Leveraging 360° Cameras and Coordinated Multi-Agent Systems for Scalable Indoor 3D Reconstruction

by

Hoi Chuen Cheng

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfilment of the Requirements for
the Degree of Master of Philosophy
in the Department of Electronic and Computer Engineering

March 2024, Hong Kong

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature redacted

Hoi Chuen Cheng

March 2024

Leveraging 360° Cameras and Coordinated Multi-Agent Systems for Scalable Indoor 3D Reconstruction

by

Hoi Chuen Cheng

This is to certify that I have examined the above MPhil thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

Signature redacted

Prof. Chik Patrick Yue, Thesis Supervisor

Signature redacted

Prof. Andrew Wing On Poon, Head of Department

Thesis Examination Committee

1. Prof. Chik Patrick Yue (Supervisor)	Department of Electronic and Computer Engineering
2. Prof. Ling Shi (Chairperson)	Department of Electronic and Computer Engineering
3. Prof. Shenghui Song	Division of Integrative Systems and Design

Department of Electronic and Computer Engineering
March 2024

ACKNOWLEDGEMENTS

I want to express my most sincere and profound gratitude to my supervisor, Prof. Chik Patrick Yue, for his guidance, support, and encouragement during my research journey. I have never been more impressed by his discipline, willpower, and charisma. Under his supervision, I learned many valuable research skills and life lessons that will fuel my future endeavors.

I also want to extend my deepest gratitude to Prof. Ling Shi for his continued support since my undergraduate studies. His insightful advice on my research direction and methodology was instrumental in my academic journey. His mentorship helped me achieve my first publication and instilled in me valuable research skills that I continue to utilize today.

I am also very grateful to Prof. Shenghui Song for being my thesis committee member and for his guidance during my undergraduate studies. His advice, particularly regarding career choices, has led me to this accomplishment.

I want to express my deep appreciation to my seniors in OWL, Dr. Babar Hussain, Dr. Frederick Ziyang Hong, and Dr. Yiru Wang, for pioneering in and paving the direction for my research topic. I would also like to thank all the members and alumni from OWL, including but not limited to, Dr. Wang Li, Dr. Bo Xu, Dr. Rehan Azmat, Mr. Chongyun Zhang, Ms. Zilu Liu, Mr. Fuzhan Chen, Ms. Xinyi Liu, Ms. Tianxin Min, Mr. Johar Abdekhoda, Mr. Hamed Fallah, Mr. Shaokang Zhao, Mr. Zhendong Li, Mr. Ruitao Ma and Ms. Shuo Feng for your assistance with countless matters and also the joy we shared.

Last but not least, I wish to give my warmest regards to my family and friends, who never cease to support me. Specifically, I wish to express my deepest appreciation to my parents, for being the biggest inspiration of my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Leveraging 360° Cameras and Coordinated Multi-Agent Systems for Scalable Indoor 3D Reconstruction

by Hoi Chuen Cheng

Department of Electronic and Computer Engineering

The Hong Kong University of Science and Technology

Abstract

Large-scale multi-robot 3D reconstruction of indoor environments poses challenges for robotics, virtual reality, construction, and more applications. This thesis proposes a scalable system architecture to advance the distributed multi-robot 3D reconstruction field and addresses problems in robot localization, coordination, and vision processing. A decentralized and communication-based multi-agent path finding framework is first developed to coordinate path planning among fleets of mobile robots through reinforcement learning. We further improve the communication overhead by a Field-of-View (FOV) study and selectively sharing information between neighboring agents. The FOV study reveals that smaller FOV sizes significantly reduce computational costs without compromising performance.

Additionally, an efficient 3D reconstruction pipeline is developed utilizing 360° cameras. This leverages equirectangular projection conversion to facilitate deep learning-based 3D reconstruction models. Experimental evaluations demonstrate that 360° camera data achieves more accurate and efficient scene reconstruction than perspective cameras. In summary, communication in multi-agent path finding, 360° imaging processing, and vision-based 3D reconstruction have been explored and developed in this work. The resulting framework facilitates fleet-scale 3D reconstruction applications in site monitoring and beyond.

CHAPTER 1

INTRODUCTION

1.1 Overview

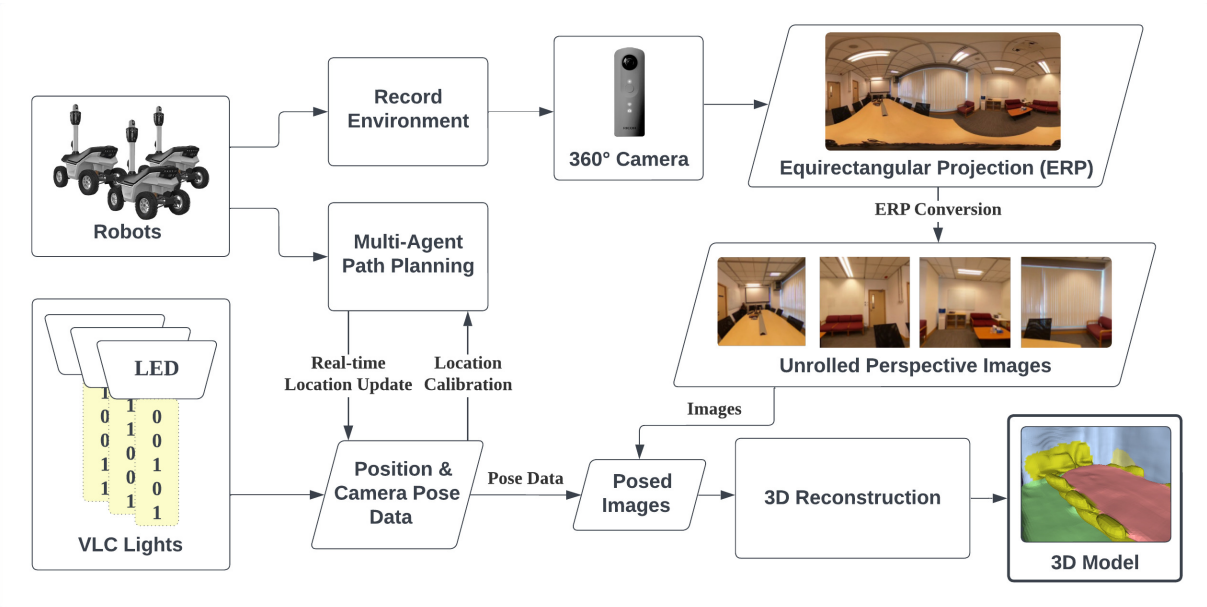

Figure 1.1: Conceptual system architecture of a multi-agent-based indoor 3D reconstruction system

3D reconstruction has increasingly become an essential area of focus due to its ability to generate 3D representations from images or depth data. Applications for 3D reconstructions are widespread, spanning domains such as virtual and augmented reality technologies [5, 6], robotics [7], construction processes involving Building Information Modelling (BIM) [8, 9], and autonomous navigation in self-driving vehicles [10, 11]. Furthermore, automated 3D scene modeling has a significant advantage in maintaining objects' fine-grained details and spatial contexts without much human labor. This technology enables robots with 3D perception to better perceive their surroundings, while humans can gain a deeper understanding of digital environments through 3D visualization.

As the intelligent construction industry and related fields such as surveillance, robotics, and inspections continue to develop and expand, core enabling technologies that power these sectors are becoming increasingly crucial. The degree of intelligence integrated into systems directly af-

fects the quality and efficiency of task completion and indirectly impacts value creation. Within the construction domain, conventional inspection methods that rely heavily on human workers utilizing handheld devices inevitably result in many issues. These include low efficiency due to the time required to gather data, high costs associated with personnel and equipment usage, an inability to provide support over extended periods consistently, uncertain coverage of inspection areas, and safety risks when people must enter dangerous or unstable sites. Advancements in technologies like machine learning have the potential to help mitigate these challenges by automating inspection and localization functions.

Coordinating multiple robots has considerable potential to improve the accuracy and efficiency of 3D reconstruction efforts. This multi-agent approach is incredibly impactful in more complex and more extensive environments. Beyond enhancing productivity, fleets can save time by closely monitoring construction sites to ensure work adheres to schedules and safety compliance. Effective oversight enables improved collaboration between project members, reducing delays and expensive issues. With up-to-date tracking, construction firms can meet intended timelines and deliver projects as planned. As technologies advance, leveraging multiple robots for 3D reconstruction becomes increasingly essential.

However, large-scale indoor 3D modeling presents challenges. Applications must first resolve the Multi-Agent Path Finding (MAPF) problem to define collision-free trajectories within operational spaces. Calculating optimal MAPF solutions suffers enormous computational complexities, spurring algorithm development. Nonetheless, directly employing such solutions fails to accommodate fleets efficiently for expansive areas. Some techniques adopt search strategies like Conflict-based Search, while others reframe MAPF as satisfiability problems. However, systems using these methods typically scale only for small agent teams. The immense overload of direct employment introduces barriers to coordinating large reconstruction fleets. Decentralized methods modeling MAPF as partially observable games merit exploration to distribute computation across aware, autonomous robots.

While 3D reconstruction techniques commonly rely on complicated arrangements of numerous cameras or specialized equipment [12] to function correctly, the popularity and uptake of 360° cameras have grown significantly in recent times. 360° cameras provide a more all-encompassing Field-of-View (FOV) than regular perspective cameras, rendering them appropriately suited for various uses. For example, 360° cameras are now commonly employed in the construction sector since they permit a more practical approach to overseeing entire construction sites through a single vantage point.

360° cameras provide additional key benefits for large-scale 3D reconstruction using multi-

robot systems. Unlike traditional cameras requiring complex arrangements, a single 360° camera simultaneously captures a comprehensive spherical panorama of the entire environment. This allows each robot to acquire a holistic view using only one image, reducing redundant data collection across a fleet. The wide FOV also facilitates collaboration, where robots can efficiently localize themselves and communicate intentions based on the improved awareness of other agents and obstacles. Together, these attributes of 360° cameras make them especially well-suited for scalable 3D reconstruction applications with coordinated robot teams, in contrast to traditional multi-camera systems that cannot capture and leverage such holistic spherical contexts.

To address the calibration complications arising when utilizing cameras featuring an ultra-wide FOV, we introduce a potential method for handling the calibration challenges. Our approach eliminates the need for oversized checkerboard calibration patterns that are typically necessary for modeling lens distortion across an immense angular range. Concurrently, we project the commonly used yet distorted Equirectangular Projection (ERP) format of 360° panoramic imagery onto a cube map resembling orthographic perspective views. This projected representation facilitates compatibility with deep learning networks pre-trained on undistorted perspective images, broadening the applicability of low-cost commercial 360° cameras to computer vision tasks involving 3D scene reconstruction. By resolving distortion and aligning the panoramic visual domain with established deep learning models, our technique aims to increase further the usefulness of affordable wide-angle cameras in applications ranging from virtual/augmented reality to robotic vision.

1.2 Thesis Organization and Contributions

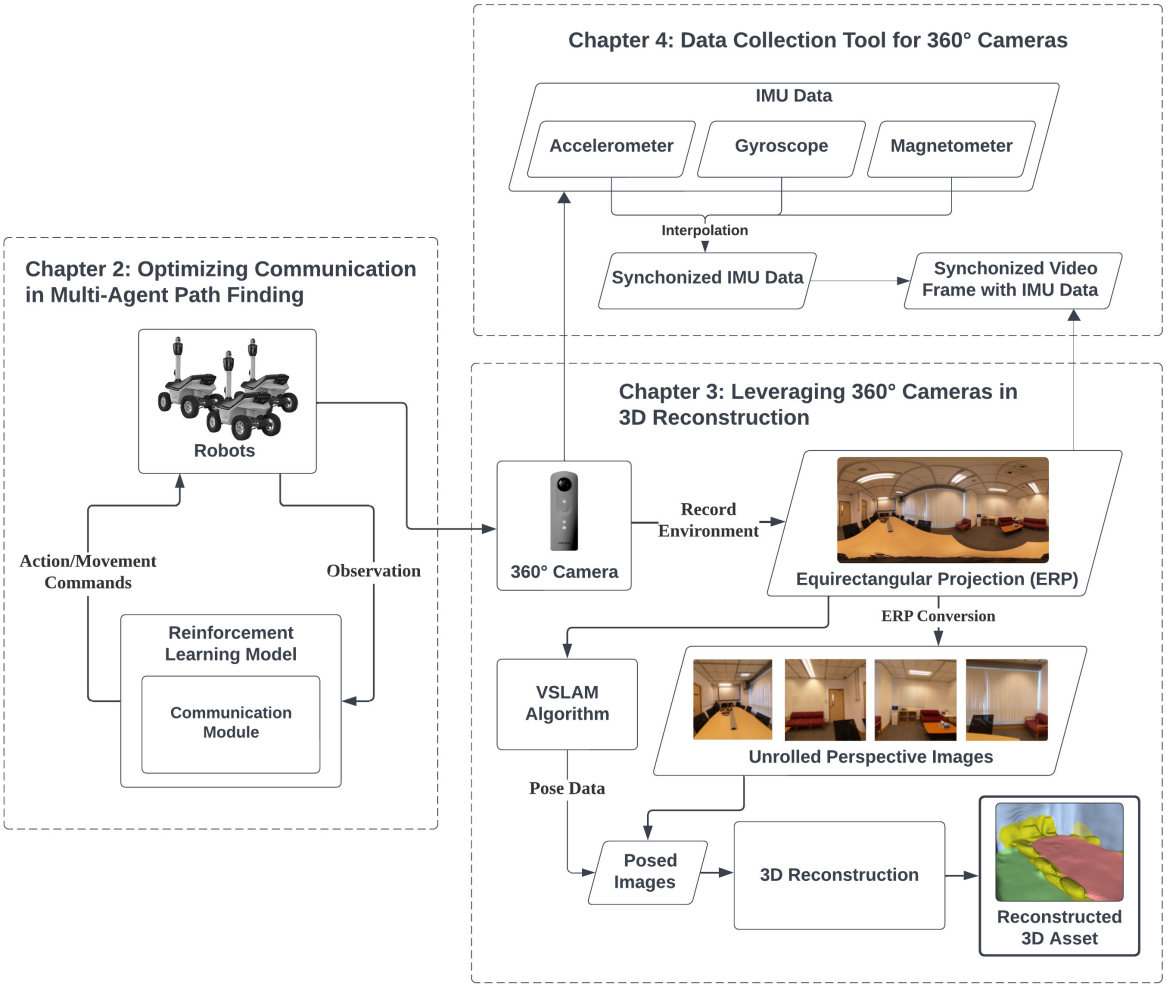

Figure 1.2: Thesis organization and contribution.

This thesis focuses on robot localization, 3D reconstruction using 360° cameras, and a data collection tool for 360° cameras. An illustration of the thesis organization is presented in Figure 1.2.

1.2.1 Chapter 2: Optimizing Communication in MAPF

This chapter presents a decentralized and communication-based MAPF framework utilizing reinforcement learning. This framework aims to coordinate path planning among fleets of mobile robots operating in a partially observable grid environment. The focus is improving communication overhead through selective information sharing between neighboring agents and a Field-of-View (FOV) study.

1.2.2 Chapter 3: Leveraging 360° Cameras in 3D Reconstruction

This chapter proposes a novel vision-based 3D reconstruction pipeline for 360° cameras. The pipeline utilizes equirectangular projections to facilitate the application of deep learning-based models for 3D reconstruction. Experimental evaluations demonstrate the superiority of 360° camera data in achieving more accurate and efficient data collection compared to conventional perspective cameras.

1.2.3 Chapter 4: Data Collection Tool for 360° Cameras

This chapter describes a data collection tool designed to acquire video and IMU data from 360° cameras to facilitate various experiments.

1.2.4 Chapter 5: Conclusion and Future Works

This chapter summarizes the thesis and discusses potential future works in communication-based MAPF and 3D reconstruction with 360° cameras.

CHAPTER 2

OPTIMIZING COMMUNICATION IN MULTI-AGENT PATH FINDING

2.1 Introduction

The utilization of multiple robots collaborating on tasks offers immense potential to enhance the accuracy and efficiency of 3D reconstruction efforts significantly. This multi-agent approach proves to be particularly impactful in complex and expansive environments where a single robot may face limitations. By leveraging multiple robots' collective capabilities and diverse perspectives, the accuracy of the collaborative 3D reconstruction can be greatly improved, capturing a more comprehensive representation of the environment.

In addition to boosting accuracy and productivity, utilizing multiple robots equipped with up-to-date tracking and 3D reconstruction technologies empowers construction firms to adhere to their intended timelines and deliver projects as planned. By continuously capturing and analyzing data from various robot perspectives, construction processes can be closely monitored, enabling proactive decision-making and timely adjustments. This level of real-time tracking and analysis facilitates efficient coordination among the robots. It enables project managers to make informed decisions, optimize resource allocation, and promptly address emerging issues. Furthermore, a coordinated multi-agent or multi-robot system can significantly offload human workers for a better work environment. In the context of construction sites, where different kinds of accidents can happen, a safer and more efficient solution must be provided to the workers. Using multiple robots, larger construction site areas of hazardous zones can be covered simultaneously, enabling comprehensive monitoring from diverse angles and viewpoints. This enhanced surveillance capability contributes to improved safety measures for personnel, mitigating the risk of accidents or injuries.

The successful execution of applications involving 3D reconstruction in expansive indoor or outdoor areas, whether construction sites or home environments, necessitates coordinating multiple robots to model the space comprehensively and efficiently. However, to enable such applications, we must first address the challenges posed by the Multi-Agent Path Finding (MAPF)

problem [13]. This problem involves determining a collection of collision-free paths within the operational space, which presents significant computational complexities.

Various algorithms have been developed to tackle the MAPF problem. However, directly applying these algorithms may not efficiently accommodate fleets of robots aiming to leverage their collective capabilities for large-scale indoor 3D modeling of expansive areas. Some approaches adopt search-based strategies like Conflict Based Search (CBS) [14, 15], while others reformulate the problem as a Boolean Satisfiability Problem (SAT) [16]. However, systems relying on these methods typically have limitations in scaling up to many participating agents. The computational burden of directly employing these techniques introduces substantial barriers to coordinating large teams of robots engaged in 3D reconstruction tasks. The sheer scale of the operational space and the number of participating robots can overwhelm existing algorithms, resulting in inefficient planning and increased computation time.

Efforts are underway to leverage techniques such as distributed optimization, swarm robotics, and decentralized control to coordinate and plan large-scale multi-robot systems for 3D reconstruction. These advancements address the computational overload and scalability issues inherent in existing methods. Particularly, researchers have addressed the scalability issues of centralized techniques for MAPF by exploring decentralized execution methods [17–20] using Imitation Learning (IL) and Reinforcement Learning (RL). These methods model MAPF as a partially observable Markov game, allowing agents to make decisions based on localized observations instead of relying on a centralized planner with global information. This reduces overhead and computational demands, enabling coordination among agents as team sizes increase. RL-based methods incorporate behavior cloning [17, 18] to minimize divergence during training, while other techniques utilize heuristics [21] to expedite convergence. These decentralized techniques show promise in enabling collaborative large-scale 3D reconstruction applications using fleets of robots.

Recent studies have significantly improved coordination in multi-agent systems through enhanced communication strategies. Previous approaches emphasized broadcast messaging [20–22], where agents indiscriminately share information with surrounding agents. While broadcasting messages had advantages over earlier methods, it generates significant overhead as not all exchanged data equally contributes to decision-making. Additionally, extraneous information in messages could confuse agents and potentially hinder learning dynamics.

To address these challenges, frameworks [23–27] have been developed to minimize communication costs. In particular, [27] explores selective communication strategies in the MAPF setting, where agents only communicate with neighboring agents expected to influence imme-

diate actions. This approach balances coordination and efficiency in large fleets and optimizes information sharing between agents. This offers a promising solution to reduce the overhead associated with broadcast schemes while enabling cooperation necessary for distributed multi-agent coordination.

While the advancement of the communication-based MAPF framework brought huge improvements in solving MAPF problems, researchers still overlooked the importance of choosing a suitable FOV for agents in a multi-agent system. The FOV setting of a multi-agent system is essential for the following reasons.

1. Perception and Navigation: The FOV of an agent determines the ability to perceive information and be aware of opportunities in a given environment. It helps the agent to concentrate on its immediate surroundings, with the same effect as decreasing the detection distance of a LiDAR sensor. In the context of navigation and obstacle avoidance, information perceived is essential for the agent to find the best paths to its goal and decide how to maneuver itself in the environment.
2. Energy Efficiency: The FOV of an agent can greatly affect its energy efficiency. A larger FOV allows the agent to take in more information with a broader perspective, but it also demands more computational resources to maintain and processing power to operate. On the contrary, a smaller FOV reduces the agent's computation load while providing a more focused perspective. This is particularly relevant for resource-limited environments, where the agent might need to work for a prolonged period without access to an energy source.
3. Coordination and Collaboration: The FOV plays a crucial role in the agent's ability to collaborate with neighboring agents. When two agents have overlapping FOVs, they can effectively share information and communicate, enabling them to work together towards a shared objective. However, an excessively large FOV can result in unnecessary communication and potentially confuse the agent. Finding the right balance in FOV size is important to optimize collaboration while avoiding information overload.
4. Security and Privacy: FOV addresses security and privacy concerns by controlling the data that agents can access. By limiting the information an agent can perceive or share, FOV safeguards sensitive data, preventing unintentional disclosure or privacy issues. Additionally, FOV acts as a protective measure against malicious actors by restricting their access to data. This control over the amount of data accessible through the FOV helps enhance security and privacy in multi-agent systems.

While a larger FOV provides more information to an agent, there are advantages to using

a smaller FOV. A smaller FOV requires less computational load, making decision-making and reaction times faster, which is especially beneficial for agents with limited hardware resources. Moreover, a narrower FOV allows the agent to focus on its immediate, more critical surroundings. Additionally, a smaller FOV can conserve energy, prolonging battery life, which is particularly advantageous in settings with limited charging sources. In some cases, a smaller FOV can also address security concerns by restricting the data perceived and transmitted by the agent. Therefore, when designing and deploying robotic multi-agent systems, careful consideration must be given to the size and scope of the FOV. Ideally, the FOV should be sufficient for navigation and collaboration while remaining energy-efficient.

In our experiments studying the selection of FOV and its impact on performance and communication overhead in MAPF tasks, we discovered that different FOV settings could significantly affect the system's performance. Additionally, we propose a way to evaluate performance in terms of computation cost, enabling researchers and engineers to understand the trade-offs when designing cost-efficient systems. These findings are particularly relevant in technological advancements, where multi-agent 3D reconstruction is gaining prominence. The continuous growth in robot autonomy, communication, and coordination algorithms enables seamless collaboration between robots, further enhancing their collective capabilities. With the ability to navigate multiple agents efficiently through various optimizations, industries such as construction, manufacturing, and surveillance can benefit from improved productivity, ultimately leading to cost savings and better outcomes.

2.2 Related Works

2.2.1 Multi-Agent Path Finding (MAPF)

While approximate optimal solutions are available for MAPF [28], it remains an NP-hard problem. Traditional MAPF planners can generally be categorized into coupled, decoupled, and dynamically coupled methods. Coupled methods, such as A*, face significant challenges due to the curse of dimensionality. On the other hand, decoupled methods, like the one described in [29], can efficiently plan and modify paths for collision avoidance in low-dimensional search spaces. However, it's important to note that these decoupled techniques may not guarantee completeness, as they only consider a limited portion of the joint configuration space [30]. Dynamically-coupled techniques allow broader agent interactions while avoiding planning in the entire configuration space. For instance, Conflict Based Search (CBS) and its variants [14,

15, 31] avoid searching in higher dimensional space by defining a set of constraints.

2.2.2 MAPF via RL

Significant progress has been made in single-agent path planning using RL, such as [32, 33]. Recently, researchers have turned their attention to applying RL to solve MAPF problems, often relying on expert guidance from existing planners. For example, PRIMAL [17] utilizes the OD-recursive-M^* (ODrM*) planner [34], MAPPER [19] uses A*, and Global-to-Local Autonomy Synthesis (GLAS) [35] incorporates a graph-based planner [36]. Specifically in PRIMAL, an Asynchronous Advantage Actor-Critic (A3C) network is the RL module, while behavior cloning is performed using the ODrM* planner. However, these expert planners face computational challenges as the number of agents increases, and the paths designed for single-agent environments may not be optimal for multi-agent scenarios.

One possible solution is to leverage communication to foster collaboration among agents. Recent examples include Targeted Multi-Agent Communication (TarMAC) [24] and Graph Neural Network (GNN) approaches [20], which exploit communication to enhance collaboration. Another approach, Decision Causal Communication (DCC) [27], builds upon the ideas of DHC [21] for selective communication with neighboring agents. This selective communication reduces redundancy in information exchanges and significantly reduces communication overhead.

2.3 Problem Setup

Models are trained and evaluated using the classical MAPF benchmark described in [13]. The benchmark defines the following guidelines:

1. Agents in the environment can perform one of five actions on each time step: up, down, left, right, or stop.
2. An endpoint for evaluation is reached when all agents have arrived at their designated destinations or the maximum number of steps is reached.

2.3.1 Formal Definition of MAPF Problem

In the classical MAPF problem, an undirected graph $G = (V, E)$ represents the environment, where V is the set of vertices and E is the set of edges. The start and goal vertices for n agents are defined as $\{s_1, \ldots, s_n\} \in V$ and $\{d_1, \ldots, d_n\} \in V$, respectively. Agent movements correspond to traversing edges in the graph (i.e., $(v, v') \in E$). An action sequence $\pi_i = \{a_1, \ldots, a_t\}$ describes

the actions taken by agent i from the start to time t. The location of agent i at time t, $l_i(t)$, is determined by sequentially applying the actions in π_i starting from s_i. A MAPF solution consists of n action sequences $\pi = \{\pi_1, \ldots, \pi_n\}$ that determine the paths for all n agents from their start to goal locations while avoiding collisions, represented as occupying the same vertex at the same time step. The task is to find collision-free action progressions π_i for each agent i that solve the combined constraints of the MAPF problem over G.

2.3.2 Types of Conflicts

As described in [13], the MAPF problem allows for different definitions of conflict between agents. However, the works surveyed in [13] and the present study all prohibit the following two types of collisions:

1. Vertex collision: This occurs when two agents i and j attempt to occupy the same vertex at the same time t, such that $l_i(t) = l_j(t)$.
2. Edge collision: An edge collision arises if agents i and j try to traverse the same edge $(v, v') \in E$ simultaneously, meaning $l_i(t) = l_j(t)$.

Specifically, the approaches considered forbid scenarios where multiple agents would intersect at either a shared position on G during any given time of agent motion planning and execution.

2.3.3 MAPF Environment

The predefined MAPF problem occurs in a discrete grid environment where the n agents navigate simultaneously. Within a $k \times k$ map containing n agents, each agent is assigned a unique start location and goal position. Agents can traverse this grid world through the four possible movements at each time t, or remaining stationary. Consequently, the action space available to each agent contains five options (i.e., up, down, left, right, or stop).

To model real-world deployment scenarios more realistically, we consider the MAPF task from the partially observable perspective of each agent. This constrained FOV simulates an agent's limited awareness when making navigation decisions. Details regarding FOV settings used in this work are provided in Section 2.5.1.

The reward structure, shown in Table 2.1, was adopted from DHC [21] and DCC [27]. To encourage agents to cede paths to others when there are blockages, remaining on non-goal vertices is penalized. While such behavior is not as heavily penalized as in the PRIMAL [17] and

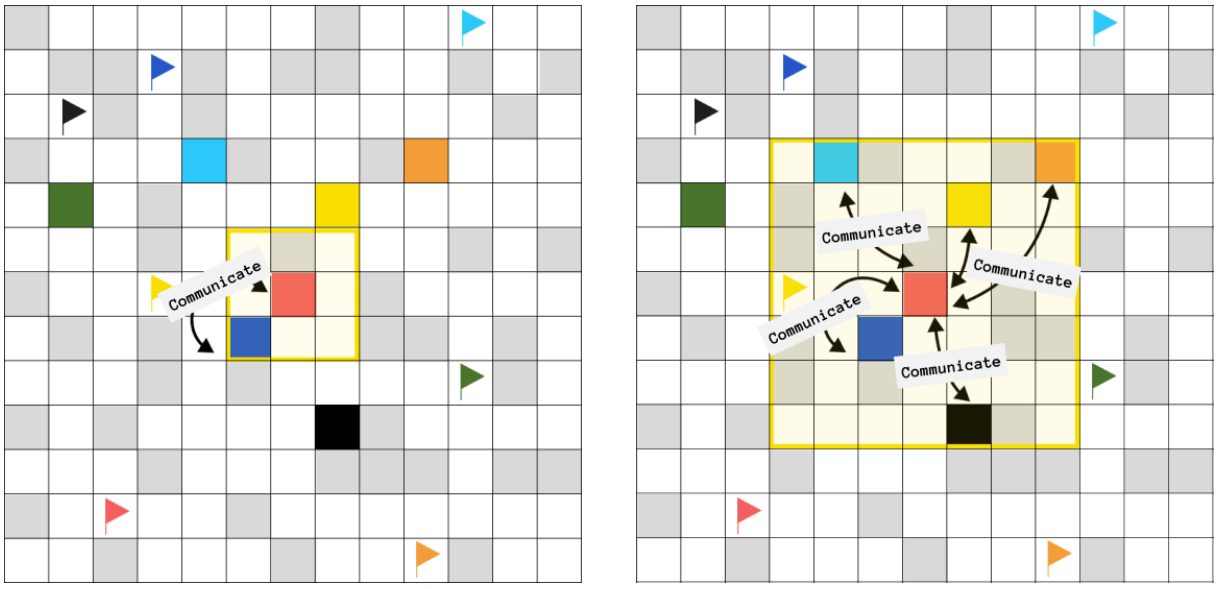

Figure 2.1: The image showcases two different field of view (FOV) settings: a 3×3 FOV on the left and a 7×7 FOV on the right. A red square represents the central agent, while other agents are depicted as colored squares, with their respective destinations indicated by colored flags. Grey squares represent obstacles, while white squares represent walkable tiles. Agents falling within the yellow area, which represents the FOV of the red agent, can communicate in a request-reply style [1].

MAPPER [19]. This reward design aims to allow the model to learn coordinated multi-agent behaviors focused on collective performance throughout decentralized decision-making.

Table 2.1: Reward structure

Actions	**Reward**
Move (Up/Down/Left/Right)	-0.075
Stay on goal vertices	0
Stay on non-goal vertices	-0.075
Collision	-5
Completion	3

2.4 Architecture of the RL Model

The RL model architecture used in this work adopts key elements from [21] and [27]. In particular, [21] enabled inter-agent communication while [27] featured selective communication inspired by Individually Inferred Communication [23]. In our experiment, the decentralized multi-agent RL model contains four components. First, an observation encoder processes local observations for each agent. Next, a decision causal unit [27] determines which neighbor to communicate with. A communication block [21] then shares information amongst relevant agents

and outputs message embeddings. Finally, a Dueling Deep Q-Network (DQN) [37] processes these embeddings and selects the optimal action.

Each agent receives a 6-channel input tensor of dimensions $l \times l \times 6$ as input, where $l \times l$ represents the size of the agent's FOV. This input contains two binary matrices indicating the locations of other agents and obstacles observed within the agent's local view. It also includes four heuristic channels derived from DHC [21], corresponding to the up, down, left, and right actions. Within these action channels, positions that move the agent closer to its goal are marked with a one, while other locations are marked with zero.

First, the observation encoder generates modified local observation embeddings $\{e_{i,-j}\}_{j\in\mathbb{N}_i}$, based on excluding each neighbor j from i's observation $\{o_{i,-j}\}_{j\in\mathbb{N}_i}$. $\mathbb{N}_i$ is defined as the neighboring agents within agent i's FOV. To determine whether communication should be triggered between agent i and its neighbors $\mathbb{N}_i$, the decision casual unit assesses neighbor influences on i's decision making. These embeddings are input to the Dueling DQN, which produces provisional actions $\tilde{a}_i$ from the original observation, and $\{\tilde{a}_{i,-j}\}_{j\in\mathbb{N}_i}$ from the modified observations lacking individual neighbor j. By contrasting $\tilde{a}_i$ to $\{\tilde{a}_{i,-j}\}_{j\in\mathbb{N}_i}$, the communication scope is defined as neighbors whose exclusion causes $\{\tilde{a}_{i,-j}\}_{j\in\mathbb{N}_i}$ to differ from $\tilde{a}_i$, i.e.,

$$\mathbb{C}_i = \{j|\tilde{a}_i \neq \tilde{a}_{i,-j}\}_{j\in\mathbb{N}_i}. \tag{2.1}$$

This identifies agents that are likely to impact i's navigation, thus enabling selective communication.

To enhance efficiency, communication between neighbors takes place using a request-reply approach. Given a defined communication scope $\mathbb{C}_i$, both the modified observation embedding e_i generated by agent i's observation encoder and the relative positions l_i of neighbors, are passed from agent i to each corresponding agent $j \in \mathbb{C}_i$ within its FOV. Inside the communication block, the concatenation of e_i and l_i is projected into key and value vectors with matrices W_K^h (Equation 2.2) and W_V^h (Equation 2.3), respectively. Similarly, the message e_j undergoes projection into a query vector using matrix W_Q^h. For each agent j, the receiving scope $\mathbb{O}_j$ consists of agent $\bar{i}$ where $\mathbb{O}_j = \{\bar{i}|j \in \mathbb{C}_{\bar{i}}\}$. The set $\mathbb{O}_{j+}$ is represented as $\{\mathbb{O}j, j\}$. The calculation of the relation between agent j and each sending agent $\bar{i} \in \mathbb{O}j+$ in the h-th attention head is determined by

$$\mu_{j\bar{i}}^h = softmax\left[\frac{W_Q^h e_j \cdot (W_K^h[e_{\bar{i}}, l_{\bar{i}}])^T}{\sqrt{d_K}}\right], \tag{2.2}$$

with normalization through $\sqrt{d_K}$ and d_K as the key dimension. The outputs of the attention

heads are combined across H heads through concatenation and then fed into a neural network layer f_o to produce the final output $\hat{e}_j$:

$$\hat{e}_j = f_o\left[concat\left[\sum_{\bar{i}\in\mathbb{O}_{j+}} \mu_{j\bar{i}}^h W_V^h [e_{\bar{i}}, l_{\bar{i}}], \forall h \in H\right]\right]. \tag{2.3}$$

The Gated Recurrent Unit (GRU) is utilized to aggregate the output $\hat{e}_j$ and the initial observation embedding e_j. The resulting output of the GRU, denoted as $e_i^{'}$, is then used as a new input message for the subsequent round, repeating the operations described in Equations 2.2 and 2.3. The final output of the entire communication module is represented as $e_i^{''}$. By leveraging the GRU, inputs are aggregated across time steps, allowing for the propagation of updated neighbor-aware embeddings throughout the sequential request-reply communication process. Through this targeted exchange of embedding information and spatial context between connected agents, efficient cooperative decision-making in multi-agent coordination is enabled.

A Dueling DQN model is employed to estimate the Q-value, utilizing advantage functions and considering the outputs from the communication block. Initially, we compute the mean of the advantages

$$\frac{1}{|N|}\sum_a A(e_i^{''}))$$

, with N representing the size of the action space. To get the final Q-values, we perform a stabilization, which involves subtracting the advantage value from the advantage mean and incorporating the state value as a final adjustment. The formula is described in

$$Q_{i,s,a} = V_s(e_i^{''}) + \left[A(e_i^{''}) - \frac{1}{|N|}\sum_a A(e_i^{''}))\right]. \tag{2.4}$$

Once the Q-values are obtained from the DQN model, we calculate a multi-step Temporal Difference (TD) error. The TD error is then utilized to update the model parameters by minimizing the mean squared error between the predicted Q-values and the total discounted rewards. The process is illustrated in

$$L(\theta) = MSE(R_t - Q_{s_t,a_t}(\theta)), \tag{2.5}$$

The total discounted rewards, denoted as R_t, are the sum of rewards at each time step. This summation includes future rewards up to a certain horizon, as captured by the term $R_t = r_t + \gamma r_{t+1} + \cdots + \gamma^n Q_{s_{t+n},a_{t+n}}(\bar{\theta})$. In this equation, γ represents the discount factor applied to future

rewards, and $\bar{\theta}$ represents a periodic copy of the model parameters θ maintained by the target network.

2.5 Experiments

2.5.1 FOV Settings

As previously mentioned, our multi-agent RL approach employs a decentralized, communication-enabled model to facilitate robot coordination. A crucial element of this model is the communication module, which enables information sharing between agents based on their designated communication scopes. The agents' FOV is pivotal in defining these communication scopes, as depicted in Figure 2.1. The FOV corresponds to the perception range of the agents, such as that provided by LiDAR or ultrasonic sensors. This highly affects an agent's perception, navigation, and effective collaboration with other agents in its environment. A wider FOV grants the agent access to more information, offering a broader perspective of its surroundings.

Conversely, opting for a narrower FOV presents advantages, such as reduced computational load, enhanced energy efficiency, improved collaboration, and heightened security considerations. Optimizing the FOV of the agents becomes increasingly important as it strikes a balance between reducing computational demands associated with coordinating larger communication scopes and enhancing overall performance. Therefore, investigating the impact of different FOV settings on performance and computational requirements contributes to the design of scalable multi-agent systems.

FOV governs the information intake and communication scope of agents during MAPF tasks. Our research examines the impact of FOV on performance and communication overhead. Assuming a fully observable environment is impractical in large-scale environments like warehouses and factories, we must ensure that the FOV size or the partially observable view of the agent, denoted as $l \times l$, is smaller than the map size, represented as $k \times k$, i.e., $l < k$.

Minimizing the FOV intuitively reduces the amount of information, conserving bandwidth for path planning and networking. An optimal FOV should maximize energy efficiency without significantly compromising performance. To explore more energy-efficient FOV options, we conducted experiments with various sizes on top of the 9×9 FOV used in DCC, DHC, and other related research. It is worth noting that these works have overlooked the exploration of different FOV sizes and have defaulted to a fixed size of 9×9. In light of this, we have chosen the 9×9 FOV configuration as our initial setup for baseline evaluation and analysis. In addition, the FOV

width and height must be odd to center the agents, making 3×3 the minimum feasible size. Additionally, we investigated the impact of increased information intake on decision-making and performance by evaluating larger FOVs.

Our curriculum training began with 2 agents on a 10×10 map and progressed to 20 agents on a 40×40 map, with a batch size of 128. To conduct our experiments, we employed the following FOV configurations: 3×3, 5×5, 7×7, 9×9, and 11×11.

2.5.2 Test Settings and Hardware Specifications

To evaluate our approach's navigation ability and generalization, we conducted tests on maps of sizes 40×40 and 80×80, both containing 30% obstacle vertices. Including the 80×80 map size in the tests aimed to assess the generalization capabilities of our model and its ability to scale, considering that our training process exclusively utilized maps of sizes smaller than or equal to 40×40. For each pair of map size and number of agents, we randomly generated 500 scenarios, and the number of agents in the set 4, 8, 16, 32 are tested. All test cases have a maximum list of 256 steps for each agent. All metrics were calculated as averages across the test cases. 6 Intel Xeon Gold 6230 CPUs and 2 RTX Quadro RTX 6000 GPUs from HKUST's HPC3 are used during the model training.

2.6 Result

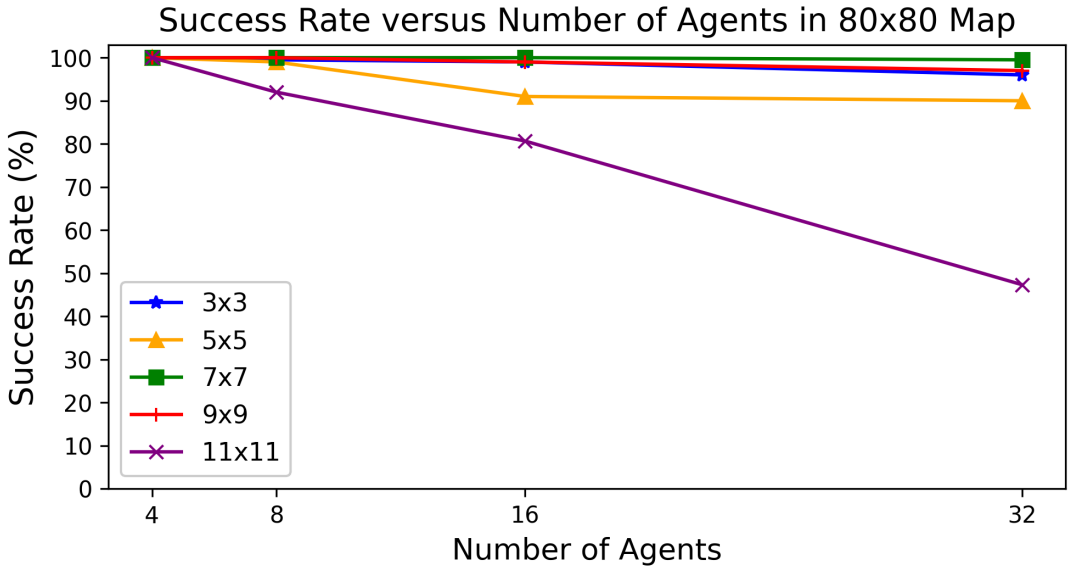

Figure 2.2: Success rate with varying FOV settings in 80×80 Map

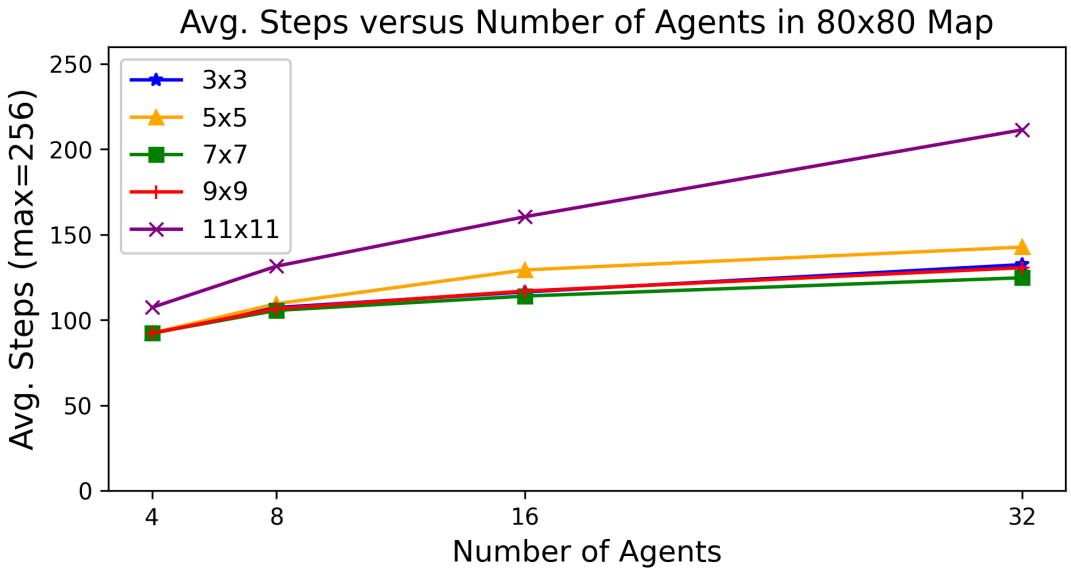

Figure 2.3: Average steps with varying FOV settings in 80×80 Map

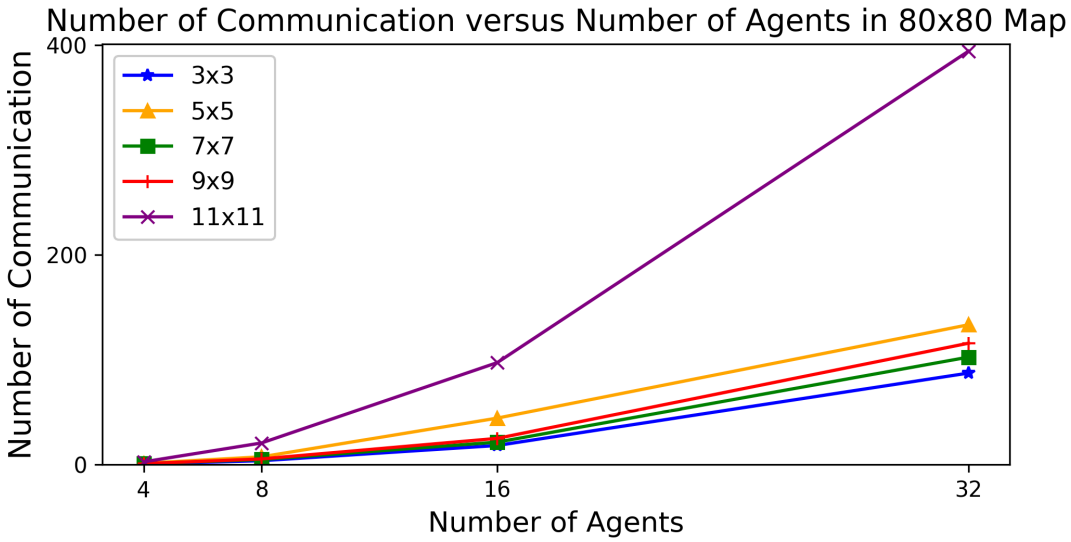

Figure 2.4: Number of communications with varying FOV settings in 80×80 Map

2.6.1 Success Rate, Average Steps and Number of Communications

We assess the performance of various FOV sizes by measuring the success rate, the average number of steps taken, and the number of communications required. The success rate represents the percentage of agents that reach their destination within the maximum allotted steps (256 steps). The average steps measure the mean number of steps taken to complete a MAPF task across all agents. The number of communications indicates the total count of request-reply pairs generated during a MAPF task.

Figures 2.2 and 2.3 depict a comparison of the success rate and average steps, respectively, for different FOV sizes in an 80×80 map. On average, the 7×7 FOV outperforms the original baseline (9×9) by 4.2% in terms of success rate and 3.0% in average steps. Despite receiving the least amount of information, the 3×3 FOV exhibits a relatively small sacrifice in performance across the mentioned metrics. Specifically, the 3×3 FOV achieves a success rate 5.85% lower than the 7×7 FOV and 1.65% lower than the 9×9 FOV. In terms of average steps, the 3×3 FOV demonstrates 4.2% more steps compared to the 7×7 FOV and 1.0% more steps compared to the 9×9 FOV. Figure 2.4 presents the number of communications conducted during various MAPF tasks in an 80×80 map. With the 3×3 FOV, which receives the least amount of surrounding information, the communication requirements were reduced, resulting in a decrease in overheads by 28.9% compared to the 9×9 baseline and 24.4% compared to the highest-performing 7×7 FOV. The 3×3 FOV's ability to strike a balance between minimal communication and small performance impacts makes it a preferable choice in scenarios where bandwidth is constrained. This is particularly relevant when deploying numerous networked robots with limited communication devices.

The results for the 40×40 map are presented in Appendix A.

2.6.2 Network Step Time

In highly efficient robotic systems, large FOVs are often utilized despite the associated high computation costs. However, many real-world multi-robot systems often encounter constraints on computing power due to limited resources. Hence, carefully considering the computational cost of different FOV options becomes crucial. This study employs network step time as a metric to evaluate computation cost. Network step time measures the duration an agent requires to select an action given an observation, and higher decision-making computation costs are indicated by longer step times.

Figure 2.5 compares network step times across various FOV sizes. The results confirm that

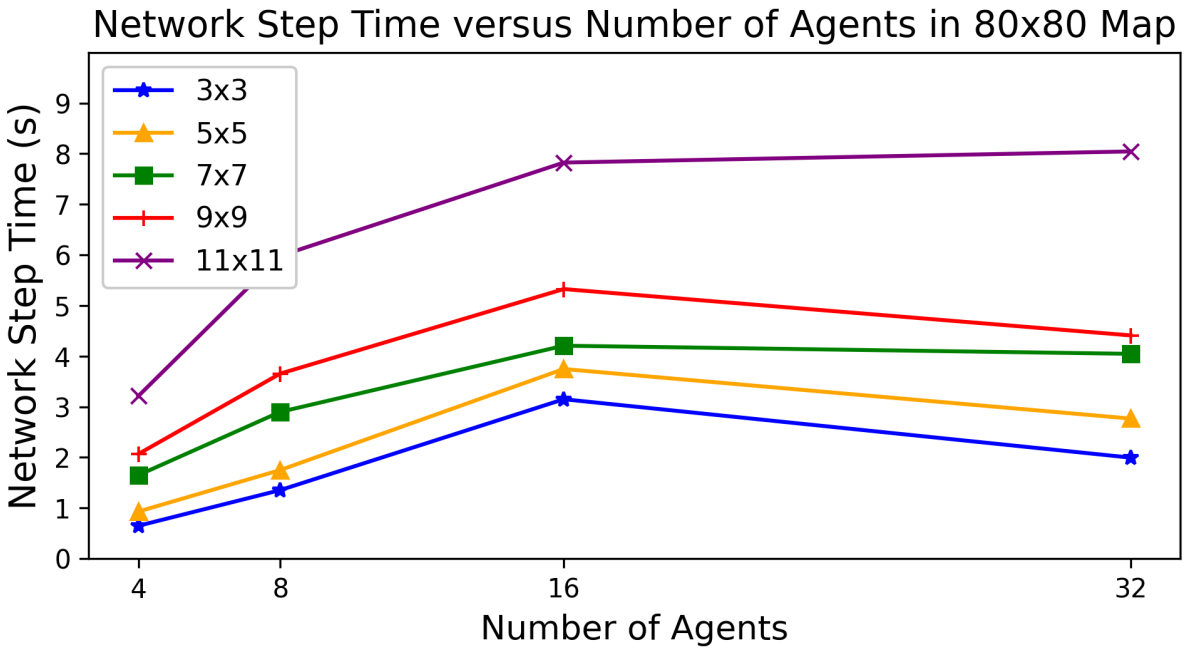

Figure 2.5: Network step time with varying FOV settings in 80×80 Map

step time scales with FOV size, with the 3×3 FOV exhibiting the lowest computation cost among the tested FOVs, while the 11×11 FOV demonstrates the highest. These findings hint at the impact on computational load when using different sizes of FOV for multi-agent systems.

Table 2.2: Network step time ratio of different FOV

FOV	**Network Step Time Ratio**
3×3	1
5×5	1.34
7×7	1.95
9×9	2.27
11×11	3.69

When selecting the appropriate FOV, considering computation cost alone is not enough, as it fails to account for the impact on performance. This realization has led to the need for a metric that considers both performance and computation cost simultaneously, which is discussed in detail in Section 2.6.3. To further standardize the comparison of step times across different FOVs, we normalized all the step times for a given combination of map size and number of agents by using the shortest step time as a reference. The resulting average ratios of networked step times for the tested FOVs are presented in Table 2.2.

2.6.3 Normalized Success Rate

To simultaneously consider computation cost and success rate, this study introduces a novel metric called the normalized success rate, denoted as r'. The value of r' is calculated by dividing the original success rate r by the normalized network step time $\bar{t}$, expressed as follows:

$$r' = \frac{r}{\bar{t}} \tag{2.6}$$

Figure 2.6 presents the normalized success rate for all tested FOVs across map size of 80×80. Our analysis considers both success rate and computation cost metrics, recognizing that optimal performance requires satisfactory outcomes in both aspects. The results indicate that the 3×3 FOV exhibits the highest level of robust performance in terms of normalized success rate, followed by the 5×5, 7×7, 9×9, and 11×11 FOVs. The 3×3 FOV emerges as the most cost-efficient option among all the FOVs examined in our study.

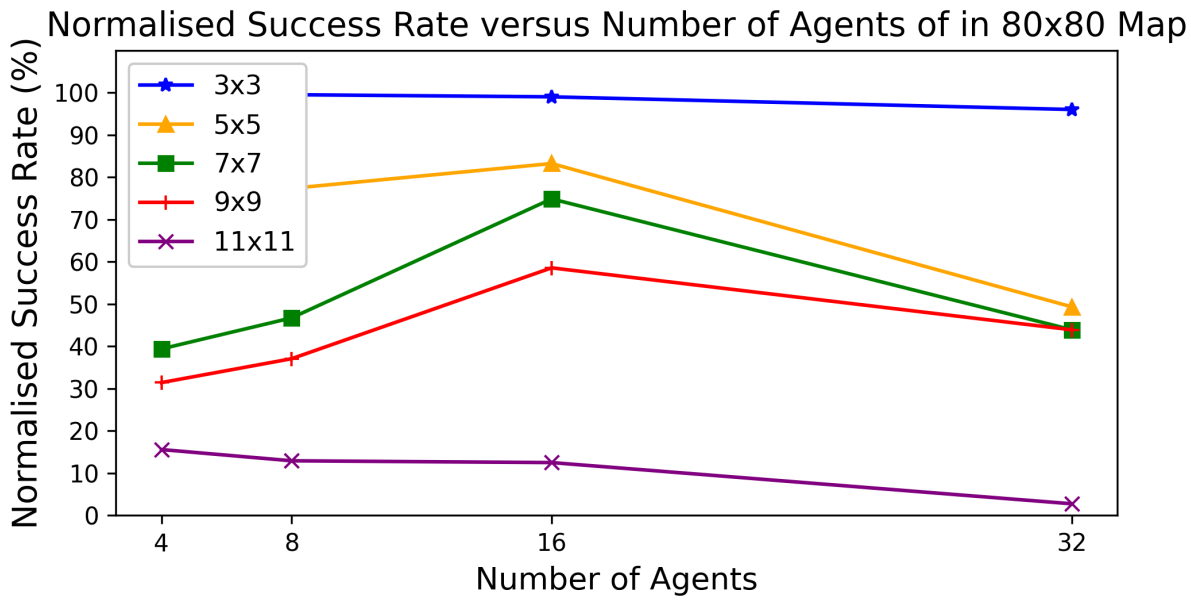

Figure 2.6: Normalized success rate with varying FOV settings in 80×80 Map

In this study, we have observed a consistent order between the network step time and normalized success rate for different FOV dimensions, specifically 3×3, 5×5, 7×7, 9×9, and 11×11. This finding highlights a strong correlation between the normalized success rate and the network step time. This correlation becomes even more pronounced when the range of success rates for different FOV sizes is relatively small (typically within 10%). At the same time, the maximum network step time can be three times longer than the minimum. Due to the reduced computation time associated with smaller FOV dimensions, these smaller dimensions will likely exhibit a comparatively higher normalized success rate.

2.6.4 Key Findings and Recommendations

- Enlarging the FOV size does not guarantee improved performance and it can potentially weaken performance. Conversely, smaller FOV sizes may prove to be more effective, as the decrease in performance is not proportional to the reduction in FOV size.
- Utilizing normalized success rates enables a comprehensive performance comparison across different FOV sizes. Conducting a thorough evaluation is crucial to identify the optimal FOV size for each specific application, as no size applies universally.

CHAPTER 3

LEVERAGING 360° CAMERAS IN 3D RECONSTRUCTION

3.1 Introduction

In recent years, the widespread adoption of 360° cameras in various industries, including construction and automotive, has provided an efficient method for capturing the complete surrounding environment. This paper introduces a groundbreaking vision-based pipeline for 3D reconstruction that leverages the capabilities of single 360° cameras while tackling the challenges associated with their usage. The decision to employ a vision-based approach instead of relying on inertial measurement units (IMU) or depth sensors like LiDAR is based on the accessibility and ease of use of video data alone, as opposed to requiring dedicated hardware setups and specialized software.

3D reconstruction plays a vital role in various fields, including robotics [7], Augmented Reality (AR) [5, 6], Building Information Modelling (BIM) [8, 9], and autonomous navigation [10, 11]. However, conventional 3D reconstruction methods often involve complex setups with multiple cameras or specialized hardware [12]. The emergence of consumer-grade 360° cameras has revolutionized the field by offering an affordable and user-friendly alternative. These cameras have gained significant popularity due to their ease of use, affordability, and ability to capture the entire surrounding environment in a single shot. As a result, they have become an ideal choice for a wide range of applications, democratizing access to 3D reconstruction technology and enabling its deployment in various industries and domains.

In order to address the calibration difficulties associated with 360° cameras, we propose a practical solution that eliminates the requirement for large checkerboard patterns. Furthermore, we introduce a technique to transform the distorted Equirectangular Projection (ERP), a commonly used image representation for 360° cameras, into four perspective views resembling cube maps. This conversion enables compatibility with deep learning models trained on undistorted perspective images, thereby expanding the potential applications of consumer-grade 360° cameras in the field of 3D reconstruction. By overcoming calibration challenges and providing a

standardized image representation, our approach offers a more accessible and efficient method for utilizing these cameras in various 3D reconstruction tasks.

Our approach to 3D reconstruction relies on Visual Simultaneous Localization and Mapping (VSLAM) techniques, which have gained prominence in the field. VSLAM enables real-time 3D mapping and camera pose tracking by leveraging the camera's visual input, making it well-suited for our research. Unlike traditional SLAM systems that depend on external infrastructure, our vision-based approach operates in a self-contained manner, relying solely on the camera's visual information. This simplicity and flexibility make it highly suitable for different scenarios. By combining the camera's pose estimation obtained from VSLAM with the cube map views, we can accurately determine the camera's position and orientation. This information is critical in generating detailed 3D mesh representations of indoor environments. Leveraging a recent 3D reconstruction method [38], our framework facilitates the creation of realistic 3D meshes based on the extracted camera poses and corresponding images from 360° cameras.

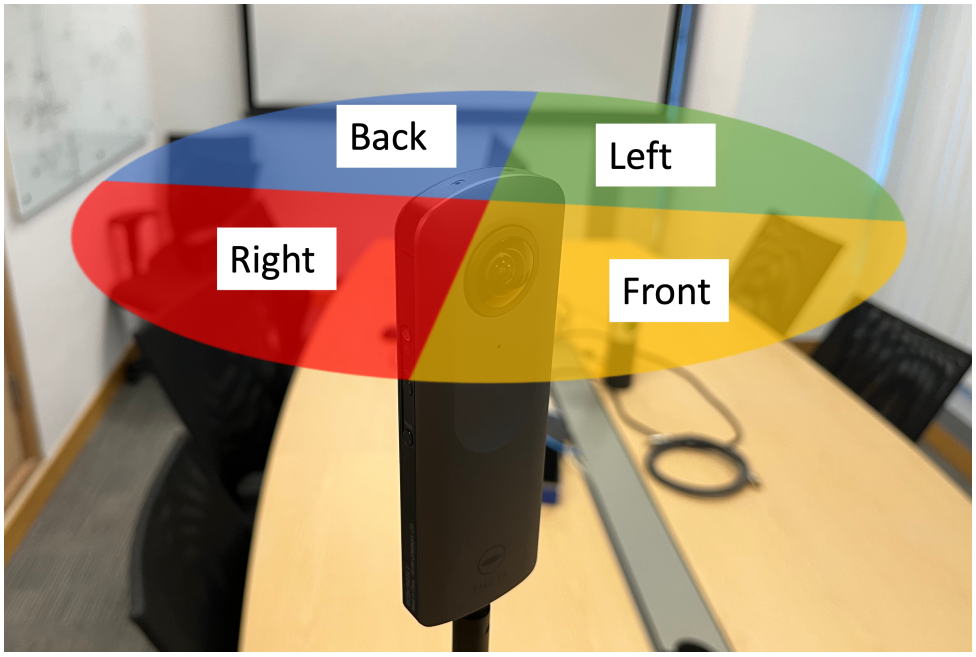

Figure 3.1: Visualizing the front, right, back, and left views after converting ERP into perspective images [2].

In order to assess the effectiveness of our approach, we conducted an experimental study to compare the performance of 360° cameras with traditional perspective cameras. By capturing videos of an indoor environment while walking a complete circle, we evaluated their ability to capture the scene's geometry. Furthermore, we compared the generated 3D meshes from each camera type with ground truth data obtained through a LiDAR scanner. This comparative analysis provides valuable insights into the accuracy and performance of our vision-based approach.

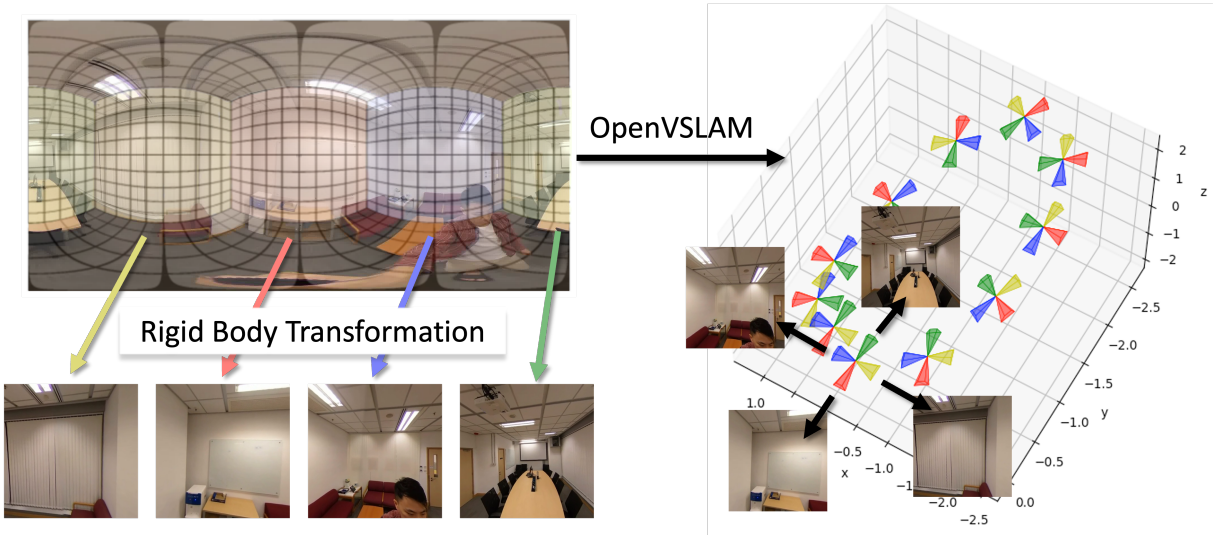

Figure 3.2: Overview of the processing pipeline in converting an ERP into perspective images and corresponding poses. An ERP is shown in the top left corner with a cube map projection overlaid on top, and the converted perspective images are shown in the bottom left. On the right side, the pose visualization graph illustrates the changes in the location and rotation of the poses for every 50 frames. Each color represents a particular view and its pose: yellow for the front, red for the right, blue for the back, and green for the left [2].

Our proposed vision-based approach, utilizing a single 360° camera, aims to provide a simplified and accessible solution for 3D reconstruction. By harnessing the advancements in consumer technology and capitalizing on the unique advantages of 360° cameras, we unlock their potential for achieving accessible 3D reconstructions. This breakthrough opens up new possibilities for various consumer-oriented applications across multiple fields, paving the way for exciting advancements in robotics, surveillance, and augmented reality.

3.2 Related Works

3.2.1 3D Reconstruction

Reconstructing a 3D model typically involves acquiring depth information from a sequence of images and integrating these depth maps. Traditional methods often rely on specialized hardware, such as LiDAR scanners or stereo cameras, to capture the environment's geometry. However, with recent advancements in consumer-grade cameras and computer vision techniques, more accessible and cost-effective approaches have emerged. These advancements have paved the way for new possibilities in 3D reconstruction, making it more attainable and affordable for a wider range of applications and users.

Structure from Motion (SfM) [39–41] and Multi-View Stereo (MVS) [42] are two commonly

used techniques for 3D reconstruction. SfM primarily relies on feature detection and matching algorithms to predict camera poses and reconstruct the 3D geometry of a scene. On the other hand, MVS focuses on reconstructing the 3D geometry from calibrated input images. In recent years, deep learning-based methods have emerged as a promising alternative for 3D reconstruction. These approaches have shown significant potential by utilizing Convolutional Neural Network (CNN) to learn feature representations from input images and estimate the scene's 3D geometry, especially in scenarios with limited data or challenging lighting conditions. Further advancements in deep learning have propelled the field of 3D reconstruction, enabling models to learn powerful feature representations directly from data [43–45] and improved the accuracy and effectiveness of 3D reconstruction processes.

Recent methods, such as Atlas [38], NeuralRecon [46], and CDRNet [47], employ neural networks to directly regress a truncated signed distance function (TSDF) volume for generating 3D models. Atlas, for instance, leverages extracted 3D features and feeds them to semantic heads for scene labeling. Labeling or semantics can enhance the quality of 3D reconstruction by incorporating knowledge about objects, textures, and scenes, thereby providing useful priors and constraints to generate more accurate models for applications like robotic navigation.

When working with fisheye or 360° cameras, the primary approach is to leverage the wide FOV of the camera to capture the environment from various viewpoints. One common approach involves using multiple cameras to capture the environment from different viewpoints and then merging the resulting images to create a consistent 3D model. However, this approach can be costly as it requires more hardware. Previous works, such as [48], have developed fisheye stereo matching algorithms. Another strategy involves employing a single 360° camera and leveraging its wide FOV to capture the environment from various viewpoints. This approach has been explored in several studies, with various techniques proposed to address the challenges associated with using 360° cameras for 3D reconstruction, including calibration and distortion correction. More recently, deep learning techniques have been employed for monocular depth estimation in the context of 360° cameras [49, 50]. These deep learning methods and 360° cameras have become a crucial foundation for current and future 3D reconstruction development. In particular, existing methods like MVSNet [44] have been adapted and applied to this domain, as seen in the case of 360MVSNet [51]. These advancements in leveraging deep learning have shown great potential in enhancing the accuracy and effectiveness of 3D reconstruction processes.

Overall, 3D reconstruction is a vibrant and active research area, with numerous approaches and techniques being explored. Using consumer-grade 360° cameras and deep learning-based methods has opened up new possibilities for more accessible and cost-effective 3D reconstruc-

tion. These advancements pave the way for exciting developments in augmented reality, virtual reality, and computer vision.

3.2.2 Visual-based Pose Estimation

VSLAM techniques, exemplified by [52–54], utilize image data to generate 3D representations of the environment and estimate camera poses. While these techniques can accommodate various camera configurations, such as monocular and stereo setups, they encounter difficulties in dynamic environments, especially when using monocular setups. By employing sensor fusion techniques, such as integrating IMU [55] and LiDAR sensors [56], algorithms can operate effectively in environments with limited visual information. However, setting up and calibrating these sensor fusion systems can be complex and challenging. As an alternative approach, widening the FOV of sensors can provide additional input data, which can enhance the algorithms' performance and accuracy.

Cameras with a wide FOV, such as fisheye and 360˚ cameras, can capture extensive environmental visual information, leading to more accurate pose estimation. Approaches like [57, 58] have successfully extended existing techniques [53, 59] to work effectively with 360˚ cameras. Notably, OpenVSLAM [3], which utilizes Oriented FAST and Rotated BRIEF (ORB), employs spatial feature matching for pose estimation with 360˚ cameras. It is also the first open-source VSLAM algorithm that supports equirectangular imagery. As 360˚ cameras and their supporting algorithms continue to develop, utilizing them has become a cost-efficient alternative to traditional perspective cameras, offering great potential for improved 3D reconstruction and pose estimation.

3.3 Proposed Framework

3.3.1 ERP Conversion

Established deep learning models, designed primarily for undistorted perspective images, are not well-suited for handling 360˚ images due to the inherent distortions they possess. Additionally, calibrating 360˚ cameras presents challenges and complexities, especially when using large checkerboard patterns. To overcome these obstacles, we propose a straightforward solution. Our approach involves converting ERP into four perspectives: front, back, left, and right. During the conversion process, the pixels from the surface of the 360˚ sphere are mapped onto a tangent plane, resulting in a more easily manageable perspective representation for subsequent

processing. These transformed views resemble cube maps and can be treated as outputs from four virtual cameras positioned in different directions.

We first establish a coordinate system for reference. A unit sphere is centered at the origin, and the image size of the output perspective image is denoted as (H, W). The viewing angles, both horizontal and vertical, originate from the sphere's center (Figure 3.3) and are defined to determine the size of a region of interest (ROI) on the sphere's surface. With the above information, we could find $H * W$ 3D coordinates of the ERP pixels on the ROI.

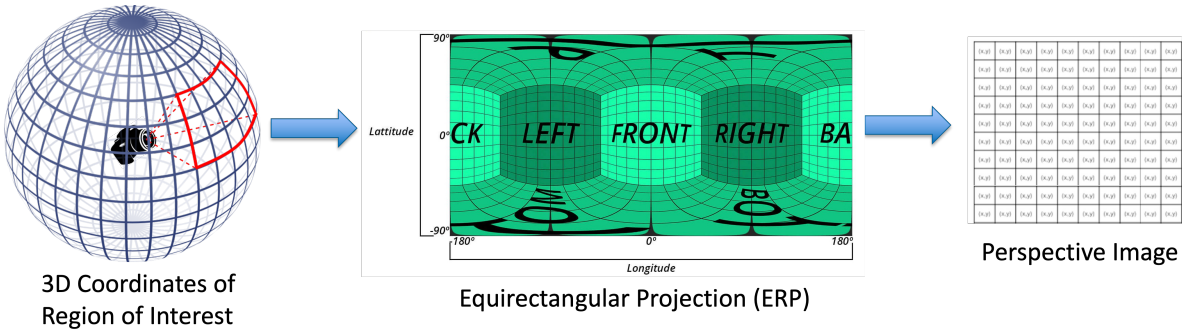

Figure 3.3: Overview of the ERP conversion process.

Next, to select a specific section on the sphere, we introduce an offset angle α for both horizontal and vertical directions. Each of the 3D coordinates of the ERP pixels can be considered a vector originating from the center of the sphere. Rodrigues' rotation formula (Formula 3.2) is applied to rotate these vectors along the horizontal and vertical axes based on the corresponding offset angles α.

The Rodrigues' rotation formula can be presented as

$$v_r = v + \sin\alpha(k \times v) + (1 - \cos\alpha)k \times (k \times v) \tag{3.1}$$

, where v represents the vector to be rotated, v_r is the resulting rotated vector, and k denotes the unit vector of the rotation axis. The formula can also be expressed in matrix form

$$R = I + (\sin\alpha)K + (1 - \cos\alpha)K^2 \tag{3.2}$$

, where I is the identity matrix, and R is the rotation matrix. Additionally, we have

$$v_r = Rv \tag{3.3}$$

, and the unit vector k can be represented as

$$K = \begin{bmatrix} 0, -k_z, k_y \\ k_z, 0, -k_x \\ -k_y, k_x, 0 \end{bmatrix}. \tag{3.4}$$

Here, k_x, k_y, k_z represent the unit vector k's components. While any unit vector can be used in principle, the ERP conversion pipeline only considers the unit vector along the y and z axes.

Following the rotation, the 3D coordinates of the ERP pixels are converted into latitude and longitude values using the following equations, with r representing the sphere's radius (assumed to be 1 in our case).

$$latitude = \sin^{-1} \frac{z}{r} \tag{3.5}$$

$$longitude = \tan \frac{y}{x} \tag{3.6}$$

Finally, the pixel coordinates on the ERP image are calculated using the obtained latitude and longitude values.

$$x_{ERP} = \frac{longitude}{180} \cdot x_{EPR_{center}} + x_{EPR_{center}} \tag{3.7}$$

$$y_{ERP} = \frac{latitude}{90} \cdot y_{EPR_{center}} + y_{EPR_{center}} \tag{3.8}$$

$(x_{EPR_{center}}, y_{EPR_{center}})$ are defined as the center coordinates of the ERP. At the end we would collect $H * W$ set of (x_{ERP}, y_{ERP}) pixel coordinate pairs. These coordinate values are then interpolated to integers before extracting the corresponding ERP pixel value and mapping it back to the perspective image.

In our experiment, we fixed both horizontal and vertical viewing angles that define the ROI on the sphere as 90˚. The horizontal α is defined as { 0˚, 90˚, 180˚, 270˚ } for the four views. The top and bottom views are excluded due to their high distortion and limitations inherent in the stitching process; thus, the vertical α is kept at 0˚.

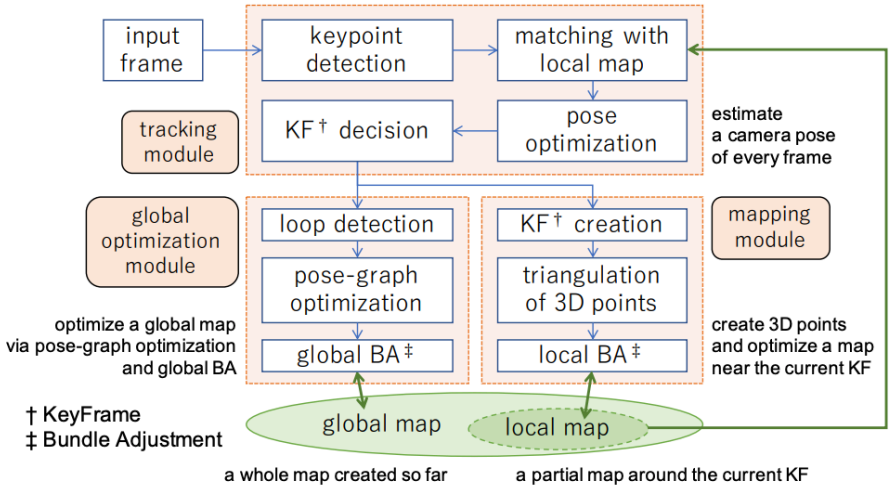

Figure 3.4: Overview of OpenVSLAM [3]

3.3.2 Pose Estimation

To determine the camera pose of ERP, we employ a VSLAM approach that depends exclusively on image input. In this study, we propose using OpenVSLAM, an ORB feature extractor-based VSLAM algorithm designed to be compatible with 360° cameras. The algorithm comprises three essential modules: tracking, mapping, and global optimization.

The tracking module estimates the camera pose for each frame by extracting features using the ORB feature extractor. It also determines if a frame qualifies as a keyframe for further processing. The keyframes are then passed to the mapping module, which utilizes them to triangulate 3D points, generating a comprehensive map of the environment's geometry. This reconstruction of spatial information is enabled by determining the 3D locations of landmarks from the 360° imagery. Finally, the global optimization module refines and optimizes the overall map through loop detection and global bundle adjustment. This ensures accurate and consistent camera poses and 3D points by minimizing errors across observations of scene elements from different viewpoints.

3.3.3 Pose Extraction of Cube Map Views

To calculate the poses of the four perspective views derived from ERP, we utilized the pose estimated by OpenVSLAM and applied a rigid body rotation for the corresponding views. This rotation incorporated four distinct rotation matrix sets mapping to each perspective view. As shown in Figure 3.2, the poses of the front, right, back, and left views are represented with yellow, red, blue, and green markers on the pose visualization graph.

The rigid body rotation is done by applying the 3×3 rotation matrix

$$R = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

to the rotation component $\bar{R}$ of a pose matrix P. The right, back, and left views are created through rotation θ of 90˚, 180˚, and 270˚ along the z-axis.

The pose matrix P, which represents a linear transformation from the origin and orientation of the current position, contains a 3x3 rotation matrix component $\bar{R}$ and a translation vector component t. The pose matrix is defined as:

$$P = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \bar{R} & t \\ 0 & 1 \end{bmatrix}.$$

Finally, the pose is updated as follows:

$$\bar{R}' = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$P' = \begin{bmatrix} R' & t \\ 0 & 1 \end{bmatrix}$$

3.3.4 3D Mesh Generation

By leveraging the perspective images obtained through ERP conversion, we can combine the extracted poses with their corresponding images. These combined posed images are then fed into our 3D reconstruction pipeline. To cater to applications such as BIM and VR solely on visual perception, we need a 3D reconstruction system that does not require depth inputs. To meet this requirement, we adopted Atlas [38], an end-to-end 3D reconstruction model that directly predicts TSDF from posed images or input RGB image sequences. In Atlas, the 2D CNN layers are first utilized to extract features from each image. Using the camera intrinsics and extrinsics, these 2D features are projected back and consolidated into a voxel volume. Subsequently, a 3D

CNN is employed to refine the aggregated voxel volume, predicting TSDF values and generating the final 3D model.

3.4 Experiment and Evaluation

To assess the effectiveness of utilizing 360° cameras, we conducted a series of experimental studies that involved comparing the performance of a 360° camera and a perspective camera, as well as evaluating the quality of the 3D mesh generated from the 360° camera under different conditions. For this experiment, we captured videos of an indoor environment using both camera types. The test environment was a conference room in our laboratory at HKUST, where a camera operator walked in a complete circle while the camera recorded the scene. We selected the popular Ricoh THETA V as our 360° camera and the iPhone 13 (with an FOV of approximately 72°) as our perspective camera. Each camera captured five videos, and we chose the videos that produced the best 3D meshes for the final comparison. The selected video from the 360° camera consisted of 484 ERP, while the perspective camera video contained 986 image frames.

3.4.1 Comparison between 360° Camera and Perspective Camera

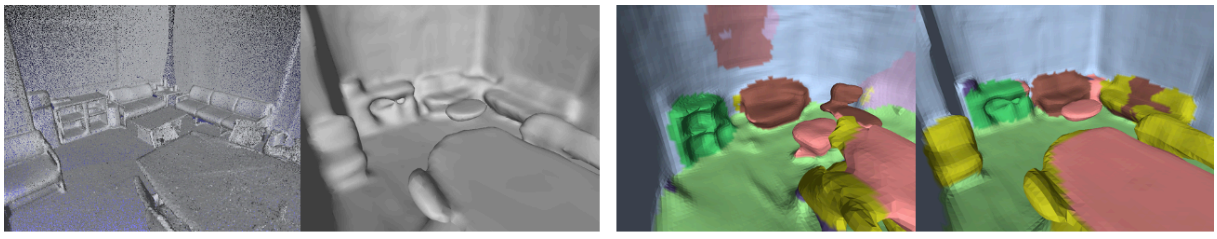

Figure 3.5: Qualitative 3D reconstruction results. (a) Ground truth LiDAR point cloud vs 3D model (without semantics) generated by 360° camera's data (b) 3D model (with semantics) generated by perspective camera's data vs 3D model (with semantics) generated by 360° camera's data [2].

In the initial experiment, we aimed to compare the 3D reconstruction performance between a 360° camera and a conventional perspective camera to determine which device produces higher-quality results. To achieve this, we first converted the captured panoramic videos from the 360° camera into perspective images. We then extracted the camera poses from the converted perspective images using OpenVSLAM (in equirectangular mode) and rigid body rotation. Similarly, we converted the captured video into image sequences for the perspective camera and extracted the camera poses using OpenVSLAM (in perspective mode). Both camera types' camera poses

and image sequences are combined as posed images. Subsequently, we employed the Atlas model to generate 3D meshes of the indoor environment based on the extracted camera poses and the corresponding images for both camera types.

To accurately assess the precision of the generated 3D models, we acquired ground truth data of the test environment using a LiDAR scanner. These ground truth data were presented as point clouds in Figure 3.5a. To evaluate the performances and compare the accuracy of the 3D meshes generated by each camera type, we calculated the F-score, representing the predictive performance by combining precision and recall. This is done by comparing the output 3D meshes against the ground truth data.

As depicted in Figure 3.6, the 3D mesh generated from the 360˚ camera exhibits better performance than that of the perspective camera, as indicated by its higher peak F-score of 0.297. This superiority can be attributed primarily to the panoramic image view provided by ERP. A qualitative comparison is presented in Figure 3.5. In our processing pipeline for the 360˚ camera, a single ERP image is converted into four perspective images. Therefore, given an equal number of raw images, the processing pipeline can provide four times the data compared to the perspective camera. This enhanced data collection efficiency of 360˚ cameras enables practical, real-world applications, as operators (human or robot) can reduce the time required to capture the scene and reconstruct an environment.

3.4.2 Quantifying Data Requirements for 3D Reconstruction using 360° Camera

In addition to assessing the optimal performance of the 360˚ camera in 3D reconstruction, we investigated the model's performance with varying amounts of data. To accomplish this, we systematically varied the quantity of data used for 3D reconstruction through random selection. Subsequently, we examined the minimum data required to achieve near-optimal performance, defined as an optimal F-score within a range of ±5%, for 3D reconstruction in each area. The results in Figure 3.6 indicate that approximately 400 perspective images or 100 raw ERP frames from the 360˚ camera are necessary to generate a near-optimal 3D mesh in our test environment. More specifically, in our experiment, using a 360° camera, capturing an average of 3.34 ERP frames per square meter provides sufficient image data for achieving near-optimal 3D reconstruction quality. This conclusion is based on our test environment size of $4.4m \times 6.8m = 29.92m^2$.

3.4.3 Impact of Camera Man Removal on 3D Mesh Quality

This experiment analyzes how removing the camera operator from 360° camera images affects 3D model quality. We implemented a simple approach where the camera operator's position was fixed on every i-th image on each horizontal cube map face, and every i-th image was filtered out from processing. Our results indicate that removing the camera operator can improve the quality of 3D models in certain scenarios. Specifically, we observed that the filtered version performed better with limited data as interference from the camera operator was eliminated. However, as more data became available, the non-filtered 3D mesh performed better, presumably due to information loss in the filtered version. The results are shown in Figure 3.6.

Overall, this experiment offers useful insights into using 360° cameras for 3D reconstruction. The results highlight how 3D reconstruction quality is impacted as the available frame count varies. Additionally, we demonstrate how factors like the presence of the camera operator affect the final 3D mesh. These findings can help inform best practices for deploying 360° cameras in applications requiring robust 3D modeling with limited image sets or for scenarios where equipment obstructs the view.

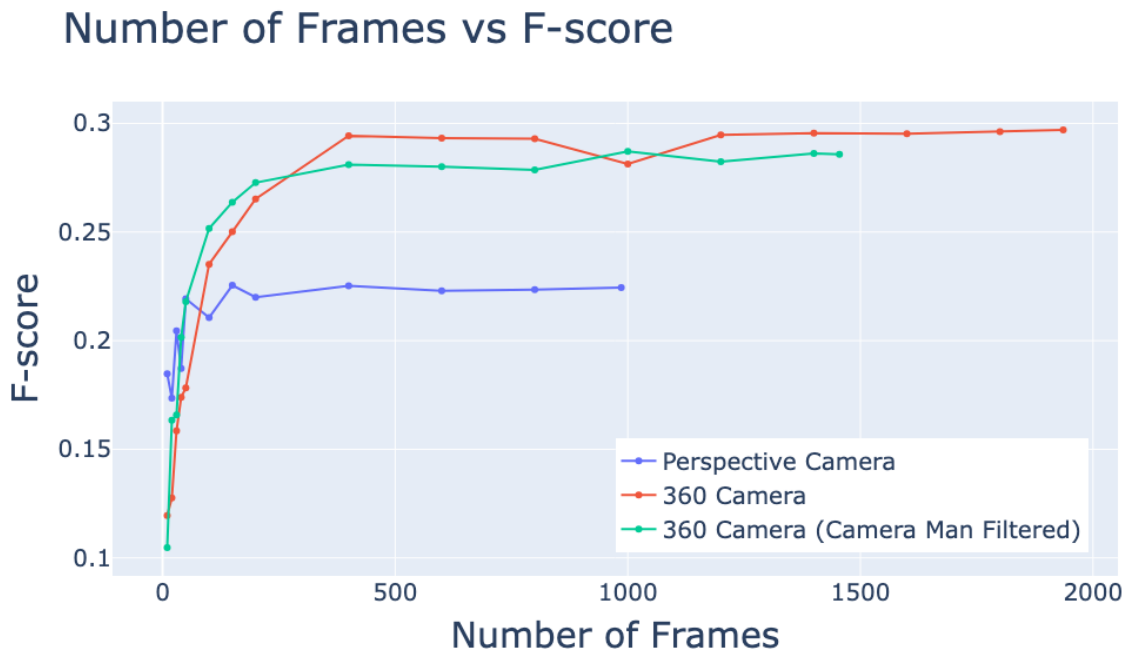

Figure 3.6: Comparing F-score between perspective camera, 360° camera, and 360° camera with camera man filtered. The F-score is evaluated for varying numbers of frames. [2].

3.5 Discussion

The outcomes of our study showcase the potential of utilizing 360° cameras to achieve high-quality 3D reconstructions, opening up new prospects for diverse applications across various domains. However, despite the promising results, certain limitations are still associated with our approach. Firstly, our method relies on converting ERP into cube maps, which could introduce artifacts and inaccuracies in the model inputs. Future investigations could explore alternative representations or develop models capable of directly handling ERP. Secondly, our approach is currently confined to indoor environments, and its outdoor or dynamic scenes performance requires further exploration. Lastly, our method does not explicitly address occlusions or reflections, which could impact the quality of the 3D reconstruction.

CHAPTER 4

DATA COLLECTION TOOL FOR 360° CAMERAS

4.1 Introduction

Our previous work explored monocular vision-based approaches for 3D reconstruction and pose estimation using 360° cameras. While such techniques can provide reasonable results, integrating additional sensor measurements, such as inertial measurements, can help improve accuracy. IMU sensors are commonly available on mobile devices and can enhance pose estimation robustness when combined with video frames through sensor fusion. Therefore, we wish to build a data collection tool for researchers to utilize the raw data on 360° cameras and explore its full potential.

Existing tools for collecting synchronized sensor data from mobile devices for visual-inertial research include Grafika [60], Mobile AR Sensor (MARS) Sensor Logger [61], and VideoIMUCapture [62]. Grafika is an SDK app developed by Google to exercise graphics and video capabilities, but it is not intended for stable data collection purposes. MARS Logger is built upon Grafika to record camera frames and IMU measurements from Android and iOS devices. Specifically, the camera frames from the MARS Logger are saved into H.264/MP4 videos using the OpenGL ES, Camera2 API, and MediaCodec. Furthermore, VideoIMUCapture extends the capability of MARS Logger by processing the frame metadata and IMU data in a Protobuf3 format for better efficiency. This provides researchers with temporally aligned visual and inertial recordings for tasks such as SLAM and 3D reconstruction. However, neither MARS Logger nor VideoIMUCapture supports 360° cameras, which are of strong interest for large-scale modeling and navigation applications. Our work aims to address this gap by developing a new data collection tool based on VideoIMUCapture, which has been modified to be compatible with 360° cameras. This will enable researchers to leverage the full sensory capabilities of 360° cameras coupled with IMU data through a stable, synchronized recording solution.

Additionally, combining stored video frames and IMU data enables sensor fusion-based pose estimation like visual-inertial odometry. This section describes the details of the implementation of retrieving and interpolating IMU data from the Android APIs. Experimental results demonstrate recording synchronized videos and IMU recordings from a Ricoh Theta 360° camera onto

a laptop. The collected data can be valuable for researchers working with 360° camera-based SLAM, 3D reconstruction, and other related tasks. All the codes are open-sourced and shared in [https://github.com/HKUST-ECE-IC-Design-Center-OWL/Theta-IMU]

4.2 Obtaining Video and IMU Data from 360° camera

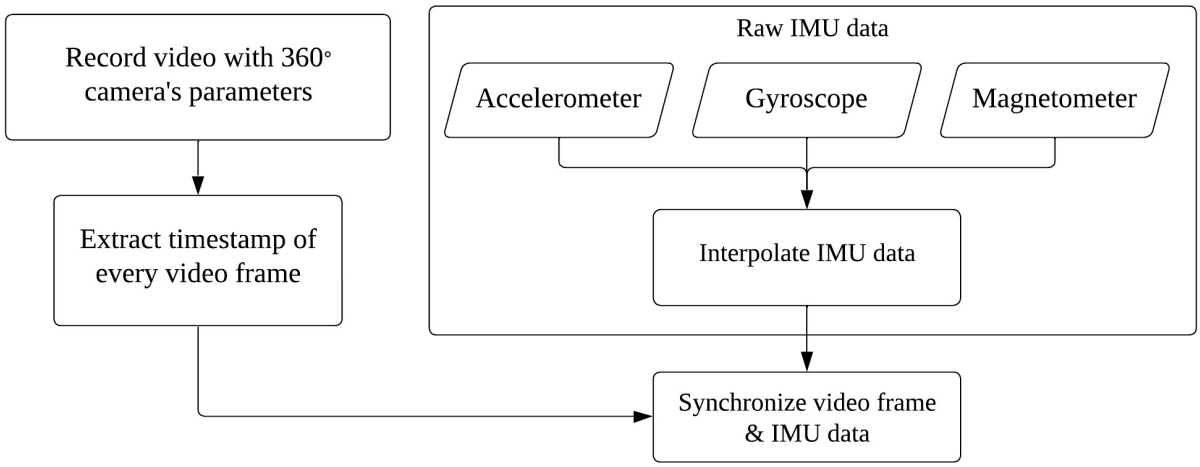

Figure 4.1: System diagram of Theta-IMU

The system for collecting synchronized video and IMU data from a 360° camera is shown in Figure 4.1. The Ricoh Theta V 360° camera, which natively runs the Android operating system, is used in our experiment. To support the Ricoh Theta V camera, we integrate the Ricoh Theta Plug-in Library [63] with our custom data collection application. We then install our plug-in application, Theta-IMU, on the 360° camera to retrieve raw sensor data. The application retrieves IMU measurements from the Android SensorManager at the highest available rate, typically around 100Hz. Since the sensors have different sampling rates and timestamps, the raw data is not perfectly synchronized between the three sensors. To address this, we apply a two-stage interpolation to synchronize the measurements. We first interpolate the accelerometer data and then the magnetometer data based on the gyroscope samples. This results in IMU readings from all three sensors having matched timestamps. This preprocessing enables further data fusion with the retrieved 360° video frames.

In addition to IMU data, our application captures video frames from the camera. However, Theta V only supports the older Camera API rather than the Camera2 API, so it does not provide reliable timestamps for each individual video frame. Support for Theta V was established using the Theta Plug-in Library, which allows us to handle callbacks for various camera operations. On the laptop, our collection application receives the data streams and saves the measurements to files with the data structure defined in Appendix B. This process allows reliably capturing and storing raw 360° visual and inertial recordings for various sensor fusion applications.

4.3 Experiment

Video and IMU data are generated only when the Theta V runs our Theta-IMU plug-in app. The Theta-IMU plug-in app can be activated through the plug-in management feature in the official Ricoh Theta desktop app or by directly launching the app via an emulator such as Vysor. However, it is important to note that normal USB transfer is not feasible for acquiring the data due to access restrictions imposed by the manufacturer. To circumvent this limitation and retrieve the data from the app, we rely on an additional plug-in app [4], as described in Figure 4.2, which facilitates wireless data transfer.

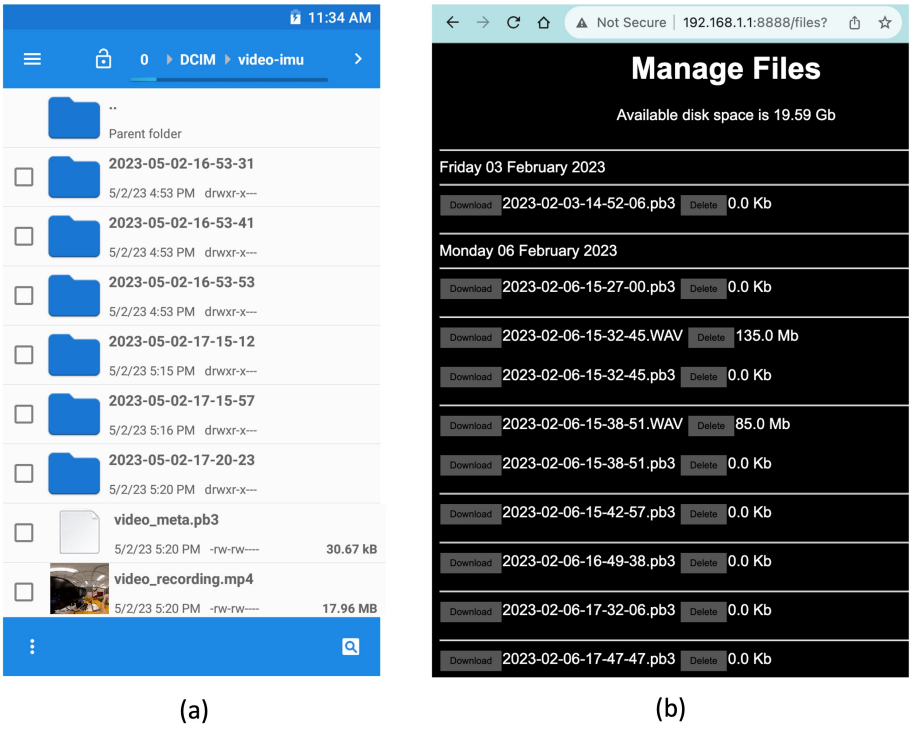

Figure 4.2: (a) Theta-IMU data snapshot (b) Data transfer panel on a computer [4]

While direct retrieval of synchronized video and IMU data through our app is not possible due to the absence of Camera2 API support, we have devised an alternative approach to address this limitation. By utilizing the OpenCV library [64], we extract the relative timestamps $\{t_1, t_2, ..., t_n\}$ for each video frame given the length of frames as n. Subsequently, we determine the initial timestamp t_s by analyzing the IMU data graph. To establish synchronization, we apply an offset to the video frame timestamps using the starting timestamps derived from the IMU data, i.e., $\{t_1 + t_s, t_2 + t_s, ..., t_n + t_s\}$

Figure 4.3 depicts the ERPs captured during the data collection. The ERP can be converted into perspective images to facilitate further processing using the methodology outlined in Section 3.3.1. Additionally, Figure 4.4 illustrates the collected IMU sensor data, which undergoes interpolation based on gyroscope samples. It is important to note that the IMU data guarantees a consistent starting timestamp, as samples collected when not all sensors are ready are discarded.

(a)

(b)

Figure 4.3: Examples of captured ERP. (a) is captured when the 360° camera is mounted on a robot. (b) is captured when the 360° camera is handheld

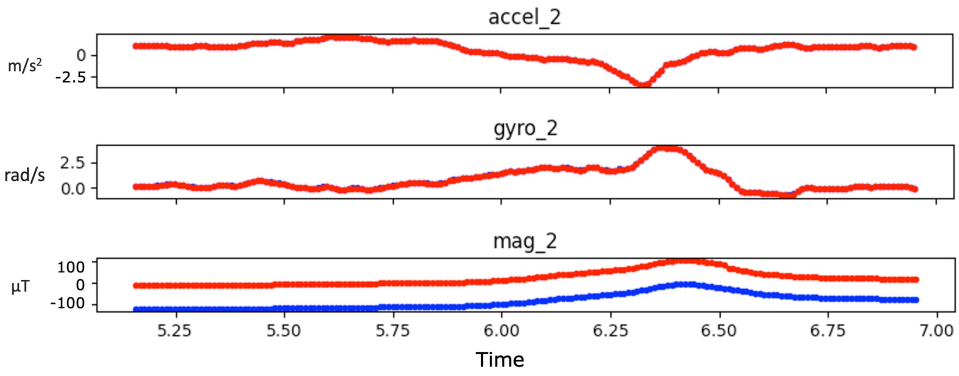

Figure 4.4: Captured accelerometer, gyroscope, and magnetometer data along z-axis

CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 Conclusion

This thesis explored key challenges in multi-agent 3D reconstruction of indoor environments and proposed novel solutions encompassing MAPF optimization, improved 360° camera-based 3D reconstruction techniques, and data processing. A decentralized RL framework is proposed to address communication overhead in the MAPF problem. It highlights the potential for computational cost reduction without compromising performance, offering valuable insights into designing efficient multi-agent systems. Additionally, a pipeline was introduced for accurate and data-efficient 3D reconstruction using 360° camera, leveraging its panoramic view to facilitate more efficient data collection and scene reconstruction. Integrating ERP conversion and VSLAM techniques establishes a vision-based approach for 3D reconstruction. Finally, a data collection tool is developed to acquire synchronized data from 360° cameras, enabling the construction of comprehensive training datasets. The resulting framework empowers scalable 3D reconstruction applications employing autonomous robot fleets across diverse domains. The future works are presented as follows.

5.2 Future Works

5.2.1 Communication-based MAPF

Building upon the foundation of this work on communication-based MAPF, several exciting avenues exist for future exploration. While the current work focuses on a simplified grid environment for initial validation, the next step is to evaluate the performance and scalability in more realistic settings. This involves utilizing established simulation platforms like Gazebo or Isaac Sim, which offer high-fidelity physics engines and comprehensive environment modeling capabilities. The evaluation will move beyond the simplified environment to assess performance in complex scenarios featuring dynamic obstacles and variable agent densities. Simultaneously, we will rigorously examine the algorithm's ability to navigate these situations safely, prioritizing collision avoidance and strict adherence to established safety protocols.

In addition, this work focuses on homogeneous agents with identical capabilities and communication protocols. Future development should encompass heterogeneous agents with diverse communica-

tion, sensing, and movement capabilities. This necessitates the development of adaptable communication strategies and potentially hierarchical decision-making structures for coordination.

5.2.2 3D Reconstruction with 360° Cameras

The proposed 3D reconstruction pipeline utilizing 360° cameras presents a promising approach for various applications. Building on the pipeline, we can explore exciting avenues to unlock its full potential. For instance, while the current work only utilizes 360° visual information, real-world scenarios demand a richer data pool to achieve optimal results with higher accuracy and usability of the reconstructed 3D models in domains such as construction. Potential data sources include LiDAR, which provides precise depth information. By fusing data from multiple sources, the reconstructed 3D models will be more robust to noise and occlusions, leading to a more complete representation of the real world.

In our current work, we only employ a specific 3D reconstruction algorithm. Future research can explore the potential of incorporating more advanced algorithms. Some promising candidates include Neural Radiance Fields (NeRF), 3D Gaussian Splatting, and 3D-aware diffusion models. By investigating and integrating these advanced algorithms, the 3D reconstruction pipeline can achieve even greater accuracy and detail in the reconstructed models.

Lastly, our current work only focuses on offline reconstruction of 3D scenes. However, achieving real-time reconstruction would offer more advantages. To enable this capability, future research will explore real-time reconstruction techniques, such as NeuralRecon and CDRNet. This shift towards real-time reconstruction would open doors to a wider range of applications. For example, augmented reality could seamlessly overlay virtual objects with semantic labeling onto the real world, and robots equipped with 360° cameras could leverage real-time reconstruction for building dynamic maps during autonomous navigation, facilitating safe and efficient movement. By achieving real-time reconstruction, the 360° camera-based pipeline can unlock a whole new realm of possibilities in various interactive and dynamic applications.

REFERENCES

[1] H. C. Cheng, L. Shi, and C. P. Yue, "Optimizing field-of-view for multi-agent path finding via reinforcement learning: A performance and communication overhead study," in *2023 62nd IEEE Conference on Decision and Control (CDC)*, 2023, pp. 2141–2146 (cit. on pp. viii, 12).

[2] H. C. Cheng, B. Hussain, Z. Hong, and C. P. Yue, "Leveraging 360° camera in 3d reconstruction: A vision-based approach," in *International Journal of Signal Processing Systems*, vol. 12, 2024, pp. 1–6 (cit. on pp. viii, 23, 24, 31, 33).

[3] S. Sumikura, M. Shibuya, and K. Sakurada, "Openvslam: A versatile visual slam framework," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 2292–2295 (cit. on pp. viii, 26, 29).

[4] K. Oerlemans, *Authydra*, https://github.com/ricohapi/theta-plugins/tree/main/plugins/com.kasper.authydra (cit. on pp. viii, 37).

[5] F. Gherardini, M. Santachiara, and F. Leali, "3d virtual reconstruction and augmented reality visualization of damaged stone sculptures," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 364, 2018, p. 012 018 (cit. on pp. 1, 22).

[6] S. González Izard, R. Sánchez Torres, O. Alonso Plaza, J. A. Juanes Mendez, and F. J. García-Peñalvo, "Nextmed: Automatic imaging segmentation, 3d reconstruction, and 3d model visualization platform using augmented and virtual reality," *Sensors*, vol. 20, no. 10, p. 2962, 2020 (cit. on pp. 1, 22).

[7] Y. Tao, M. Popović, Y. Wang, S. T. Digumarti, N. Chebrolu, and M. Fallon, "3d lidar reconstruction with probabilistic depth completion for robotic navigation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 5339–5346 (cit. on pp. 1, 22).

[8] B. Wang, Q. Wang, J. C. Cheng, C. Song, and C. Yin, "Vision-assisted bim reconstruction from 3d lidar point clouds for mep scenes," *Automation in Construction*, vol. 133, p. 103 997, 2022 (cit. on pp. 1, 22).

[9] J. Mahmud, T. Price, A. Bapat, and J.-M. Frahm, "Boundary-aware 3d building reconstruction from a single overhead image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 441–451 (cit. on pp. 1, 22).

[10] Y. Huang, W. Zheng, Y. Zhang, J. Zhou, and J. Lu, "Tri-perspective view for vision-based 3d semantic occupancy prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9223–9232 (cit. on pp. 1, 22).

[11] X. Yan *et al.*, "Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 3101–3109 (cit. on pp. 1, 22).

[12] R. Ren, H. Fu, H. Xue, Z. Sun, K. Ding, and P. Wang, "Towards a fully automated 3d reconstruction system based on lidar and gnss in challenging scenarios," *Remote Sensing*, vol. 13, no. 10, p. 1981, 2021 (cit. on pp. 2, 22).

[13] R. Stern *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, 2019, pp. 151–158 (cit. on pp. 7, 10, 11).

[14] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015 (cit. on pp. 7, 9).

[15] V. Rybár and P. Surynek, "Highways in warehouse multi-agent path finding: A case study.," in *ICAART (1)*, 2022, pp. 274–281 (cit. on pp. 7, 9).

[16] P. Surynek, A. Felner, R. Stern, and E. Boyarski, "Efficient sat approach to multi-agent path finding under the sum of costs objective," in *Proceedings of the twenty-second european conference on artificial intelligence*, 2016, pp. 810–818 (cit. on p. 7).

[17] G. Sartoretti *et al.*, "Primal: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2378–2385, 2019 (cit. on pp. 7, 10, 11).

[18] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti, "Primal _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2666–2673, 2021 (cit. on p. 7).

[19] Z. Liu, B. Chen, H. Zhou, G. Koushik, M. Hebert, and D. Zhao, "Mapper: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 11 748–11 754 (cit. on pp. 7, 10, 12).

[20] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 11 785–11 792 (cit. on pp. 7, 10).

[21] Z. Ma, Y. Luo, and H. Ma, "Distributed heuristic multi-agent path finding with communication," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 8699–8705 (cit. on pp. 7, 10–13).

[22] Q. Li, W. Lin, Z. Liu, and A. Prorok, "Message-aware graph attention networks for large-scale multi-robot path planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5533–5540, 2021 (cit. on p. 7).

[23] Z. Ding, T. Huang, and Z. Lu, "Learning individually inferred communication for multi-agent cooperation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 069–22 079, 2020 (cit. on pp. 7, 12).

[24] A. Das *et al.*, "Tarmac: Targeted multi-agent communication," in *International Conference on Machine Learning*, PMLR, 2019, pp. 1538–1546 (cit. on pp. 7, 10).

[25] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 29, 2016 (cit. on p. 7).

[26] S. Q. Zhang, Q. Zhang, and J. Lin, "Efficient communication in multi-agent reinforcement learning via variance based control," *Advances in Neural Information Processing Systems*, vol. 32, 2019 (cit. on p. 7).

[27] Z. Ma, Y. Luo, and J. Pan, "Learning selective communication for multi-agent path finding," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1455–1462, 2021 (cit. on pp. 7, 10–12).

[28] S. LaValle, "Planning algorithms," *Cambridge University Press google schola*, vol. 2, pp. 3671–3678, 2006 (cit. on p. 9).

[29] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research: The 14th International Symposium ISRR*, Springer, 2011, pp. 3–19 (cit. on p. 9).

[30] G. Sanchez and J.-C. Latombe, "Using a prm planner to compare centralized and decoupled planning for multi-robot systems," in *Proceedings 2002 IEEE international conference on robotics and automation (Cat. No. 02CH37292)*, IEEE, vol. 2, 2002, pp. 2112–2119 (cit. on p. 9).

[31] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 5, 2014, pp. 19–27 (cit. on p. 9).

[32] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 3052–3059 (cit. on p. 10).

[33] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 1343–1350 (cit. on p. 10).

[34] C. Ferner, G. Wagner, and H. Choset, "Odrm* optimal multirobot path planning in low dimensional search spaces," in *2013 IEEE international conference on robotics and automation*, IEEE, 2013, pp. 3854–3859 (cit. on p. 10).

[35] B. Riviere, W. Hönig, Y. Yue, and S.-J. Chung, "Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning," *IEEE robotics and automation letters*, vol. 5, no. 3, pp. 4249–4256, 2020 (cit. on p. 10).

[36] W. Hönig, J. A. Preiss, T. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018 (cit. on p. 10).

[37] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*, PMLR, 2016, pp. 1995–2003 (cit. on p. 13).

[38] Z. Murez, T. Van As, J. Bartolozzi, A. Sinha, V. Badrinarayanan, and A. Rabinovich, "Atlas: End-to-end 3d scene reconstruction from posed images," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, Springer, 2020, pp. 414–431 (cit. on pp. 23, 25, 30).

[39] S. Agarwal *et al.*, "Building rome in a day," *Communications of the ACM*, vol. 54, no. 10, pp. 105–112, 2011 (cit. on p. 24).

[40] J.-M. Frahm *et al.*, "Building rome on a cloudless day," in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*, Springer, 2010, pp. 368–381 (cit. on p. 24).

[41] S. B. Kang and R. Szeliski, "3-d scene data recovery using omnidirectional multibaseline stereo," *International journal of computer vision*, vol. 25, pp. 167–183, 1997 (cit. on p. 24).

[42] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, "Pixelwise view selection for unstructured multi-view stereo," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, Springer, 2016, pp. 501–518 (cit. on p. 24).

[43] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang, "Deepmvs: Learning multi-view stereopsis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2821–2830 (cit. on p. 25).

[44] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "Mvsnet: Depth inference for unstructured multi-view stereo," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 767–783 (cit. on p. 25).

[45] R. Chen, S. Han, J. Xu, and H. Su, "Point-based multi-view stereo network," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1538–1547 (cit. on p. 25).

[46] J. Sun, Y. Xie, L. Chen, X. Zhou, and H. Bao, "Neuralrecon: Real-time coherent 3d reconstruction from monocular video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 598–15 607 (cit. on p. 25).

[47] Z. Hong and C. P. Yue, "Cross-dimensional refined learning for real-time 3d visual perception from monocular video," *arXiv preprint arXiv:2303.09248*, 2023 (cit. on p. 25).

[48] C. Häne, L. Heng, G. H. Lee, A. Sizov, and M. Pollefeys, "Real-time direct dense matching on fisheye images using plane-sweeping stereo," in *2014 2nd International Conference on 3D Vision*, IEEE, vol. 1, 2014, pp. 57–64 (cit. on p. 25).

[49] N.-H. Wang, B. Solarte, Y.-H. Tsai, W.-C. Chiu, and M. Sun, "360sd-net: 360 stereo depth estimation with learnable cost volume," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 582–588 (cit. on p. 25).

[50] F.-E. Wang, Y.-H. Yeh, M. Sun, W.-C. Chiu, and Y.-H. Tsai, "Bifuse: Monocular 360 depth estimation via bi-projection fusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 462–471 (cit. on p. 25).

[51] C.-Y. Chiu, Y.-T. Wu, I. Shen, Y.-Y. Chuang, *et al.*, "360mvsnet: Deep multi-view stereo network with 360deg images for indoor scene reconstruction," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 3057–3066 (cit. on p. 25).

[52] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2014, pp. 15–22 (cit. on p. 26).

[53] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*, Springer, 2014, pp. 834–849 (cit. on p. 26).

[54] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017 (cit. on p. 26).

[55] T. Qin, S. Cao, J. Pan, and S. Shen, "A general optimization-based framework for global pose estimation with multiple sensors," *arXiv preprint arXiv:1901.03642*, 2019 (cit. on p. 26).

[56] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 2020, pp. 5135–5142 (cit. on p. 26).

[57] D. Caruso, J. Engel, and D. Cremers, "Large-scale direct slam for omnidirectional cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 141–148 (cit. on p. 26).

[58] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021 (cit. on p. 26).

[59] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017 (cit. on p. 26).

[60] Google, *Grafika*, https://github.com/google/grafika (cit. on p. 35).

[61] J. Huai, Y. Zhang, and A. Yilmaz, "The mobile ar sensor logger for android and ios devices," in *2019 IEEE SENSORS*, 2019, pp. 1–4 (cit. on p. 35).

[62] D. Gillsjö, *VideoIMUCapture-Android*, https://github.com/DavidGillsjo/VideoIMUCapture-Android, 2022 (cit. on pp. 35, 50).

[63] Ricoh, *theta-plugin-library*, https://github.com/ricohapi/theta-plugin-library/tree/master (cit. on p. 36).

[64] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000 (cit. on p. 37).

APPENDIX A

MAPF PERFORMANCE OF OUR RL MODEL IN 40 × 40 MAP

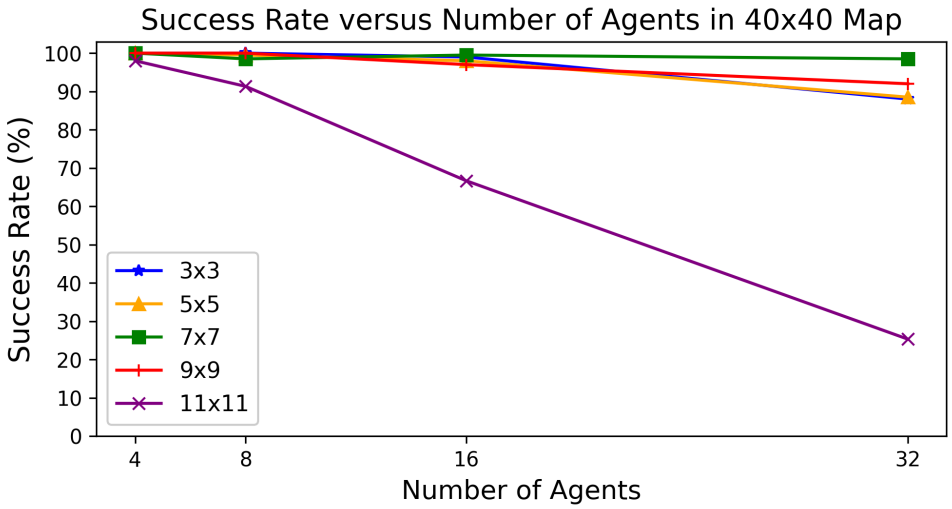

Figure A.1: Success rate with varying FOV settings in 40 × 40 Map

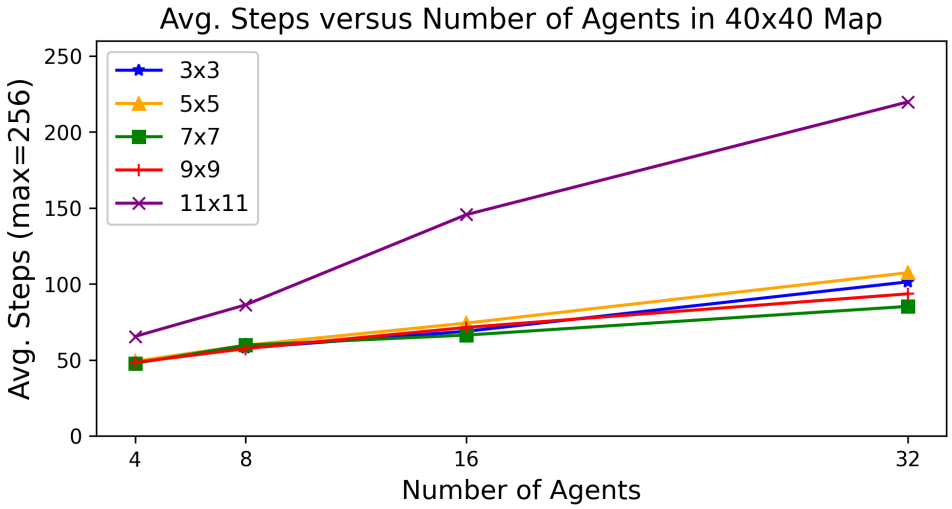

Figure A.2: Average steps with varying FOV settings in 40 × 40 Map

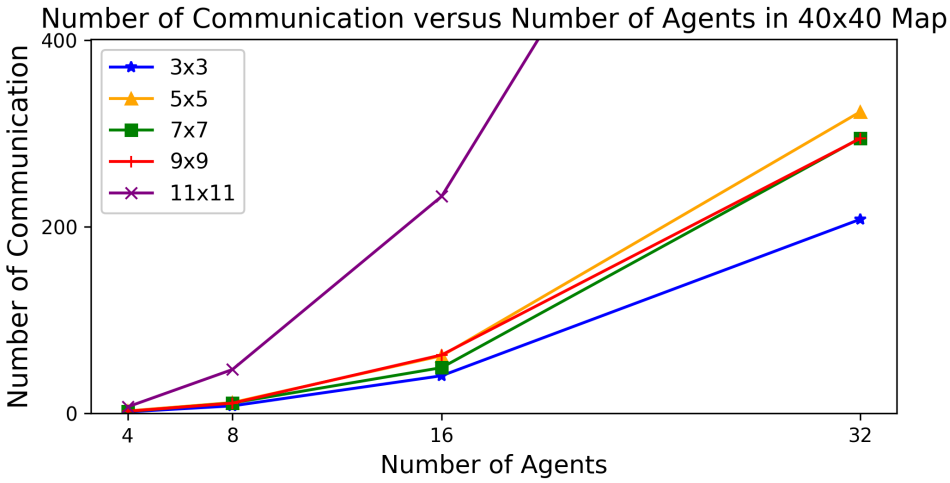

Figure A.3: Number of communication with varying FOV settings in 40×40 Map

APPENDIX B

DATA STRUCTURE OF 360° CAMERAS' DATA COLLECTION TOOL

The below data structure for Ricoh Theta V is developed based on [62].

```
syntax = "proto3";

package thetaimu;

message IMUInfo {
  string gyro_info = 1;
  float gyro_resolution = 2;
  string accel_info = 3;
  float accel_resolution = 4;
  string mag_info = 5;
  float mag_resolution = 6;
  float sample_frequency = 7; //Hz
  repeated float placement = 8;
}

message IMUData {
  int64 time_ns = 1;
  repeated float gyro = 2;
  repeated float gyro_drift = 3;
  repeated float accel = 4;
  repeated float accel_bias = 5;
  repeated float mag = 6;
  repeated float mag_bias = 7;
  enum Accuracy {
    UNRELIABLE=0;
    LOW = 1;
    MEDIUM = 2;
    HIGH = 3;
  }
  Accuracy gyro_accuracy = 8;
  Accuracy accel_accuracy = 9;
```

```
  Accuracy mag_accuracy = 10;
}

message VideoFrameMetaData {
  int64 time_ns = 1;
  int64 frame_number = 2;
  int64 exposure_time_ns = 3;
  int64 frame_duration_ns = 4;
  int64 frame_readout_ns = 5;
  int32 iso = 6;
  float focal_length_mm = 7;
  float est_focal_length_pix = 8;
  float focus_distance_diopters = 9;

  message OISSample {
    int64 time_ns = 1;
    float x_shift = 2;
    float y_shift = 3;
  }
  repeated OISSample OIS_samples =10;
  bool focus_locked = 11;
}

message CameraInfo {
  //fx, fy, cx, cy, s
  // for details on how to use the intrinsics, pose_translation ↩
   and pose_rotation.
  repeated float intrinsic_params = 1;
  //Radial: k1,k2,k3, Tangential: k4,k5
  repeated float distortion_params = 2;
  bool optical_image_stabilization = 3;
  bool video_stabilization = 4;
  bool distortion_correction = 10;
  int32 sensor_orientation = 14;

  enum FocusCalibration {
    UNCALIBRATED = 0;
    APPROXIMATE = 1;
    CALIBRATED = 2;
  }
  FocusCalibration focus_calibration = 5;
```

```
72
73    enum TimestampSource {
74      UNKNOWN = 0;
75      REALTIME = 1;
76    }
77    TimestampSource timestamp_source = 6;
78
79    enum LensPoseReference {
80      PRIMARY_CAMERA = 0;
81      GYROSCOPE = 1;
82      UNDEFINED = 2;
83    }
84    LensPoseReference lens_pose_reference = 7;
85    repeated float lens_pose_rotation = 8;
86    repeated float lens_pose_translation = 9;
87
88    message Size {
89      int32 width = 1;
90      int32 height = 2;
91    }
92    Size resolution = 11;
93    Size pre_correction_active_array_size = 12; //↩
       SENSOR_INFO_PRE_CORRECTION_ACTIVE_ARRAY_SIZE
94    repeated float original_intrinsic_params = 13;
95  }
96
97  message VideoCaptureData {
98    google.protobuf.Timestamp time = 1;
99    CameraInfo camera_meta = 2;
100   IMUInfo imu_meta = 3;
101
102   repeated IMUData imu = 4;
103   repeated VideoFrameMetaData video_meta = 5;
104 }
```