.9.

The comparison of the overall translation error and error in each axis (x, y, z) across time is shown in Fig. 4.10 and 4.11 for the Kalman filter and the distributed Kalman filter algorithm. Table 4.2 compares the absolute and relative translation errors between the algorithms. The



(a) Robot-1 **(b)** Robot-2 **(c)** Robot-3

Figure 4.8. Kalman Filter in KITTI Data with Semantic Landmarks



(a) Robot-1 **(b)** Robot-2 **(c)** Robot-3

Figure 4.9. Distributed Kalman Filter in KITTI Data with Semantic Landmarks

distributed Kalman filter again outperforms the individual Kalman filter, as is evident from the figures and table.

Table 4.2. Error Comparison in KITTI Data with Semantic Landmarks

	Distributed Kalman Filter		Kalman Filter	
Robot	t_{abs} (m)	t_{rel} (%)	t_{abs} (m)	t_{rel} (%)
1	**1.7012**	**1.9462**	3.0663	4.7336
2	**1.7499**	**2.0425**	2.1968	3.9980
3	**2.3455**	**1.8409**	2.4369	3.9247
Mean	**1.9322**	**1.9432**	2.5667	4.2188

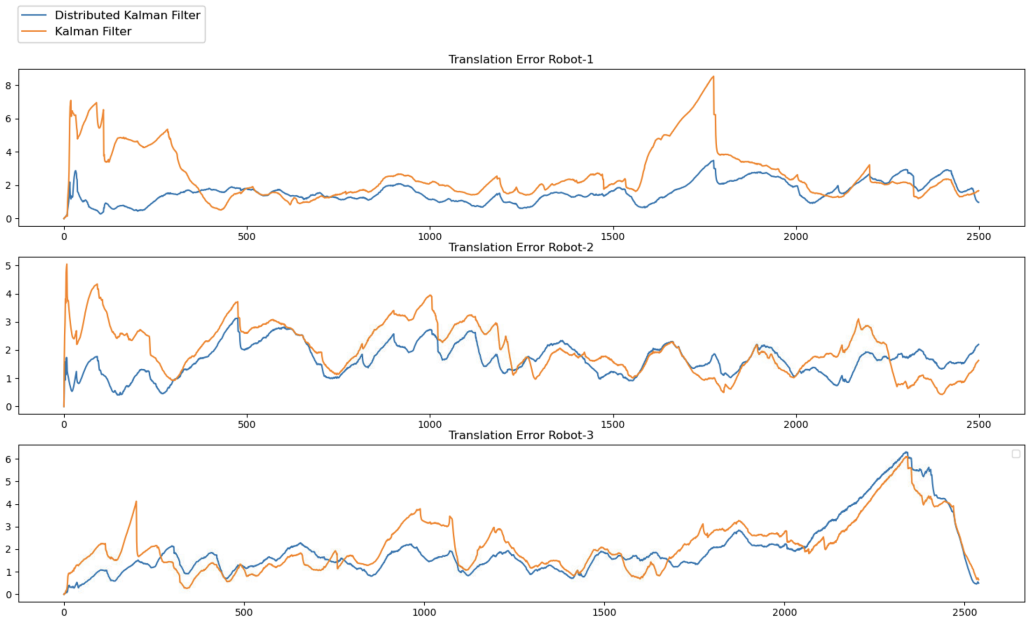


Figure 4.10. Translation Error Comparison across Time in KITTI Data with Semantic Landmarks

Figure 4.11. xyz Drift Comparison across Time in KITTI Data with Semantic Landmarks

Semantic Landmarks and Geometric Landmarks

In this section, we will consider both the semantic landmarks and geometric landmarks and run the Kalman filter and the distributed Kalman filter algorithm on the dataset.

The comparison of the overall translation error and error in each axis (x, y, z) across time is shown in Fig. 4.12 and 4.13, and the Table 4.3 compares the absolute and relative translation errors between the algorithms.

Figure 4.12. Translation Error Comparison across Time in KITTI Data with Semantic Landmarks and Geometric Landmarks

Table 4.3. Error Comparison in KITTI Data with Semantic Landmarks and Geometric Landmarks

	Distributed Kalman Filter		Kalman Filter	
Robot	t_{abs} (m)	t_{rel} (%)	t_{abs} (m)	t_{rel} (%)
1	**1.5112**	**3.1109**	1.6015	3.1229
2	1.8687	3.2929	**1.8355**	**3.1389**
3	**1.6647**	3.6140	1.6760	**3.5277**
Mean	**1.6815**	3.3393	1.7043	**3.2632**

In this case, we see little difference between the algorithms as the number of geometric

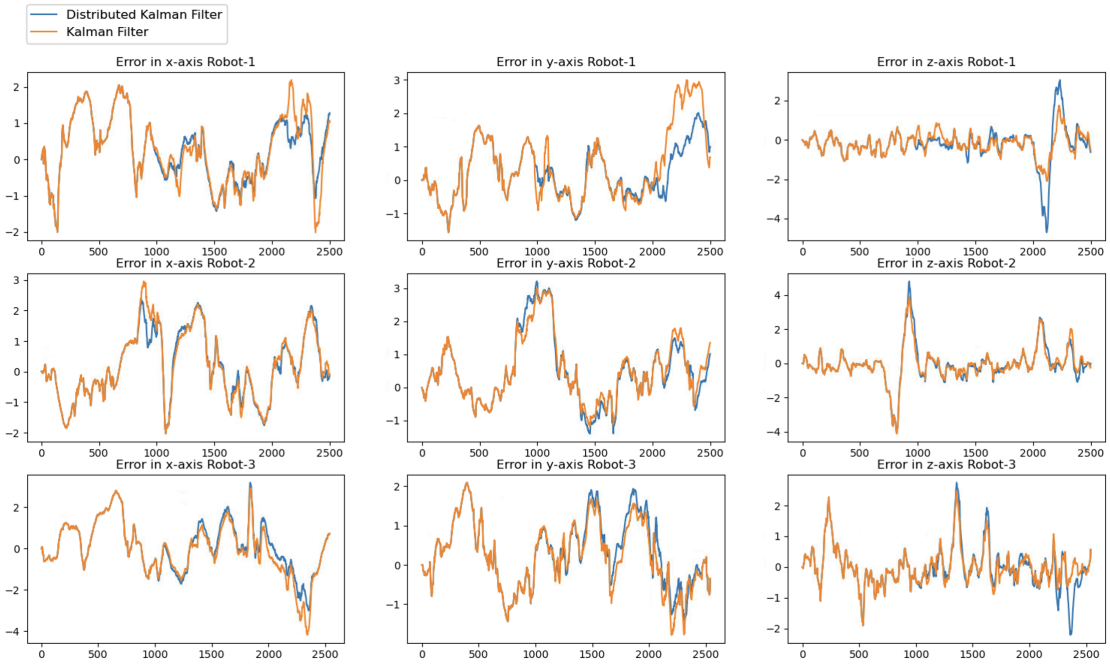


Figure 4.13. xyz Drift Comparison across Time in KITTI Data with Semantic Landmarks and Geometric Landmarks

landmarks (around 7000) is very high compared to the number of semantic landmarks. Thus, very few common features are being shared across the robots, and averaging these does not contribute much towards improving the estimation accuracy.

All the Chapters, in part, are currently being prepared for submission for publication of the material. Cao, Hanwen; Shreedharan, Sriram; Kansal, Shrey; Kumar, Shubham; Nukala, Kishore; Atanasov, Nikolay. The thesis author was one of the researchers/authors of this paper.

Chapter 5

Conclusion

5.1 Contributions

In this thesis, a simple and efficient distributed Kalman filter algorithm for multi-robot systems has been proposed. An accurate sparse global map is created by combining information from all the robots. This is achieved by employing consensus optimization, which enables the robots to come to an agreement about the common landmarks from the environment. The robots share only the sparse common landmarks with their neighbors, leading to efficient communication. Additionally, the distributed algorithm helps to leverage the computation power of all robots, thereby allowing a faster exploration of the environment. Finally, the system is fault-tolerant as the other robots can continue to explore and map the environment even if one robot experiences a breakdown or malfunction.

The proposed algorithm has been tested on a simulation and real-time dataset. In both datasets, when the robots maintain only common landmarks, the additional averaging step in the distributed Kalman filter algorithm improves the accuracy of the estimate significantly. However, it is noted that when the common landmarks that are being shared are significantly fewer when compared to local robot features, then the averaging step has little effect on the overall estimation accuracy.

Overall, this distributed SLAM approach for multi-robot systems takes us a step closer towards mapping large and complex environments quickly and accurately.

5.2 Future Work

The following extensions can be considered for future work. First, a rigorous proof can be derived for the proposed distributed Kalman filter algorithm to prove convergence and stability. The algorithm is currently inspired from the distributed SMD formulation shown in the papers [35], [36], whose proof of convergence is for static networks and therefore do not directly hold as the robots are not stationary and the pdf of the landmarks maintained by the robots differ at each time step.

Another possible improvement is to build a system that does the data association across multiple robots online instead of assuming to be known prior. This can be done by sharing the descriptors for the newly observed landmarks once with the neighboring robots.

Finally, the algorithm can be tested on a larger number of robots, and instead of assuming a fully-connected communication graph, a strongly connected graph can be considered, and to reach a consensus on the common landmarks, different numbers of iterations can be performed in the averaging step.

All the Chapters, in part, are currently being prepared for submission for publication of the material. Cao, Hanwen; Shreedharan, Sriram; Kansal, Shrey; Kumar, Shubham; Nukala, Kishore; Atanasov, Nikolay. The thesis author was one of the researchers/authors of this paper.

Bibliography

[1] Hassan Alhaija, Siva Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision (IJCV)*, 2018.

[2] Nikolay Atanasov. Bayes filter. https://natanaso.github.io/ece276a2022/ref/ECE276A_9_BayesianFiltering.pdf.

[3] Nikolay Atanasov, Roberto Tron, Victor M. Preciado, and George J. Pappas. Joint estimation and localization in sensor networks. In *53rd IEEE Conference on Decision and Control*, pages 6875–6882, 2014.

[4] Saptarshi Bandyopadhyay and Soon-Jo Chung. Distributed estimation using bayesian consensus filtering. In *2014 American Control Conference*, pages 634–641, 2014.

[5] Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2017.

[6] Timothy D Barfoot, James R Forbes, and David J Yoon. Exactly sparse gaussian variational inference with application to derivative-free batch nonlinear state estimation. *The International Journal of Robotics Research*, 39(13):1473–1502, 2020.

[7] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[8] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[9] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 476–481 vol.1, 2000.

[10] W. Burgard, M. Moors, C. Stachniss, and F.E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 2005.

[11] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios,

editors, *Computer Vision – ECCV 2010*, pages 778–792, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[12] Göksel Dedeoglu and Gaurav S Sukhatme. Landmark-based matching algorithm for cooperative mapping by autonomous robots. *Distributed autonomous robotic systems 4*, pages 251–260, 2000.

[13] Frank Dellaert and Michael Kaess. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017.

[14] Arnaud Doucet, Nando de Freitas, Kevin P. Murphy, and Stuart Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. *CoRR*, abs/1301.3853, 2013.

[15] J.W. Fenwick, P.M. Newman, and J.J. Leonard. Cooperative concurrent mapping and localization. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 2, pages 1810–1817 vol.2, 2002.

[16] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325–1339, 2006.

[17] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. Collaborative multirobot localization. In Wolfgang Förstner, Joachim M. Buhmann, Annett Faber, and Petko Faber, editors, *Mustererkennung 1999*, pages 15–26, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[18] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[19] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV)*, 2011.

[20] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. Openvins: A research platform for visual-inertial estimation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4666–4672, 2020.

[21] Trevor Halsted, Ola Shorinwa, Javier Yu, and Mac Schwager. A survey of distributed optimization methods for multi-robot systems. *arXiv preprint arXiv:2103.12840*, 2021.

[22] Syed Riaz un Nabi Jafri and Ryad Chellali. A distributed multi robot slam system for environment learning. In *2013 IEEE Workshop on Robotic Intelligence in Informationally Structured Space (RiiSS)*, pages 82–88, 2013.

[23] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[24] Bernd Kitt, Andreas Geiger, and Henning Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *Intelligent Vehicles Symposium (IV)*, 2010.

[25] Pierre-Yves Lajoie, Benjamin Ramtoula, Yun Chang, Luca Carlone, and Giovanni Beltrame. Door-slam: Distributed, online, and outlier resilient slam for robotic teams. *IEEE Robotics and Automation Letters*, 5(2):1656–1663, 2020.

[26] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.

[27] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Eighteenth National Conference on Artificial Intelligence*, page 593–598, USA, 2002. American Association for Artificial Intelligence.

[28] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[29] Raúl Mur-Artal and Juan D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[30] Angelia Nedich. Convergence rate of distributed averaging dynamics and optimization in networks. *Foundations and Trends® in Systems and Control*, 2(1):1–100, 2015.

[31] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011.

[32] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[33] OpenCV. Feature matching. https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html/.

[34] Manfred Opper and Cédric Archambeau. The variational gaussian approximation revisited. *Neural Computation*, 21(3):786–792, 2009.

[35] Parth Paritosh, Nikolay Atanasov, and Sonia Martinez. Marginal density averaging for distributed node localization from local edge measurements. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 2404–2410, 2020.

[36] Parth Paritosh, Nikolay Atanasov, and Sonia Martinez. Distributed bayesian estimation of continuous variables over time-varying directed networks. *IEEE Control Systems Letters*, 6:2545–2550, 2022.

[37] 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*, 2020.

[38] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 430–443, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[39] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[40] Charles M. Stein. Estimation of the Mean of a Multivariate Normal Distribution. *The Annals of Statistics*, 9(6):1135 – 1151, 1981.

[41] Yulun Tian, Yun Chang, Fernando Herrera Arias, Carlos Nieto-Granda, Jonathan P. How, and Luca Carlone. Kimera-multi: Robust, distributed, dense metric-semantic slam for multi-robot systems, 2021.

[42] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.