# Real-Time Visual Localization System in Changing and Challenging Environments via Visual Place Recognition

Ruohai Ge

CMU-RI-TR-22-31
July 2022

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

*To my grandfather who never saw this adventure*

# Abstract

Localization is one of the fundamental capabilities to guarantee reliable robot autonomy. Many excellent Visual-Inertial and LiDAR-based algorithms have been developed to solve the localization problem. However, deploying these methods on a real-time portable device is challenging due to high computing power and high-cost sensors (LiDAR). Another option would be Visual Place Recognition(VPR). Compared to Visual-Inertial and LiDAR-based algorithms, VPR only needs to establish relationships between observed and visited places, reducing the computing power. And it mainly uses the camera as its input sensor, which is much cheaper than LiDAR. However, most current VPR-related works do not provide an efficient and robust localization pipeline. Only a few work on VPR-based methods in real-time with portable devices in non-challenging environments. To cover the gap in the VPR field, we present a real-time VPR-based localization system in changing and challenging environments.

This pipeline mainly utilizes the concept of images sequences matching. We also choose to use the omnidirectional camera for visual inputs. Compared to a pinhole camera or a fisheye camera, an omnidirectional camera provides a broader field of view, thus increasing the performance and robustness of our proposed localization pipeline. Our pipeline can also detect an agent's off-path status and re-localize the agent once he/she is back on the path. We build a mapping rover robot with an omnidirectional camera, LiDAR, Inertial Measurement Unit(IMU), and a localization helmet with an omnidirectional camera and IMU as the portable device. Furthermore, we gather a campus-scale dataset with omnidirectional visual inputs and LiDAR inputs on ten different trajectories for eight repeated times under different illuminations and viewpoints.

We show our pipeline's quantitative performance by evaluating our collected dataset against the ground truth. We then show the qualitative performance by doing real-time demo with our robot at Carnegie Mellon University. To show the robustness and generality of our pipeline, we tested it in different changing and challenging environments, including outdoor, long, and narrow indoor corridors and narrow and dark indoor confined spaces.

# Acknowledgments

First and foremost, I am incredibly grateful to my supervisor, Prof. Sebastian Scherer, for his invaluable advice, continuous and timely support, and patience during my Master's study. His immense knowledge and hands-on experience have encouraged me in my academic research. I am also grateful for the feedback and advice from my thesis committee: Professor Kris Kitani and Cherie Ho.

This thesis would not have been possible without Dr. Peng Yin, whose guidance from the initial step of research enabled me to develop an understanding of the field. I am thankful for the extraordinary experiences he arranged for me. It is an honor to learn from him. His research vision, diligence, execution ability, and problem-solving techniques deeply inspired me.

Also, I would like to give my special thanks to Zheng Xu and Ruijie Fu, without whom this thesis would not have been possible. In addition, I would like to thank my lab mates, especially William Pridgen, Joshua Spisak, Yaoyu Hu, Shibo Zhao, Ivan Cisneros, and everyone I worked with. I also owe thanks to my friends who helped me many times during my Master's studies: Zhihao Zhang, Yifan Song, and Ruoyang Xu.

Lastly, I would like to thank Fengyi Zhang and my parents for their unconditional love and support throughout my life

## Funding

This work is in part supported by ARL Human Augmentation IDIQ.

# Contents

*When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.*

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background and Motivation

Robots have played essential roles in worldwide pandemics, space exploration, and our daily life in the current age, where localization is one of the fundamental capabilities to guarantee reliable robot autonomy. As mentioned in [38], during the battle with COVID-19, mobile robots can provide contacting-free delivery for food, medicine, and contaminated waste, as well as give remote access to front-line healthcare. In the post-pandemic world, robotics delivery and autonomous driving [34] are changing traditional transportation and our living habit. Thus, a robust, cheap, and efficient localization system is necessary since it makes these new technologies affordable and feasible for everyone in society.

In recent years, researchers have developed many excellent systems to solve the localization problem. Visual-Inertial and LiDAR based SLAM systems are two popular directions. However, localization systems based on these methods are either inefficient or expensive. Another popular direction is Visual Place Recognition(VPR). Compared to Visual-Inertial and LiDAR based SLAM systems, VPR only establishes relationships between observed and visited places and uses the camera instead of the LiDAR as its primary sensor, which makes it efficient and cheap. However, most current VPR-related works do not provide an efficient and robust localization pipeline. Only a few work on VPR-based methods in real-time with portable devices in non-challenging environments.

To cover the gap in the VPR field, we present a real-time VPR-based localization system. This system works efficiently and robustly in changing and challenging environments, as shown in figure 1.1. We integrate and improve different place match techniques and use existing place recognition techniques and mapping frameworks. This pipeline consists of four main parts: global localization, local localization, odometry fusion, and off-path detection. The global localization module can quickly localize the agent in the given map, and the local localization module can provide further continuous position tracking. The odometry fusion provides high-frequency (10HZ) relative odometry information. Furthermore, the agent might not always be on the given map. This system can detect the agent's off-path status and re-localize the agent once he/she is back on the path again.

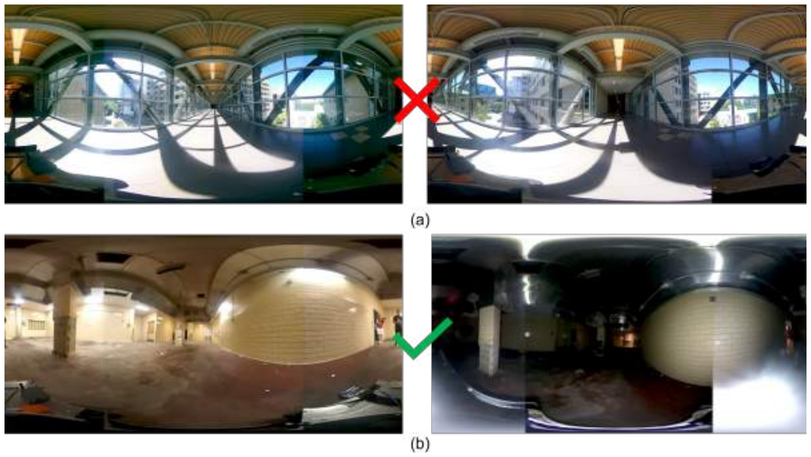

Figure 1.1: It is not easy for a VPR system to work robustly in changing and challenging environments. It needs to successfully distinguish (a), which looks very similar. And it also needs to match (b), which looks very different due to illumination and viewpoint changes.

We also build a mapping robot and a localization helmet to execute the real-time localization system. The robot and the helmet are equipped with omnidirectional cameras. An omnidirectional camera provides a wider field of view than a pinhole or a fisheye camera, leading to more robust localization. Furthermore, we collect a dataset with omnidirectional visual inputs and ground truth data to evaluate our proposed method. This dataset can also be used to assess existing or future VPR-related works.

We perform quantitative and qualitative evaluations on our system using the collected datasets. Our global localization module can achieve 0.925 Area Under Precision-Recall Curve(AUC-PR), and the localization time is around 20 seconds on a 400-meters map. And our local localization module can output the current location's estimation at 3HZ regardless of the size of the map. The local localization module reaches 0.922 AUC-PR. To perform off-path detection, both global and local localization modules must be able to differentiate between visited and new places. Global and Localization modules achieve 0.93 and 0.96 Area Under Receiver-Operating -Characteristic Curve(AUC-ROC) respectively. The details of these metrics are explained in section 7.1. Qualitative experiments are also performed at Carnegie Mellon University with the self-built mapping robot and localization helmet.

## 1.2 Contributions

Our contributions are summarized as follows:

1. We develop a real-time localization system in changing and challenging environments. This system runs on a helmet equipped with an omnidirectional camera. It is robust, efficient, and can detect off-path and re-localize the agent when he/she is back on the path again;

2. We gather a campus-scale dataset with ten different trajectories for eight repeated times under different illuminations and viewpoints. Each sequence has omnidirectional visual inputs, LiDAR inputs, and ground truth data;

3. We build a mapping rover robot with an omnidirectional camera, LiDAR, and Inertial Measurement Unit(IMU) and a localization helmet with an omnidirectional camera and IMU.

# Chapter 2

# Related Work

This chapter's structure works as follows: Section 2.1 explains the reason for using VPR-based methods to complete the proposed localization pipeline. Section 2.2 talks about VPR techniques developed for changing and challenging environments. Section 2.3 introduces different VPR's place match techniques, forming our pipeline's main parts. And section 2.4 discusses existing VPR datasets and the reason for us to collect our own dataset.

## 2.1 Why Visual Place Recognition

Many LiDAR-based SLAM algorithms [44], [9], [45] have shown excellent performances in localization. However, LiDAR is a costly sensor compared to a camera or an Inertial Measurement Unit(IMU). Making all agents use it for localization on a pre-generated high-quality map will be unfeasible and unnecessary. Despite the LiDAR-based SLAM algorithm, there are many brilliant Visual-Inertial SLAM systems, such as ORB-SLAM3 [5], VINS-Mono [29]. Nevertheless, ORB-SLAM3's graph optimization part consumes too much computation power, and VINS-Mono may become unstable given challenging localization environments or unsteady motions. Thus, Visual-Inertial SLAM is unsuitable for real-time localization with portable devices in challenging and changing environments. Another probability in localization is Visual Place Recognition (VPR). Compared to Visual-Inertial and LiDAR based SLAM systems, VPR only establishes relationships between observed and visited places and uses the camera instead of the LiDAR as its primary sensor, which makes it efficient and cheap. According to [20], early VPR works often assume the environment stays relatively similar, making VPR unsuitable for real-world localization. Consequently, researchers have been focused on creating VPR algorithms adaptive to changing and challenging environments, resulting in VPR's possibility to do real-world localization.

## 2.2 Visual Place Recognition in Changing Environments

As discussed in section 2.1, many researchers are developing VPR algorithms to tackle the problem of localization in challenging and changing environments. Their works can roughly be categorized into the following aspects: place representation techniques, mapping frameworks, and place match techniques. Since our pipeline mainly integrates and improves place match techniques, section 2.3 will introduce more details about this category. This section focuses on the other two aspects. A good place representation technique is the key to a robust VPR-based localization system. The appearance of a place can change drastically due to illumination, viewpoint, and dynamic objects. As shown in figure 1.1, an outstanding place representation technique should be able to differentiate similarly but not same places and relate the same place with vast appearance changes. Traditional (Non-learning) techniques such as CoHoG [41], HOG [11], FAB-MAP [10] and Learning-based techniques such as NetVLAD [1], RegionVLAD [16], HybridNet [8], OverlapNet [7] have all shown reasonable place representation performances in changing and challenging environments. Most of these methods will be later evaluated in chapter 8 to show our proposed pipeline's generality and compatibility.

The most simplified mapping framework is just pure images without any position information. We can have a topological map with images by adding the relative positions between images. Early VPR works FAB-MAP [10] and SeqSLAM[24] use topological maps as their mapping frameworks. Then, with their corresponding improved versions CAT-SLAM[21] and SMART [28], it is proved that adding odometry information to the topological mapping frameworks can improve VPR's performances. To deal with changing environments, some come up with the idea of using a dynamic map which can remember and forget features. [4] forgets older features at different rates depending on the timescale. [12], [25] use selective attention mechanisms and rehearsal mechanisms to remember most occurring features and forget transient features. Due to real-time requirements and portable devices' limited computation powers, our proposed localization pipeline uses a topological map with visual inputs and odometry information.

Besides the mentioned three aspects, only a few tackle the problem of developing an efficient and robust localization pipeline on portable devices in changing environments. [26] presents a real-time localization system on a mobile device, yielding good results in indoor environments. However, their pipeline needs a pre-training phase and ignores illumination changes to satisfy the real-time constraint.

## 2.3 Place Match Techniques

Section 2.2 talks about two major categories of VPR research. This section focuses on the remaining category: place match techniques, which forms the main parts of our pipeline. The most straightforward place match technique uses an individual query image to compare with the reference images. This place match concept is fast and accurate in a non-changing environment with robust place representation techniques. However, real-world environments are full of different places with similar features and the exact places with different features due to spatial/dynamic scenery differences, creating many noises for single image matching. Thus, recent approaches tend to use images sequences comparisons instead of single image comparisons.

Using image sequences instead of single images exploits the assumption that the VPR system is traversing a very similar path through the environment instead of only a specific place, thus improving the tolerance to local scene changes and reducing false positives with self-similarities environments. Seq-SLAM [24] and its related work [28], [33] use a brute force method to perform sequence matching. It improves the matching accuracy but is time-consuming in practice and cannot be directly used in real-time. Thus, [19],[31],[14] proposed different methods to increase the efficiency of SeqSLAM by using particle filter, approximate nearest neighbor and Hidden Markov Model. Nevertheless, these methods' efficiency may decrease when the number of reference sequences is huge. [37] and [2] improves SeqSLAM by using dynamic query sequences and binary descriptors. These methods' performances highly depend on the condition of the environment, which is not robust enough to perform in changing and challenging environments. Then, [39] and [40] introduce a framework that can balance efficiency and accuracy by improving the SeqSLAM with a coarse-to-fine searching method. This concept performs stably with long reference sequences and has relatively low computation costs. Thus, our proposed localization pipeline used this concept to perform global initialization. After the initial position is located, we use local SeqSLAM to perform continuous tracking due to its high accuracy in changing and challenging environments. Since we only perform SeqSLAM locally instead of globally, we can largely reduce Seq-SLAM's running time and meet the real-time requirement.

## 2.4 Visual Place Recognition Datasets

There are already many existing Visual Place Recognition datasets. Many VPR related works use existing datasets as their benchmarks: NordLand [32], Pittsburgh [35], Tokyo 24/7 [36], NCLT [6], Oxford [22]. These VPR datasets are challenging and full of query images with illumination and viewpoint changes. However, these datasets are all collected using pin-hole cameras, which cannot be used to evaluate our omnidirectional visual inputs based localization system. There are also existing datasets PanoraMIS[3], Ford-Campus-Vision-and-LiDAR-Dataset[27] with omnidirectional visual inputs and accurate ground truth. But these datasets do not have illumination and viewpoint changes, which makes them not suitable for the VPR evaluation. Thus, we collected a changing (illumination and viewpoint changes) and challenging campus-scale VPR dataset with omnidirectional visual inputs and ground truth data. We use this dataset to evaluate our proposed localization pipeline. Furthermore, we believe that this dataset will be helpful in future visual place recognition research.

# Chapter 3

# Image Sequences Matching

This chapter introduces the brute-force Seq-SLAM[24] and then explains our proposed improved Fast Seq-SLAM inspired by the concept of coarse-to-fine [39], [40].

## 3.1 Seq-SLAM

The innovation of Seq-SLAM[24] is that it does not find a single reference image that best matches a query image. It combines the query image at the current timestamp with query images from previous timestamps to form a query sequence with length Q. It then finds a sequence from the reference sequence with length Q that best matches the query sequence. The first two sections are key parts of Seq-SLAM[24] : Contrast Enhancement and Sequence Match. The last section introduces some key parameters.

### 3.1.1 Contrast Enhancement

In order to make the difference matrix between reference sequence and query sequence more obvious, Seq-SLAM[24] applies a local contrast enhancement to each row of the difference matrix as shown in figure 3.1. The red box shows the row to apply local contrast enhancement as an example. The original element's value of this row is set to $D_i$ and the value after enhancement is set to $\hat{D}_i$. $\hat{D}_i = \frac{D_i - \bar{D}_l}{\sigma_l}$. $\bar{D}_l$ is the local mean and $\sigma_l$ is the local standard deviation with rows around row i in range of $R_{window}$.

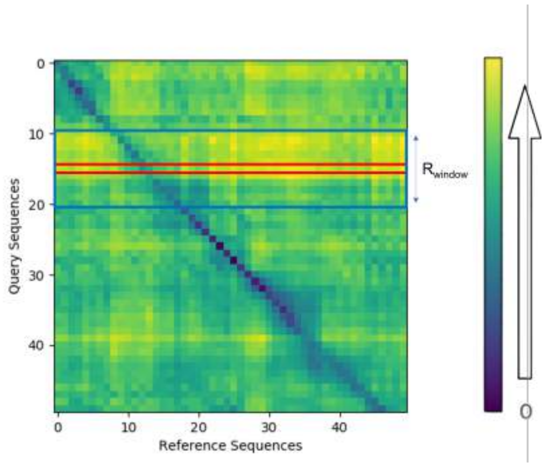

Figure 3.1: This is a difference matrix between a 50 frames query sequence and a 50 frames reference sequence, which is not the usual case. Usually, the reference sequence has far more frames than the query sequence. A lower value means a smaller difference between query and reference frames. The red box marks the row applied with local contrast enhancement, and the blue box shows the $R_{window}$ for calculating the local standard deviation and local mean.

After contrast enhancement, the darker block becomes even darker. A darker block implicates a smaller difference between the query and the reference frame, which means a better match. Figure 3.2 shows the local contrast Enhancement applied on row i only. And figure 3.3 shows the local contrast enhancement applied on the entire difference matrix.

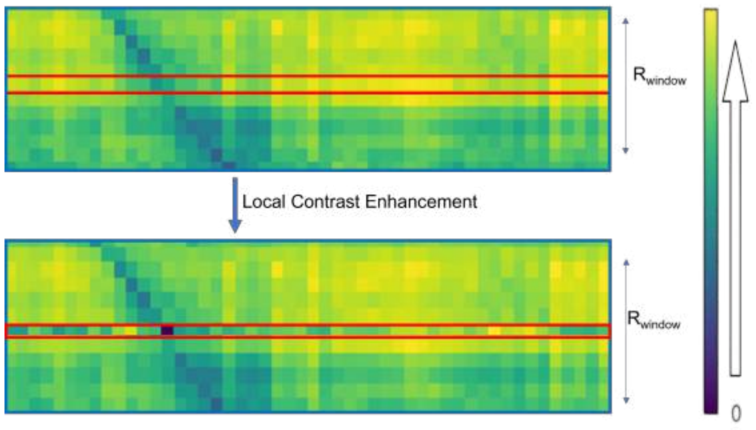

Figure 3.2: The top image is the original difference matrix. The bottom image is the difference matrix with local contrast enhancement applied to the row marked in red.

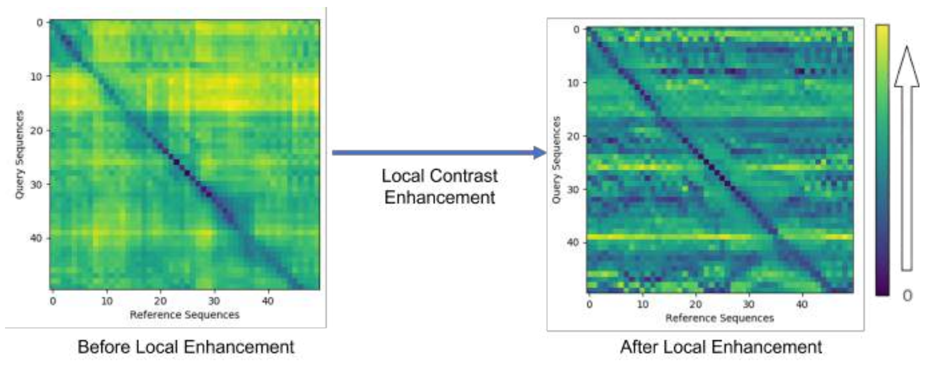

Figure 3.3: The left image is the difference matrix before local contrast enhancement and the right image is the difference matrix after local contrast enhancement with $R_{window}$ being 10.

### 3.1.2 Sequence Match

Seq-SLAM[24] performs a local search for most cells in the difference matrix. For example, we work on a cell at D[Q,R] (a cell in difference matrix at row Q and column R). A local search performs on multiple search lines with different slopes, where each search line contains $V_{ds}$ cells around D[Q,R]. The search is based on the assumption that velocities on repeated traverses of a path are roughly repeatable. So, we choose a $v$ between constant $v_{min}$ and $v_{max}$ to control the slope of the search line. For a specific $v$, its search line contains the following cells : $D[Q+v*(i-V_{ds}/2), R-V_{ds}/2+i], i \in [0, v_{ds}]$. Then we calculate the local match score $S_v$ by summing up its' search line's cells' values,

$$S_v = \sum_{i=0}^{v_{ds}} D[Q + v * (i - V_{ds}/2), R - V_{ds}/2 + i]$$

v $= v_{min} + i * v_{step}, i \in [0, (v_{max} - v_{min})/v_{step}]$. We have $(v_{max} - v_{min})/v_{step} + 1$ different $S_v$ for the cell D[Q,R]. We choose the smallest $S_v$ to be our $S_{best}$.

$$S_{best} = min(S_v), v \in \{v_{min} + k * v_{step} \mid k \in \{0, 1, ..., (v_{max} - v_{min})/v_{step}\}\}$$

The entire search process is illustrated by figure 3.4.

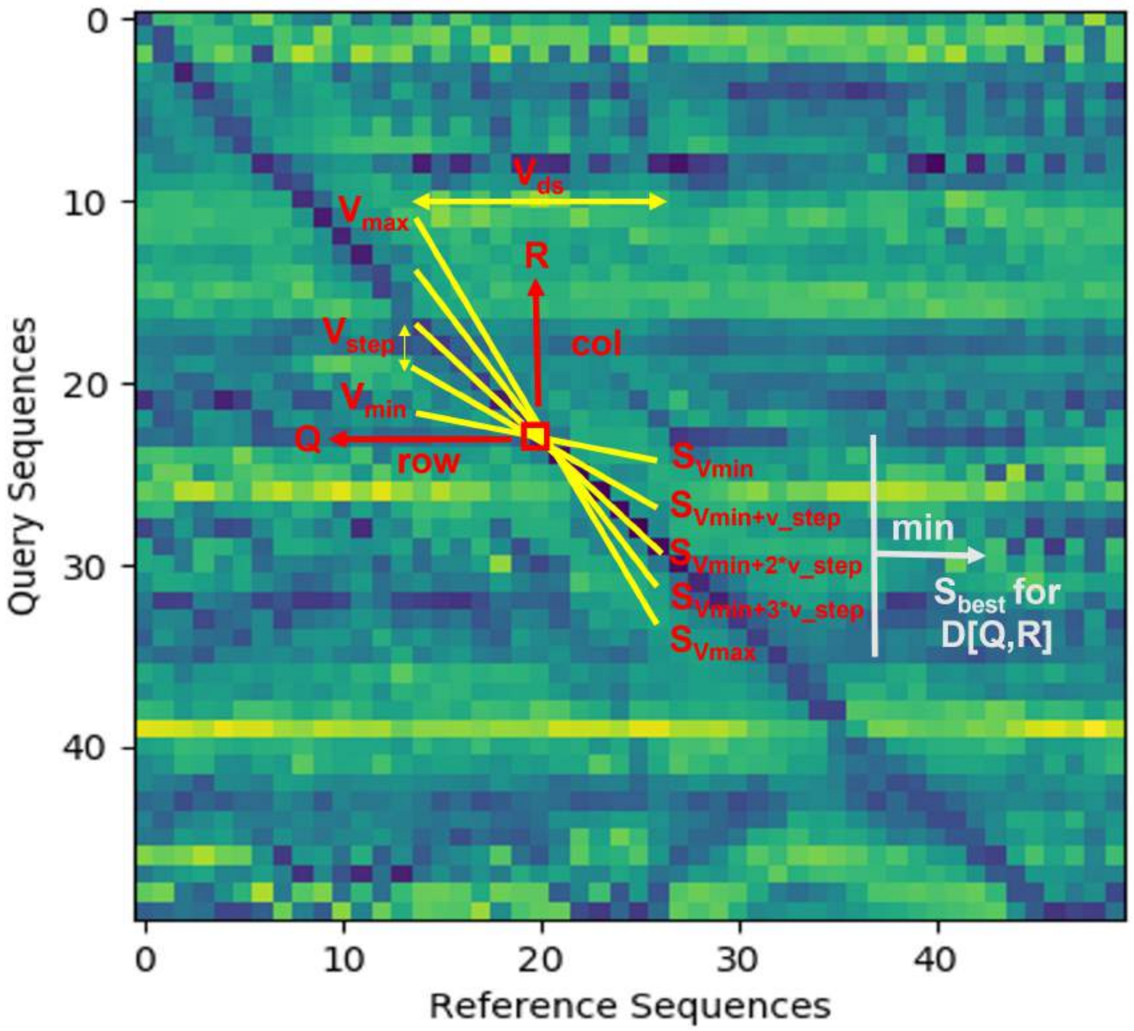

Figure 3.4: This image displays the local search process for the cell D[Q,R]. For this example, there are five yellow search lines with five different $v$ values. The smallest $s_v$ is set to be the $S_{best}$ for cell D[Q,R].

For each column R, we repeatedly do this local search process for all rows. With this repetition, we can get the $S_{best\_i}$ for each row i. We then choose the smallest $S_{best\_i}$,

$$S_{best\_in\_column\_R} = min(S_{best\_i}), i \in [0, \text{number of rows}]$$

We subsequently find the cell with $S_{best\_in\_column\_R}$ for each column. (except very first and very end depends on the length of $V_{ds}$, shown in figure 3.5).

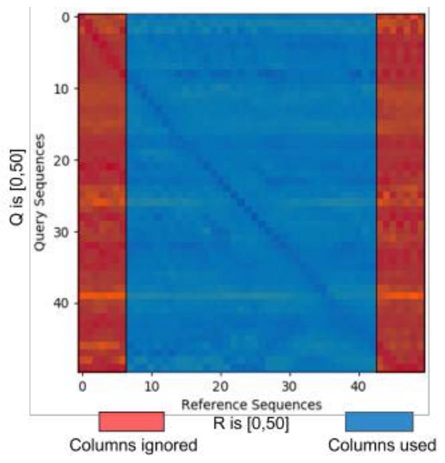

Figure 3.5: This image shows the columns being used for local searches and the columns discarded since they cannot provide search lines with length of $V_{ds}$.

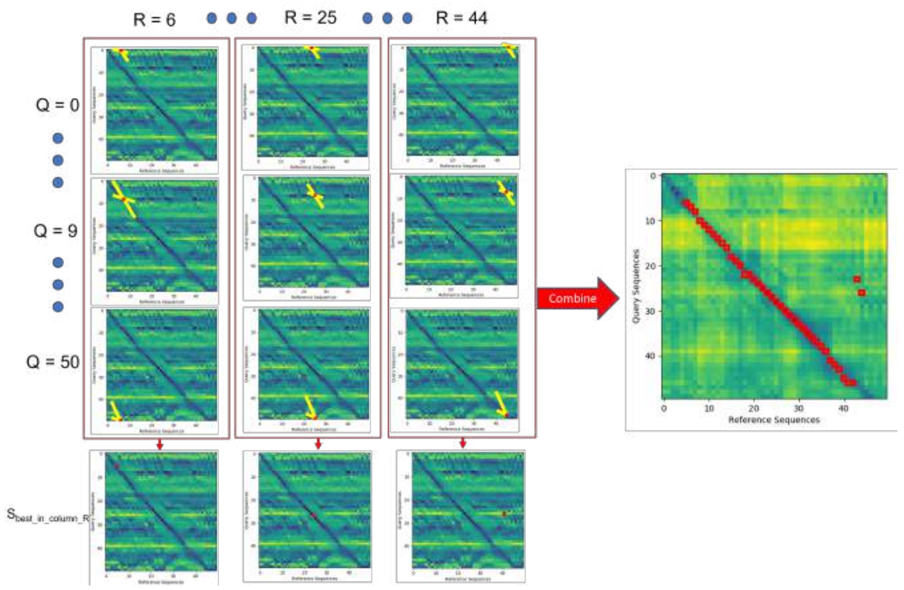

Figure 3.6: This image depicts entire process of doing local searches over a difference matrix with $V_{ds}$=10. The left matrix shows local searches through each cell, and the right image shows the final results on the difference matrix. The cell with $S_{best\_in\_column\_R}$ for each column is marked in a red box, which means the cell's row represented query frame matches with the cell's column represented reference frame.

This search process is illustrated in figure 3.6. As you can see from the graph, the cell with $S_{best\_in\_column\_R}$ for each column is marked in a red box. However, there needs to be a way to measure if the match is not an outlier based on the score. A simple way to do this is to use a constant threshold to preserve matches with scores above it and discard matches with scores below it. However, this constant threshold is too tricky to tune under changing and challenging environments. Thus, Seq-SLAM[24] comes up with a method called windowed uniqueness thresholding.

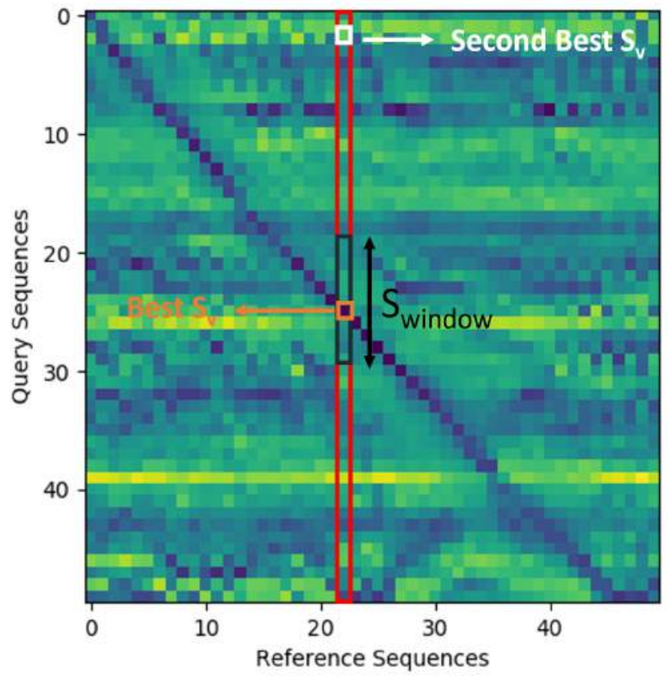

Figure 3.7: This image shows the selection of best $S_v$ (marked in orange) and second best $s_v$ (marked in white) outside a window of width $S_{window}$ (marked in black). The red box marks an example column R.

For windowed uniqueness thresholding, it looks at the uniqueness of the best $S_v$. The best $S_v$ is selected for each column. Each column has a second best $S_v$ at another row. The second best $S_v$ needs to be outside a window of width $S_{window}$ around the cell with the best $S_v$ to avoid similarity between the best and the second best. The uniqueness score is the $\frac{\text{best } S_v}{\text{second best } S_v}$. Figure 3.7 shows this process.

With the uniqueness score being larger, the difference between the best $S_v$ and the second best $S_v$ is not that big. For a big ratio, it means the uniqueness of the best $S_v$ is not that obvious since the second best $S_v$ is very close to the best $S_v$. One query image can only have one reference image match. Thus an ideal best $S_v$ should be much smaller than the second best $S_v$, which makes the uniqueness score very small. We can set a uniqueness threshold $S_{threshold}$ to eliminate matches with a uniqueness score greater than this threshold. This uniqueness threshold is easier to tune since it is less susceptible to a changing and challenging environment.

**Algorithm 1** Sequence Match algorithm

```
$v_{arr} \leftarrow linspace(v_{min}, v_{max}, v_{step})$
$qs \leftarrow$ len(query sequences)
$rs \leftarrow$ len(reference sequences)
$match \leftarrow []$                                        ▷ Matching result
for R $\leftarrow [v_{ds}/2, rs - v_{ds}/2]$ do
    $S_{best\_Q} \leftarrow \infty$
    $S_{v\_array} \leftarrow []$
    $match_Q \leftarrow 0$
    for Q $\leftarrow [0, qs]$ do
        for v $\leftarrow v_{arr}$ do
            $S_v \leftarrow \sum_{i=0}^{v_{ds}} D[Q + v * (i - V_{ds}/2), R - V_{ds}/2 + i]$
            if $S_v < S_{best\_Q}$ then
                $S_{best\_Q} \leftarrow S_v$
                $match_Q \leftarrow Q$
            end if
        end for
        $S_{v\_array}$ add $S_{best\_Q}$
    end for
    $S_{second\_best\_Q} \leftarrow \min(S_{v\_array}[: match_Q - S_{window}/2] + S_{v\_array}[match_Q + S_{window}/2 :])$
    uniqueness_score $\leftarrow \frac{S_{best\_Q}}{S_{second\_best\_Q}}$
    if uniqueness_score $< S_{threshold}$ then
        match add [R, $match_Q$, uniqueness_score]
    end if
end for
```

### 3.1.3 Key Parameters

This section introduces some key parameters of Seq-SLAM[24]. $R_{window}$ is used in local contrast enhancement. With the real-time requirements, query sequences should not have more than 60 frames. 10 is a good value for $R_{window}$ given this premise. $v_{min}$, $v_{max}$, $v_{step}$ and $V_{ds}$ are used in the sequence match. The agent's and the reference map's visual inputs are received and generated at the same interval. So we set $v_{min} = 0.8$, $v_{max} = 1.2$, $v_{step} = 0.1$, which is around 1. $V_{ds}$ is set to 10, which has been proved to be efficient and accurate in several different environments. $S_{window}$ and $S_{threshold}$ are used in match selection. Similar to the selection of $R_{window}$, given the average sequence number, 10 is a good value for $S_{window}$. And for the uniqueness $S_{threshold}$, we don't want to set it too strict since most of our environments are changing and challenging. Thus, we set the $S_{threshold}$ to be 0.97, which helps us eliminate some obvious outliers and reduce the possibility of creating false negatives during match selection.

## 3.2 Fast Seq-SLAM

This original Seq-SLAM[24] method is too slow for a real-time pipeline. The big O complexity for the algorithm is roughly O(MN), with M being the number of reference sequence's frames and N being the number of query sequence's frames. To solve this problem, we proposed the Fast Seq-SLAM. This algorithm uses the concept of coarse-to-fine to perform the Seq-SLAM[24], which largely reduces the computation cost. Details will be explained in the following sections.

### 3.2.1 Multi Resolution Sampling based Method

The high level of this coarse-to-fine concept is to set multiple resolutions for both the query and the reference sequences. We do Seq-SLAM[24] between the down-sampled query sequence and many reference sequence segments at first. After iterations, we do Seq-SLAM[24] between the original query sequence and a few reference sequence segments.

Initially, particles are spread evenly on the reference sequence. Both the reference and query sequences are set to very low resolution initially, which is every eight frames in the graph. Each particle represents a specific reference sequence segment with a length the same as the down-sampled query sequence. These particles' segments can have overlaps, controlled by a variable called $\tau$, shown in figure 3.8. The number of particles is calculated in the following way. We set $P_{init}$ to be the number of initial particles, Q to be the number of frames of the non-down-sampled query sequences and R to be the number of frames of the non-down-sampled reference sequences

$$P_{init} = \frac{R}{Q} * \tau$$

$$\tau = \frac{1}{1 - \text{overlap}}$$

The particles are more dense with larger $\tau$, leading to more overlapped regions.

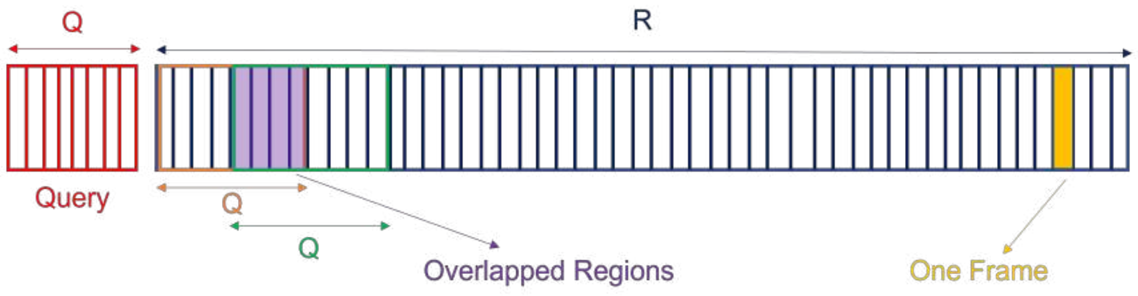

Figure 3.8: This figure shows how the number of initial particles is calculated. The red box is the query sequence, and the blue box is the reference sequence. Each small rectangle inside is a frame; one example is marked in yellow. The variables Q and R are the length of reference and query sequence. The orange and the green boxes are two segments of two particles. The segment always has the same length as the query sequence with the current resolution level. The purple part shows two particle overlapped regions. Overlap = $\frac{\text{overlapped region}}{\text{Q}}$, which is $\frac{1}{2}$ and makes $\tau = 2$ (overlapped rate 50%).

Particles have two attributes. First is the index of the segment it represents. Since our goal is to localize, we choose the last frame to be the index. The second is the weight. Initially, weights are just $\frac{1}{P_{init}}$ for every particle.

$$P = [p^1, p^2, p^3, ...p^{P_{init}}]$$

$$p^j = [i^j, w^j]$$

.

Resolution levels are related to the rate of down-sampling. Figure 3.9 shows the process to down-sample both query and reference sequence with a specific resolution level. More frames are skipped with a larger resolution level, resulting in a short reference and query sequence. The resolution level needs to be carefully tuned to balance speed and accuracy. Down-sampled sequences with different resolution levels are used in the subsequent algorithm instead of the original sequence.

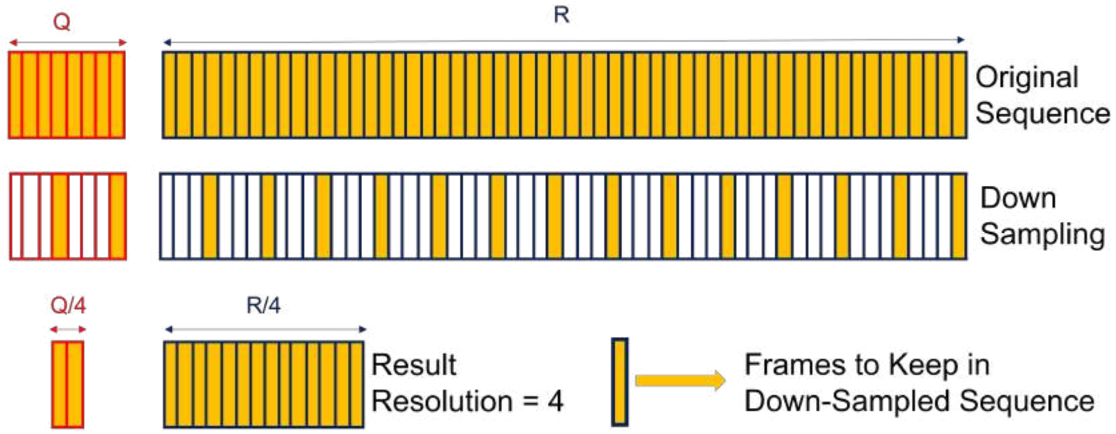

Figure 3.9: This picture illustrates the down-sampling process. The original sequences in the first row have every frame. When performing down-sampling, the algorithm starts from the last frame and only preserves one frame every re_level frames. In this graph, re_level = 4. The picked frames form the down-sampled reference and query sequence. The resulting sequences are $\frac{1}{re\_level}$ of their original lengths.

.

After one round of sequence match on all the segments, each particle's segment performs index shifting, weight updating, and segment merging. The sequence match continues until the iteration times reach max_iter or the current particles' efficiency is less than 50%. When one of these two requirements meets, the resolution level reduces by half and decreases the down-sample rate. After the resolution level reaches 1, the reference and query sequences restore to their original length. The algorithm can determine the final match by doing sequence matches on the remaining few particles' segments. The entire process is illustrated in figure 3.10.

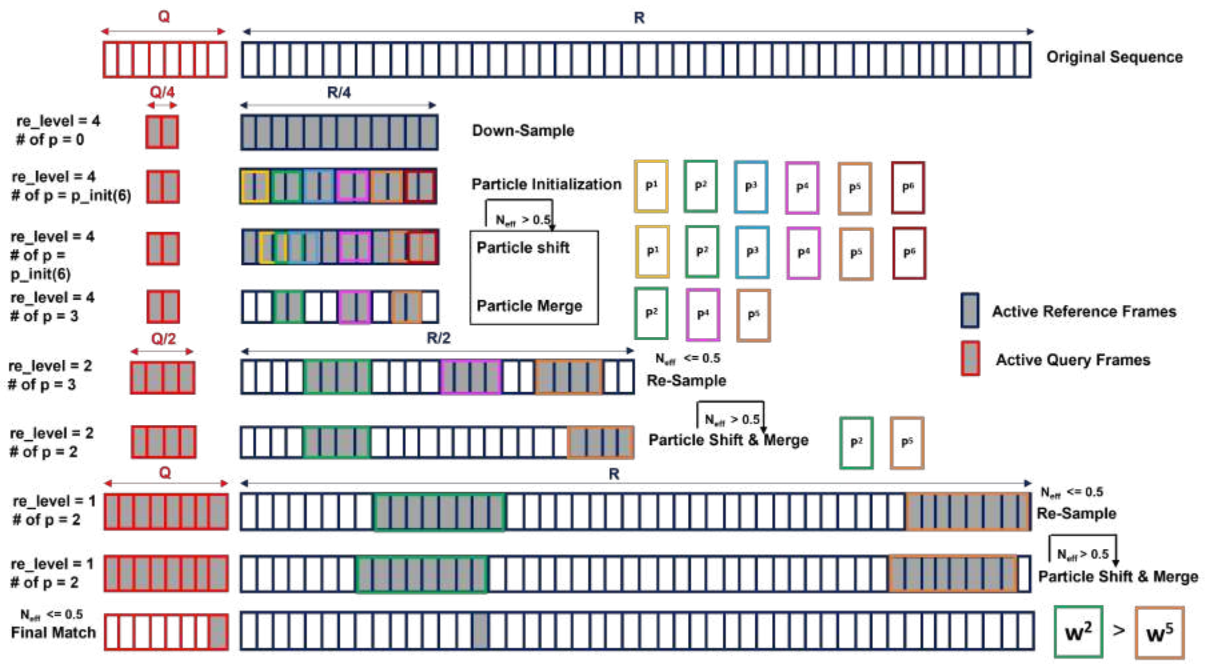

Figure 3.10: This figure depicts the entire structure of the Fast Seq-Match method. # of p means the number of particles. re_level is the resolution level, which is set as four initially. For simplicity, only six particles are used and marked in different colors. The process starts by down-sample the original reference and query sequence and ends with finding the final query-reference match. $N_{eff}$ is the particle efficiency. The grey shades mean active frames, which are used in the algorithm. The other white frames are discarded. Since our system is used for localization, we match the last frame of the particle segment to the query sequence's last frame since the last frame represents roughly the current location.

### 3.2.2 Particle Shift

When the sequence is down-sampled, and the particles are evenly distributed, all particle segments go through the process of Seq-SLAM[24]. Each particle's segment generates an array of matches. Given the assumption that the query sequence is part of the reference sequence and both query and reference frames are generated using roughly the same distance intervals, the ideal matches should form a line across the difference matrix with the slope around -1 and intercept around 0 (set the top-left corner of the difference matrix as (0,0) in the Cartesian coordinate plane), which is shown in figure 3.11. We want to shift the segment to make all matches stay on the ideal line.

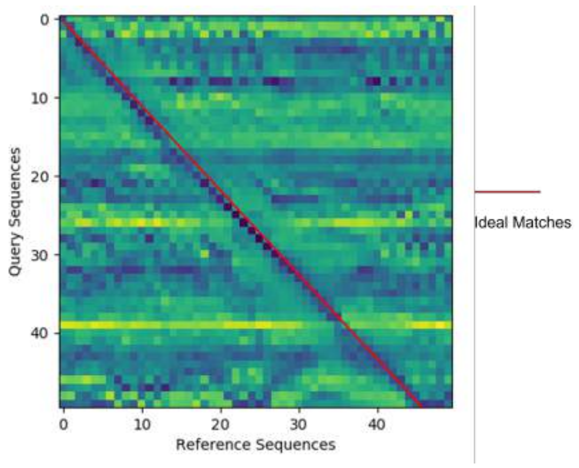

Figure 3.11: This graphs shows an example difference matrix. As one can observe, the ideal match pattern (marked in red line) is roughly on a line with slope around -1.

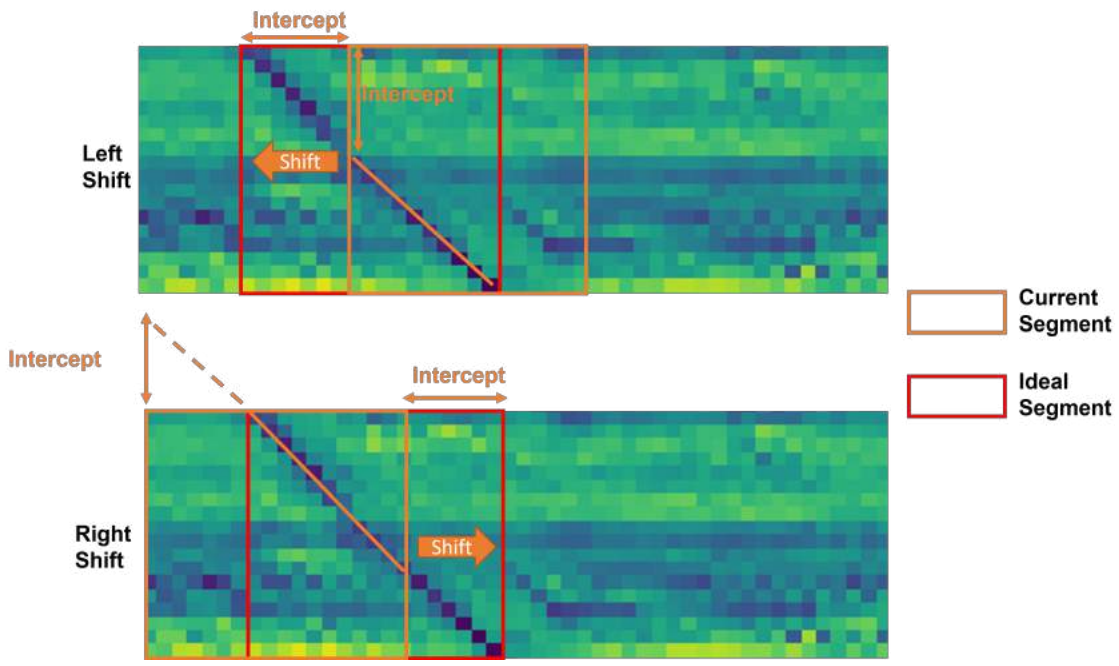

Figure 3.12: This graphs illustrates the concept of segment shifting. The shift amount is decided by the value of the intercept. When intercept is positive, the segment shift to the right. While when intercept is negative, the segment shift to the left. $P^j.i = P^j.i -$ intercept. We set the top-left corner of the difference matrix as (0,0) in the Cartesian coordinate plane.

We can calculate the intercept of the line formed by the matches. We shift the particle's segment to make the line's intercept reaching zero. The whole process is illustrated in figure 3.12. To obtain the intercept of the line formed by matches of the segment, we use a brute-force method to search all the potential lines with slope = -1 passing the segment's difference matrix. Given Q as the number of frames of the query sequence, potential lines can have intercepts from -Q to Q. Due to our multi-resolution method, the difference matrix is very small. Thus the brute-force method has a low computation cost. We find the line with most inliers. When a match's distance to the line is less than 2, it is considered an inlier of the line. Given a match at row Q and col R, the following equations show its relationship to the line x + y - intercept = 0.

$$D[Q,R] = \begin{cases} \text{inlier}, & \text{if } \frac{Q+R-intercept}{\sqrt{2}} \leq 2 \\ \text{not inlier}, & \text{otherwise} \end{cases}$$

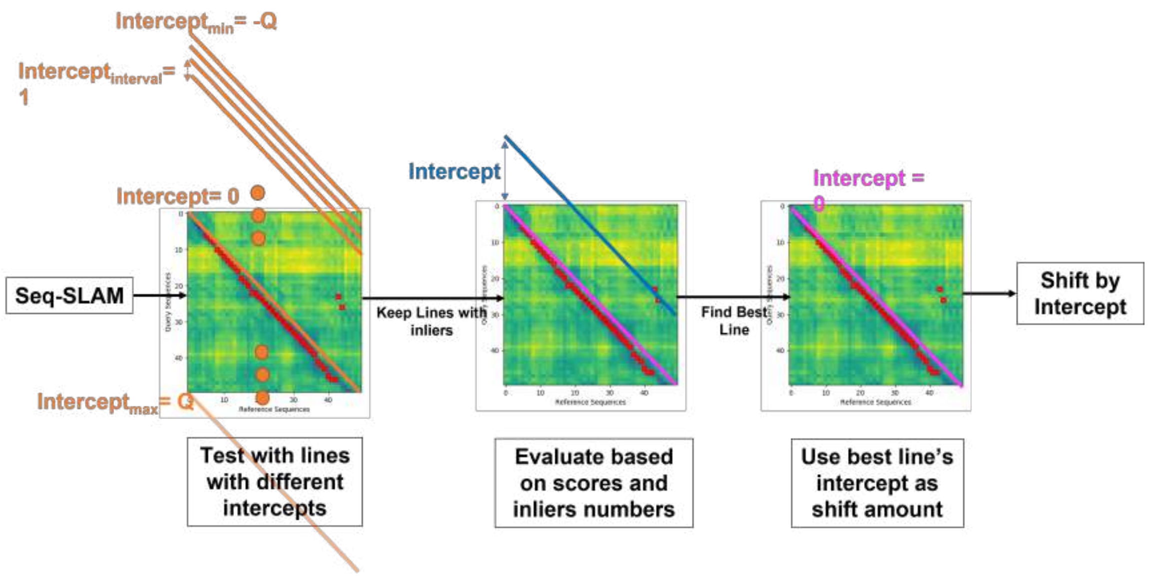

Figure 3.13: This figure depicts the process of finding the amount to shift. Q is the number of frames of the query sequence. This process starts after obtained matches from the sequence match. Orange lines are all the potential lines that could pass through the matrix. Only the pink line and the blue line has inliers, which are then evaluated based on their inlier numbers and line score. Finally, the pink line has been chosen, and its intercept is set to be this segment's shift index, which is just 0.

Each line has a score and a inliers' count number. The line with most inliers is set to be the best line. When there is a tie, the line with the highest score is chosen. The score is one minus the average of all its inliers' uniqueness scores. We set the number of inliers to be N and the uniqueness score of each match be $s_u$

$$\text{line\_score} = \frac{\sum_{i=1}^{N} \text{inlier}_{s_u}^{i}}{N}$$

The intercept of this chosen line is used as this segment's shifting amount. Figure 3.13 depicts this entire procedure.

### 3.2.3 Particle Merge and Resolution Update

After all particles' segments shift to their optimal position, the particle merge starts. With particle shifting, some particles' segments may have large overlaps or even are at the exact same place. In order to increase efficiency, particles with very close indexes are merged. We set the closeness to be three frames, and the particle with a larger weight is preserved. The weight of the particle is updated by multiplying the best line's line_score. With a higher line_score, the particle's segment has more possibility of being at the final matched index. The weight is normalized and the particle efficiency is calculated after particle's merge. For weight of remaining particles 1 to N,

$$w_n^i = \frac{w_n^i}{\sum_{i=1}^{N} w^i}$$

The particle efficiency for N particles is calculated using the following formula:

$$N_{eff} = \frac{\sum_{i=1}^{N} (w_n^i)^2}{N}$$

The most effective particles' distribution would be one particle's weight is 1 and all other particles' weights are 0. Worst distribution would be all particle's weight is $\frac{1}{N}$, which is the initial distribution. For the best distribution: $N_{eff} = \frac{1}{N}$ For the worst distribution, $N_{eff} = \frac{N}{N} = 1$. Thus, we set the effective threshold to be 0.5. When $N_{eff}$ is lower than 0.5, it means the particles are effective enough for the next resolution level. In order to prevent the algorithm to loop forever, we set a maximum iteration limits to wait for $N_{eff}$ to reach below 0.5. If the iteration times reach the maximum iteration limits, the resolution reduces by half ignoring the value of $N_{eff}$. The maximum iteration is set to 3 to balance time and accuracy.

With resolution increasing from the old level to the new level, the current particle's index needs to increase as well. Previously skipped frames are now restored with re-sampling. We set the old resolution level to a and the new resolution level to b.

$$\text{new\_index} = \begin{cases} \text{old\_index} * \frac{a}{b} + 1, & \text{if } [(a-1) \mod \frac{a}{b}] = 0 \\ \text{old\_index} * \frac{a}{b} + [(a-1) \mod \frac{a}{b}] - 1, & \text{otherwise} \end{cases}$$

$$\begin{aligned}
O_{\text{Fast}} &= O(\sum_{i=0}^{re\_max} \frac{\frac{M}{N} * \tau}{2^{re\_max-i}} * ((N/2^i)^2 + N/2^i) + \frac{\frac{M}{N} * \tau}{2^{re\_max-i}}) && (3.1)\\
&= O(\sum_{i=0}^{re\_max} \frac{\frac{M}{N} * \tau}{2^{re\_max-i}} * ((N/2^i)^2 + 1)) && (3.2)\\
&= O(\sum_{i=0}^{re\_max} \frac{\frac{M}{N} * \tau}{2^{re\_max-i}} * (N/2^i)^2) && (3.3)\\
&= O(\sum_{i=0}^{re\_max} \frac{M * \tau}{N * 2^{re\_max-i}} * \frac{N^2}{2^{2i}}) && (3.4)\\
&= O(\sum_{i=0}^{re\_max} \frac{M * \tau}{N * 2^{re\_max-i}} * \frac{N^2}{2^{2i}}) && (3.5)\\
&= O(\sum_{i=0}^{re\_max} \frac{M * N * \tau}{2^{re\_max+i}}) && (3.6)\\
&= O(\frac{M * N * \tau}{2^{2*re\_max}} * \frac{1 - 2^{re\_max}}{1-2}) && (3.7)\\
&= O(\frac{M * N * \tau * (2^{re\_max} - 1)}{2^{2*re\_max}}) && (3.8)\\
& && (3.9)
\end{aligned}$$

### 3.2.4 Complexity Analysis

The complexity in Big O for Seq-SLAM[24] is O(MN). The same complexity analysis must be performed for Fast Seq-SLAM as well to prove its improvement. The number of reference sequence frames is M and the number of query sequence frames is N. For each particle, Seq-SLAM[24] takes $O((N/2^i)^2)$ and index shift takes O(N/$2^i$)

at resolution level $2^i$. Particles merging takes O(p). With p particles, the total complexity is roughly O($p * ((N/2^i)^2 + N/2^i) + p$). Particles number decrease roughly by half on each resolution level. We set lowest resolution level be $2^{re\_max}$ and highest resolution level be 1. p = $\frac{p_{init}}{2^{re\_max-i}} = \frac{\frac{M}{N} * \tau}{2^{re\_max-i}}$.$O_{\text{Fast}}$ is calculated from equations (3.1) to (3.9). Given the parameters tuned in section 8.1 : $\tau = 3.0$, re_max = 2. $O_{\text{Fast}} = O(\frac{M*N*3*3}{16}) = \frac{9}{16}O(MN)$. Fast Seq-SLAM's big O complexity reduces the original big O complexity by nearly 50%. With even lower $\tau$ and bigger re_max, the complexity can be further decreased.

# Chapter 4

# Robotic System

After introducing the main algorithm of our localization pipeline, we present the robotic system we build.

## 4.1 Mapping Rover Robot

This mapping rover robot (shown in figure 4.1) includes a Ricoh THETA V omnidirectional camera, a Velodyne VLP-16 LiDAR, a T265 Intel Realsense Tracking Camera (VIO), and an Xsens MTI IMU, and a Jetson AGX Xavier Nvidia AI Robot Driver Development Kit as its main computing unit. It is used for data collection and high-quality LiDAR odometry map generation. We also use a surface pro as a tablet to visualize data and control sensors.

## 4.2 Localization Helmet

The localization helmet(shown in figure 4.2) has four main components: Two Leopard Imaging Fisheye Cameras, an Epson IMU, a Jetson AGX Xavier as the main computing unit, and a WIFI module for communication. The omnidirectional image is generated using two fisheye cameras. The image stitching techniques are explained in section 4.3. The Localization Agent is equipped with a helmet, a bag to store battery, and a surface pro tablet to visualize data and control sensors and algorithms.

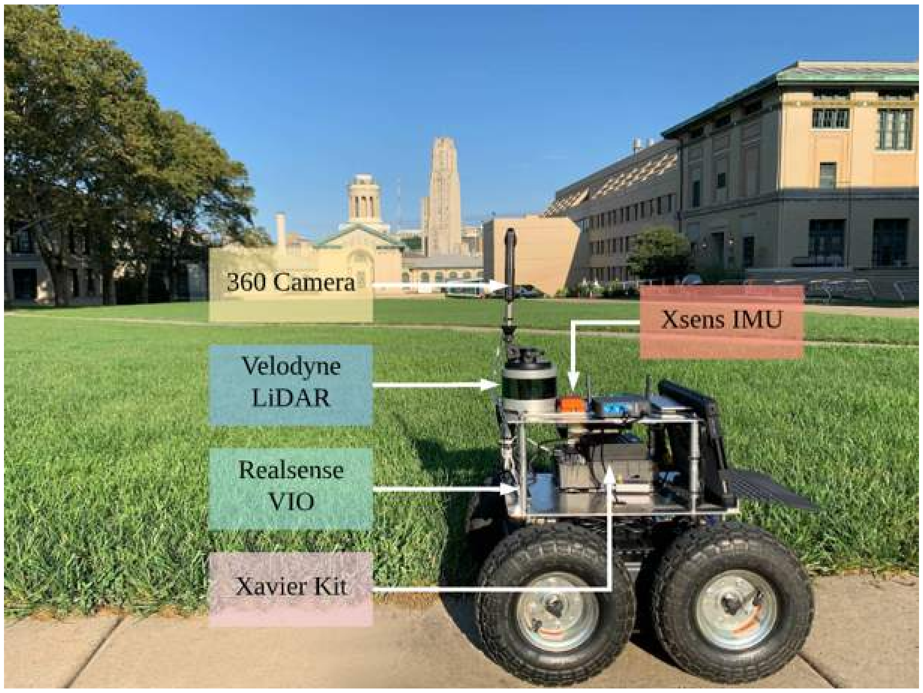

Figure 4.1: Mapping Rover Robot.

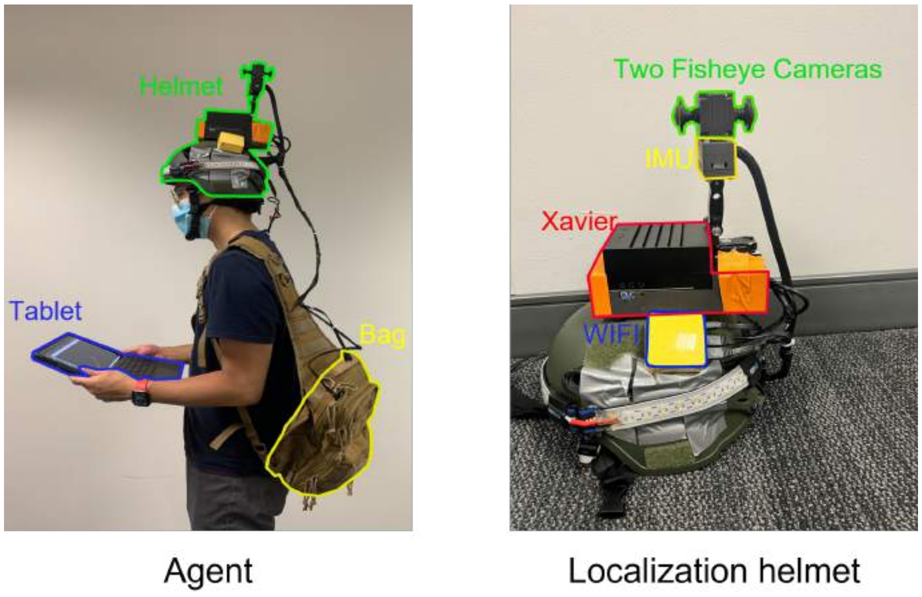

Figure 4.2: The left picture is the localization agent and the right picture is the localization helmet with omnidirectional camera.

## 4.3 Image Stitching

We use the algorithm RIGID MOVING LEAST SQUARES proposed by [15] and [30] to complete panoramic image stitching. Figure 4.3 shows the main pipeline of the stitching algorithm.

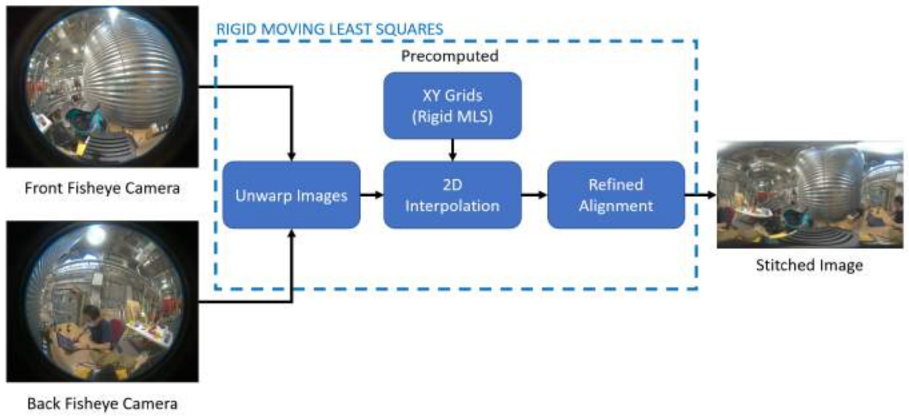

Figure 4.3: The processing flow of the RIGID MOVING LEAST SQUARES algorithm.

The algorithm first unwarps both fisheye images. Then it performs a 2d-interpolation using a pre-computed Rigid moving least square (MLS). The rigid MLS aligns points around the stitching boundary, which registers two unwarped fisheye images. However, some misalignments still remain. Thus, a refined alignment is used to further align the images. For the rigid MLS, control points p are selected from the front fisheye's unwarp image, and q is selected from the back fisheye's unwarp image. These points are located in the overlapping regions of the two images, where $p_i$ and $q_i$ represent the same real-world location. Rigid MLS is used to find a function $f_r$ to deform the back fisheye's unwarp image and align it with the front fisheye's unwarp image. The example images with points p and q are shown in figure 4.4. For every point v in the front fisheye's unwarp image and the selected control points p and q,

$$f_r(v) = \sum_i \hat{q}_i(\frac{1}{\mu_s}A_i + q_*)$$

$w_i$ is the weights for each $p_i$ and $q_i$

$$w_i = \frac{1}{|p_i - v|^{2\alpha}}$$

$p_*, q_*$ are weighted centroids and $\perp$ is an operator on 2D vectors such that $(x, y)^\perp = (-y, x)$

$$p_* = \frac{\sum_i w_i p_i}{\sum_i w_i}$$

$$q_* = \frac{\sum_i w_i q_i}{\sum_i w_i}$$

$$\hat{p}_i = v - p_*$$

$$\hat{q}_i = v - q_*$$

$$\mu_s = \sum_i w_i \hat{p}_i \hat{p}_i^T$$

$$A_i = w_i \begin{pmatrix} \hat{p}_i \\ -\hat{p}_i^{\perp} \end{pmatrix} \begin{pmatrix} v - p_* \\ -(v - p_*)^{\perp} \end{pmatrix}^T$$

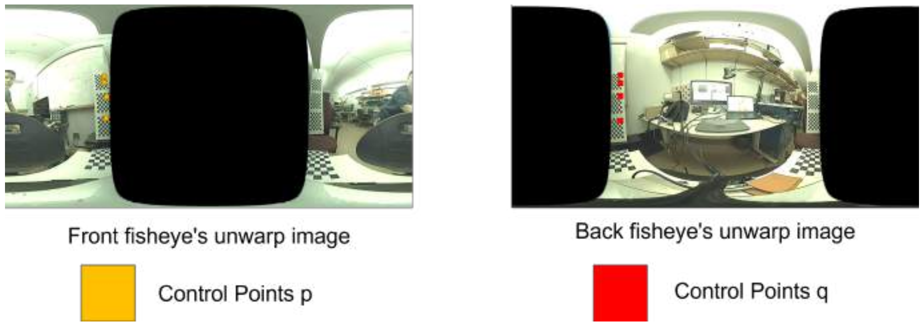

Figure 4.4: Left: front unwarp fisheye image and $p_i$ on the overlapping areas(yellow boxes). Right: back unwarp fisheye image and $q_i$ on the overlapping areas(green boxes).

The refined alignment performs a fast template matching and utilizes the matching displacements on both stitching boundaries to create eight pairs of control points. These points are used to solve for a 3*3 affine matrix to warp the deformed image.

# Chapter 5

# Localization System

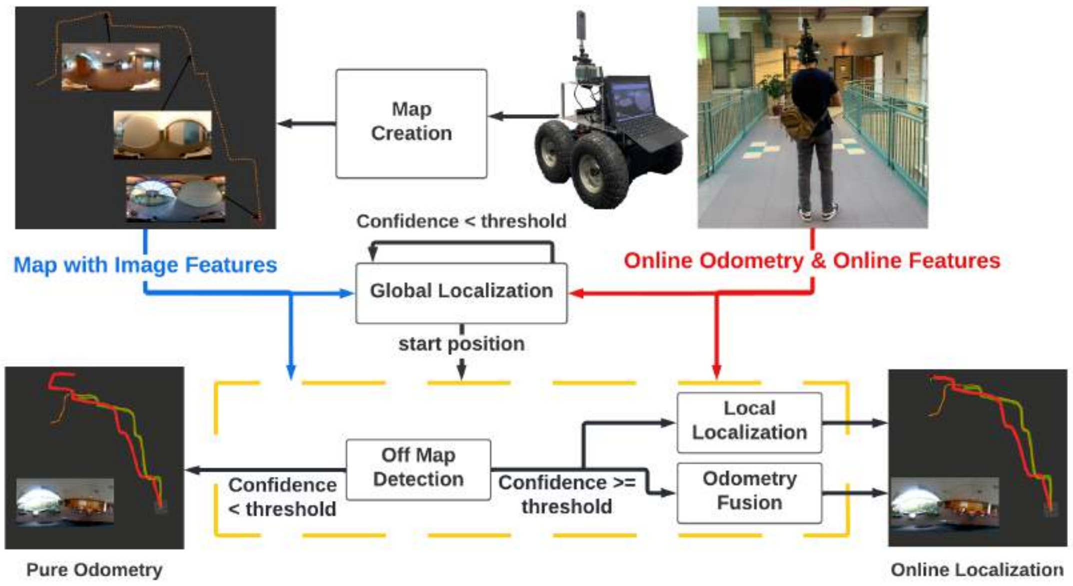

Figure 5.1: The localization system framework.

Chapter 3 and chapter 4 introduce the main algorithm and localization devices. This chapter presents the framework and different modules of the localization system.

## 5.1 System Overview

The system has five main parts: Map creation, Global Localization, Local Localization, Odometry Fusion, and Off Map Detection. Each part will be explained with details in the following sections. The created map has two types of information: LiDAR odometry [43] and Omnidirectional images. Agent receives two types of information: Omnidirectional visual inputs and real-time odometry generated by VINS-MONO [29]. Figure 5.1 shows the system framework.

## 5.2 Map Creation

Our localization system is based on VPR. Thus, creating a high-quality map with visual inputs is the first step. The mapping robot (introduced in 4) generates a LiDAR odometry trajectory with refined point cloud data. The LiDAR odometry is saved every 0.25 meters. For each saved odometry, an associated image's feature array is also saved, which is generated by CoHOG [41](section 8.2 explains detailed reasons). Figure 5.2 shows the generated high-quality map's format. There is a height difference between the agent's visual input and the mapping rover's visual input. Figure 5.3 shows visual inputs' comparisons. Our pipeline needs to be robust enough to handle the altitude changes.

## 5.3 Global Localization

When an agent enters a new environment with the map, the pipeline does not know the agent's starting position. Thus, a global localization module is needed. The module uses Fast Seq-SLAM. Since we utilize the concept of sequence matching, the system needs to store enough frames to form a sequence to start global localization. Thus, the agent needs to walk around 7 to 12 meters initially (depending on the query sequence length) for the pipeline to localize its initial position on the map. The module's output position is categorized as success when the matched segment's line_score is greater than $threshold_G$, which is set to 0.2 based on experiments. Figure 5.4 shows the module's working flow. Agent receives the visual input and the odometry every

0.25 meters, which is the same as the pre-generated map.

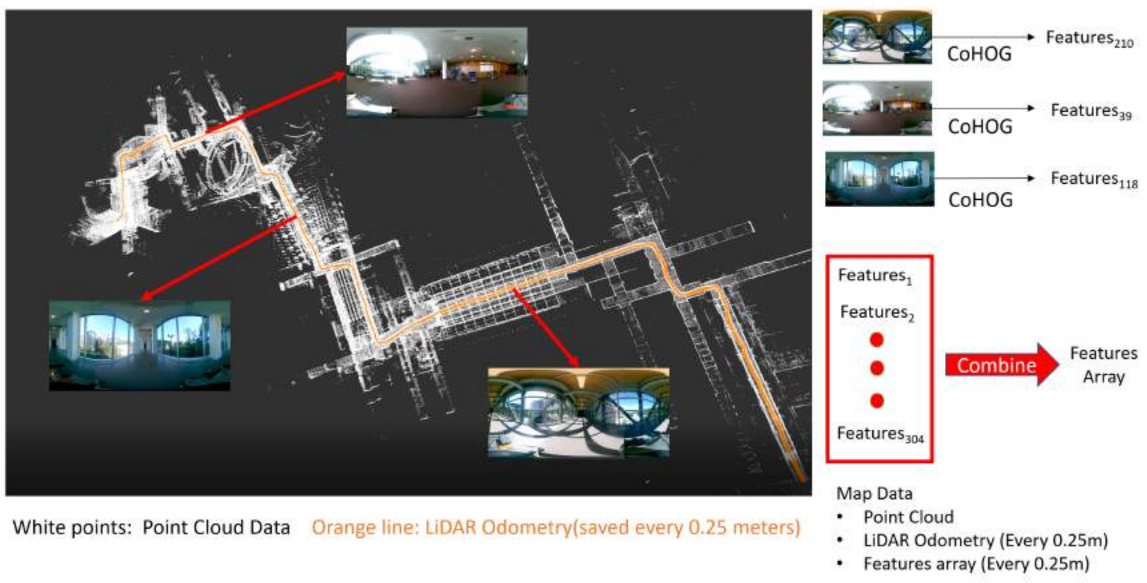

Figure 5.2: This graphs shows the format of the created map. In this example, 304 odometry are stored in the map. Each odometry is associated with an image, which is marked by red arrows. These images are then transferred to features and stored in the map.

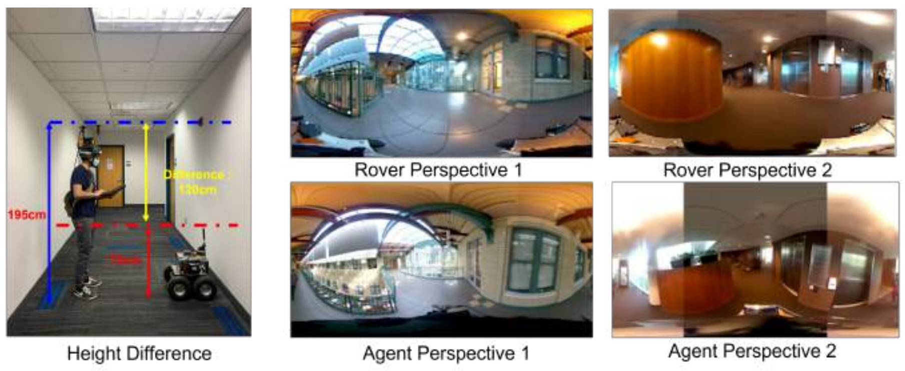

Figure 5.3: Viewpoint differences between the mapping robot and the localization agent.

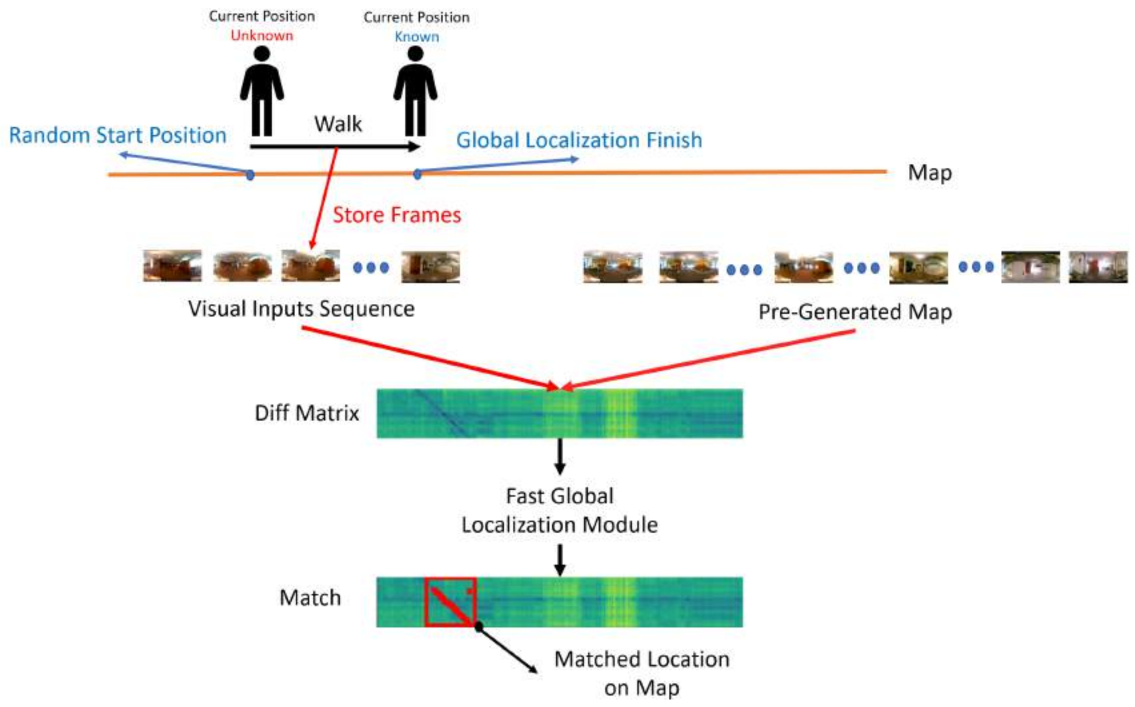

Figure 5.4: The working flow of the global localization module. The agent starts from a random unknown position on the map. After walking for a certain distance, the agent receives enough visual inputs, which can form into a query sequence. The query sequence and the reference sequence(generated map) form a difference matrix through the place representation technique. The module computes the match with the difference matrix using Fast Seq-SLAM and outputs the agent's current position on the map.

## 5.4 Local Localization

After global localization, the pipeline knows the agent's current location on the map. Instead of doing Seq-SLAM with the entire reference sequence, we only need to perform it with a local neighborhood of the reference sequence around the current location. In this way, we can benefit from the high accuracy of Seq-SLAM and not worry about its high computation cost. The big O complexity for Seq-SLAM in this pipeline is now O($N^2$) instead of O(NM), where N << M (N is the number of query sequence's frames and M is the number of reference sequence's frames, which is the map). We obtain a matched reference sequence's frame index after global localization. We set it to $I_R$. When a new visual input comes in, we first increase $I_R$ by one since we know that reference and query sequences have roughly the same distance intervals. Since the visual input is received per 0.25 meters, around 10HZ. The local localization can only perform around 3HZ(more details in section 8.3.3). So, we cannot perform local localization on every visual input. We record the missing visual input numbers as $I_M$. The system stores previous inputs' features, and it performs a Seq-SLAM between query features[len(query features)-S+1:] and reference features[$I_R + 1 + I_M - N : I_R + 1 + I_M$] for the forward direction. And between query features[len(query features)-S+1:] and reference features[$I_R - 1 + I_M - N : I_R - 1 + I_M$] for the backward direction. With the output matches, we want its formed line's intercept to reach zero, as stated in section 3.2.2. Thus, we shift the matched reference sequence's frame index by the amount of the intercept. The updated $I_R$ after global localization is $I_R$+1+M-intercept or $I_R$-1+M-intercept depending on the directions, in which the current position is the map's odometry data's $I_R$ element.

## 5.5 Odometry Fusion

The agent receives odometry information every 0.25 meters (around 10 HZ) from VINS-MONO[29]. We transform the received odometry positions using the map's coordinates system and add the agent's starting position as an offset. We fuse the transformed odometry positions and the local localization's output map's positions to make the system more robust. The fused information is passed into gtsam[13], which is our back-end factor graph optimizer. Each transformed odometry position is associated with visual input. Since local localization can only give matched positions at 3HZ instead of 10HZ, not all the transformed odometry is associated with a matched position on the given map. The transformed odometry is passed into the back-end as a factor. The matched position is passed into the back-end as a ground truth value for the associated odometry factor. The back-end optimizer does a quick optimization every time it receives a ground truth value. The local localization matched position is only added to the gtsam back-end when the match result's line_score is greater than $threshold_l$, which is set to 0.1 based on experiments.

## 5.6 Off-Path Detection

The agent might not always stay on the pre-generated map. So, our localization system includes an off-path detection module. As shown in 5.1, the off-path detection module decides whether to use pure odometry or the fusion of local localization's match results and odometry. Off path is defined when the agent deviates from the map, which means the agent starts to explore places not visited before. In this case, Seq-SLAM cannot output a match on the given map. Our system needs to have the ability to find out when the agent is off the path and when is back on the path again. 5.5 shows an example of off-path localization, including the off-path and back-on-path parts.

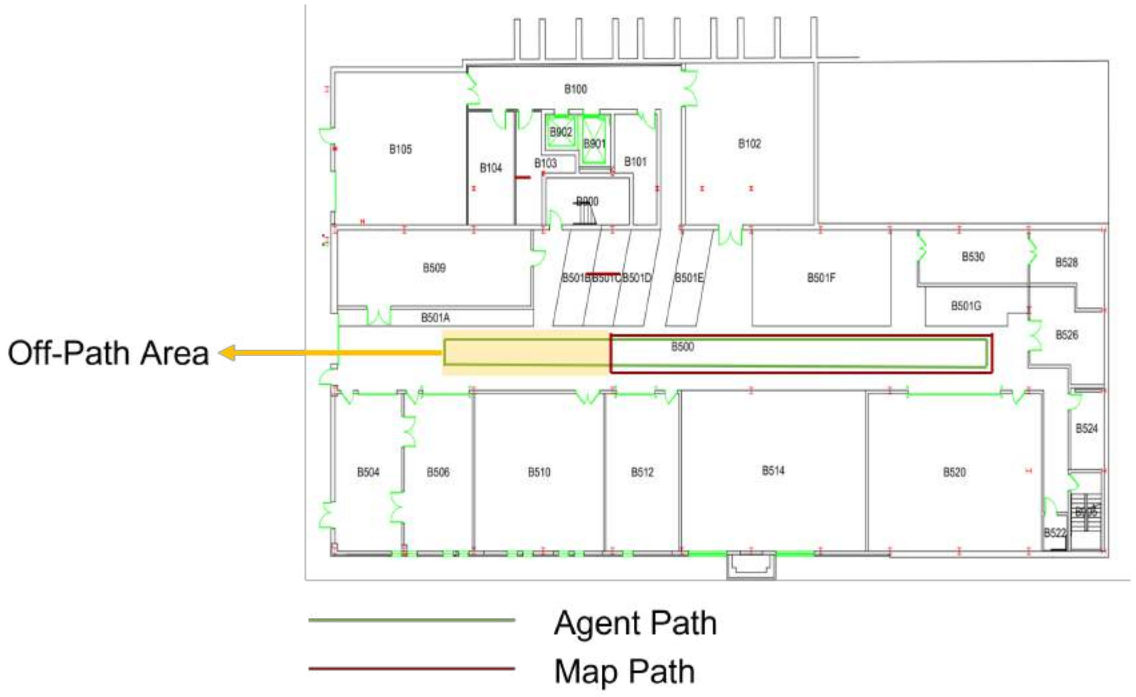

Figure 5.5: The figure is an example of an agent's movement, including off-path. The red line is the pre-generated map, and the green line is the agent's path. When the agent enters the yellow-shaded area, he/she is off the map. When the agent moves out of the yellow-shaded area, he/she is back on the map again.

When the local localization module outputs match with line_score below $threshold_{ol}$ (0.05 based on experiments) 5 times continuously, the off-path detection module determines the agent is now off the path. The local localization module stops running, and the system keeps running global localization until the agent is back on the map. Meanwhile, the output estimated location only relies on pure transformed odometry at off-path status. When the global localization module generates a match with line_score greater than $threshold_G$ (0.2 based on experiments), the system determines that the agent moves back to the map again. The local localization module starts running again, and the global localization stops. The local localization module uses the match generated by global localization as the current $I_R$.

# Chapter 6

# Dataset

To evaluate our pipeline and facilitate future research, we collect a visual place recognition dataset with omnidirectional images in challenging and changing environments.

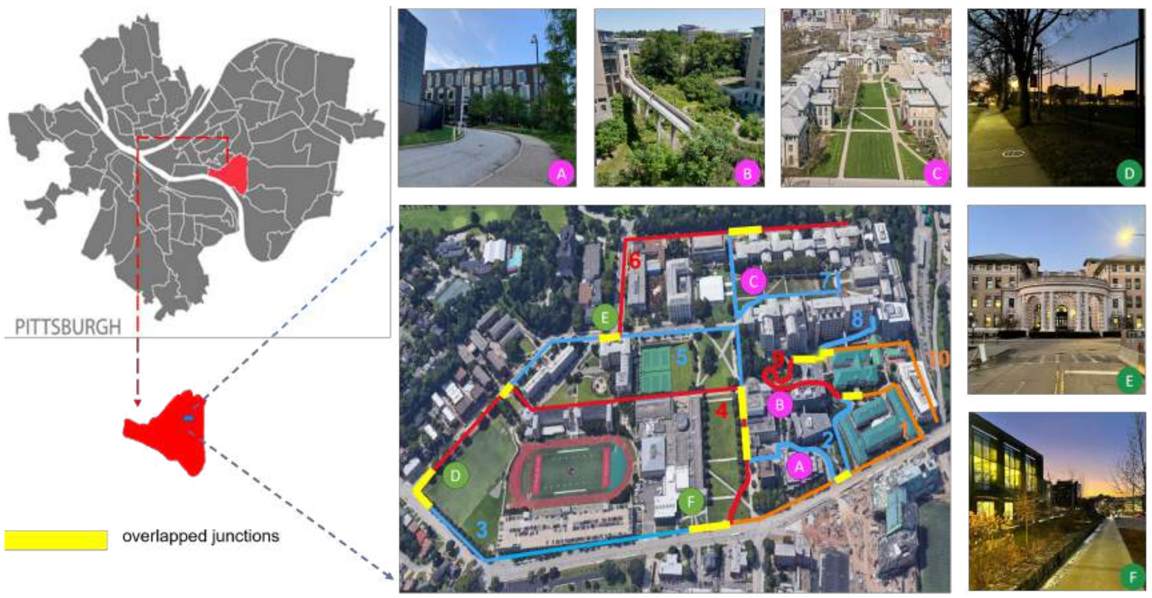

Figure 6.1: This picture shows ten collected trajectories inside Carnegie Mellon University. Their overlapped junctions are marked in yellow, around 20 to 30 meters. Different colors of trajectories do not have a specific meaning, which is only used for better clarity since they are inter-connected. The trajectory number are also marked for future reference.

## 6.1 Overview

This dataset focuses on localization with omnidirectional visual inputs in outdoor campus-type environments. We collected 80 real-world unmanned ground vehicles (UGV) sequences using a rover robot (mentioned in section 4.1). There are in total ten different trajectories. For each trajectory, we traversed eight times, including forward(start point to end point)/backward(endpoint to start point) directions and day-light (2pm to 4:30pm)/night-light (6am to 7am or 5pm to 6pm). Eight times includes two forward sequences and two backward sequences during day-light and two forward and two backward sequences during dawn-light. Each trajectory is at least overlapped at one junction with the others, and some trajectories even have multiple junctions. This feature enables the dataset to be used in LiDAR place recognition and multi-map fusion tasks. Figure 6.1 is the overview of ten collected trajectories on the google map. Their overlapped junctions are also marked. Backward and forward creates viewpoint changes. Day-light and night-light make illumination changes. Dynamic objects and Cars' flashing headlights on the street create challenges for VPR. Figure 6.2 shows these changes and challenges with detailed examples.

## 6.2 Data Format

The dataset consists of 4 types of data, described as :

- *Global maps*: Global maps are processed to contain the 3D structure of each trajectory which is provided in Point Cloud Data (`PCD`) file format.
- *Odometry*: Odometry is generated by LOAM [43] and provided in (`TXT`) file format.
- *Ground Truth*: Ground Truth is processed by Interactive SLAM [17] to generate the relative positions of the other sequences compared to the reference sequence. The data is provided in (`TXT`) file format.
- *Omnidirectional pictures*: For each key pose within odometry, a corresponding high resolution (1024x512) omnidirectional picture is provided in (`PNG`) file format.

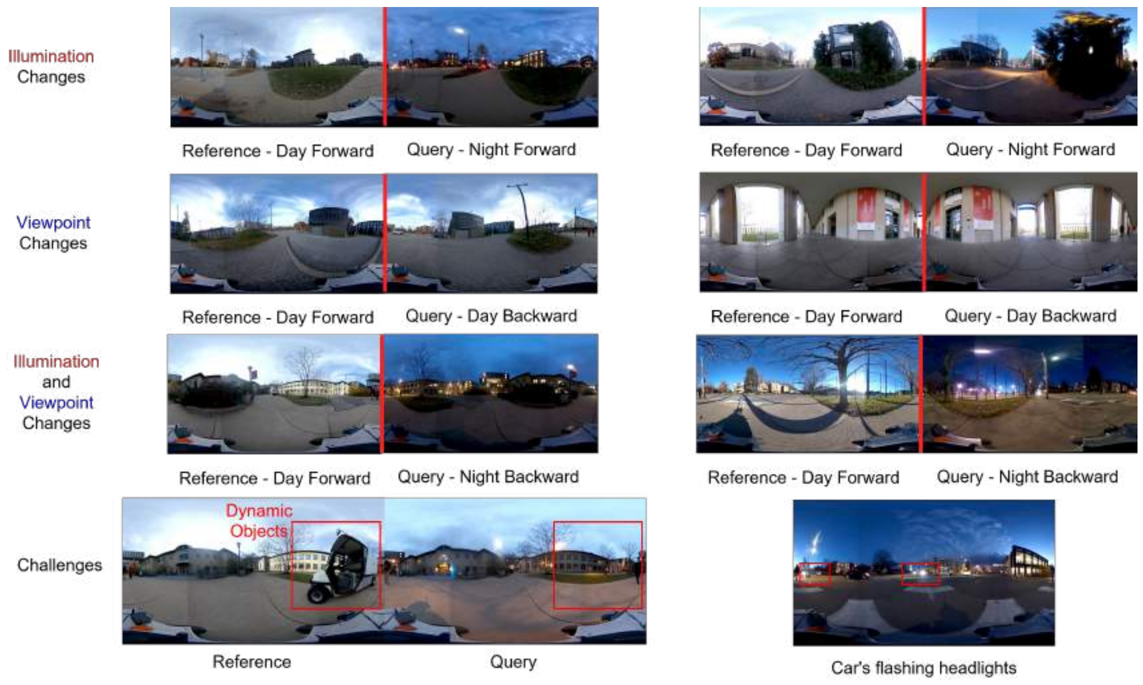

Figure 6.2: This picture shows the changes and challenges described in the paragraph above, such as viewpoint changes, illumination changes and some challenges.

## 6.3 Ground Truth Generation

For each trajectory, one out of the eight sequences is selected as the reference sequence and the rest seven sequences are function as non-reference sequence, which are segmented to many query sequences during evaluations. Its poses are generated by LOAM [43] and later optimized by Interactive SLAM[17], as shown in figure 6.3. We then used Interactive SLAM[17] to generate the relative positions of the other sequences compared to the reference sequence. These data can be used as relative ground truth data. Interactive SLAM[17] needs the user to first do a manual loop closure to connect the two sequences. We fix our reference sequence, and the manual loop closure only shifts the other sequence. After shifting, the software does an automatic loop closure detection with each point on both sequences to refine the relative positions. When the automatic loop closure reaches a certain threshold, users can stop it and output the relative position of the other sequence in the world frame of the reference sequence. Figure 6.4 depicts the whole process.

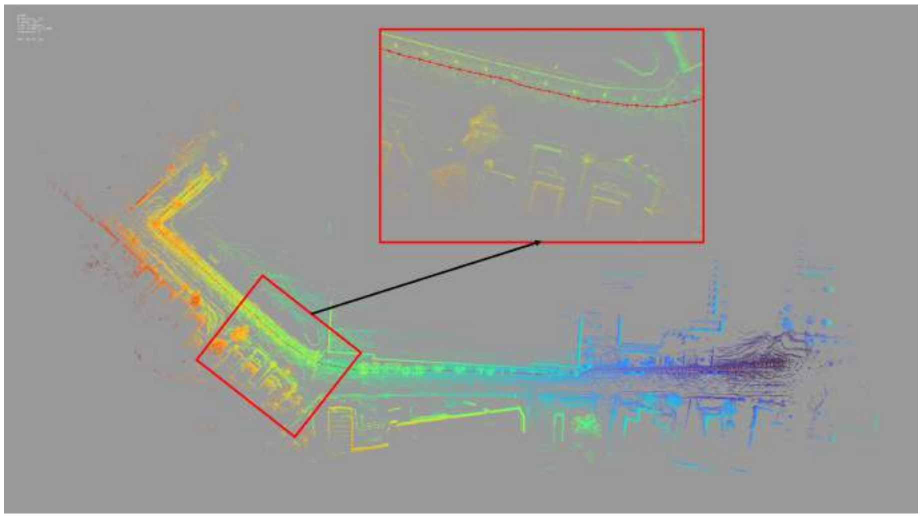

Figure 6.3: The Interactive SLAM's[17] representation of a trajectory. Each key pose's odometry is represented by a red dot and each of them has its associated point cloud data. Trajectory 3 is used for this example.

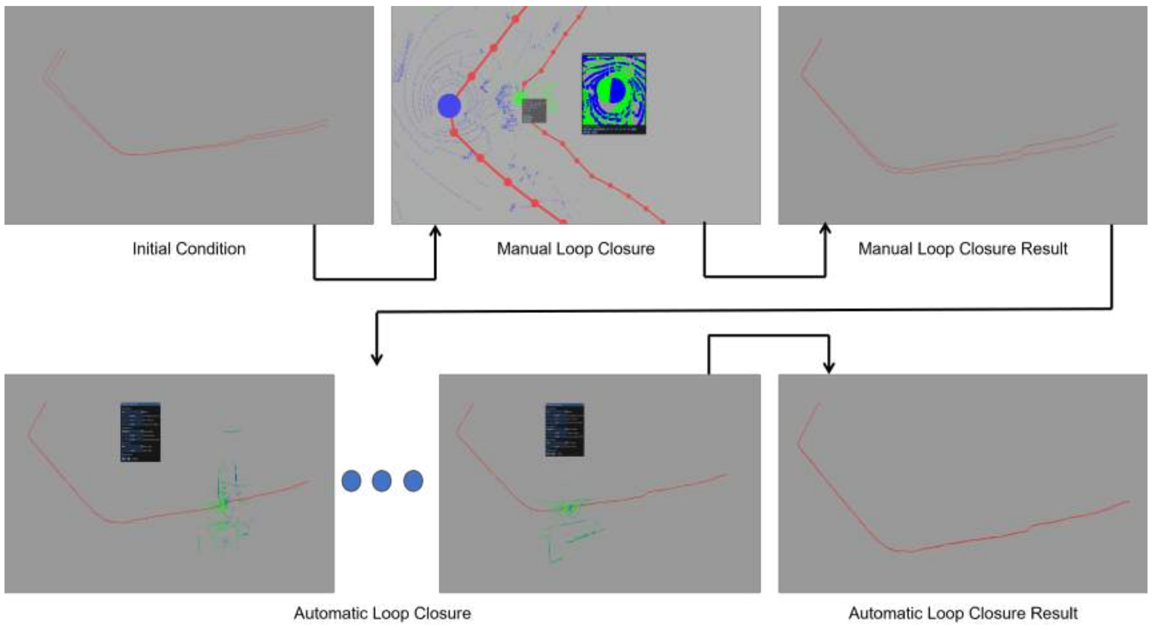

Figure 6.4: This picture shows the entire process to generate the ground truth data using Interactive SLAM[17]. Trajectory 3 is used for this example.

# Chapter 7

# Experimental Setup

The chapter first introduces the metrics we used for reasoning. The following section introduces two extra indoor datasets used for evaluation. The third section introduces three different place match techniques we used for evaluations and comparisons. The final two sections explain how we measure time and accuracy for both global and local localization modules.

## 7.1 Metrics

For metrics, we have two primary metrics for VPR based algorithm. VPR algorithms usually are evaluated with query sequences as part of the reference sequences. Thus, there are no true negatives. Therefore, precision and recall are used to evaluate true positives, false positives, and false negatives. Since our system has the off-path detection module, we test with query sequences with true negatives. In this case, Receiver-Operating -Characteristic(ROC) curve is used.

### 7.1.1 PR Curve

We set True-Positive to TP, False-Positive to FP and False-Negative to FN.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

For VPR, with a query frame and a score_threshold, TP is a correctly retrieved image of a place based on ground-truth information. FP represents an incorrectly retrieved image of a place based on ground-truth information. And FN is a correctly retrieved image, but with a match score less than score_threshold. For most VPR datasets, the query sequences do not have unseen parts compared to the reference sequences. Thus there are no True-Negatives. Ground-truth matches exist for all the query images in the datasets.

Different score_threshold values can lead to different precision and recall pairs, which form a curve. The area under the curve is called AUC-PR. For AUC-PR, its ideal value is one, which means the precision equals one for all recall values.

The odometry difference between the retrieved reference image and the query image is used to decide the correctness of this retrieval. We set the odometry difference tolerance to 2.5 meters.

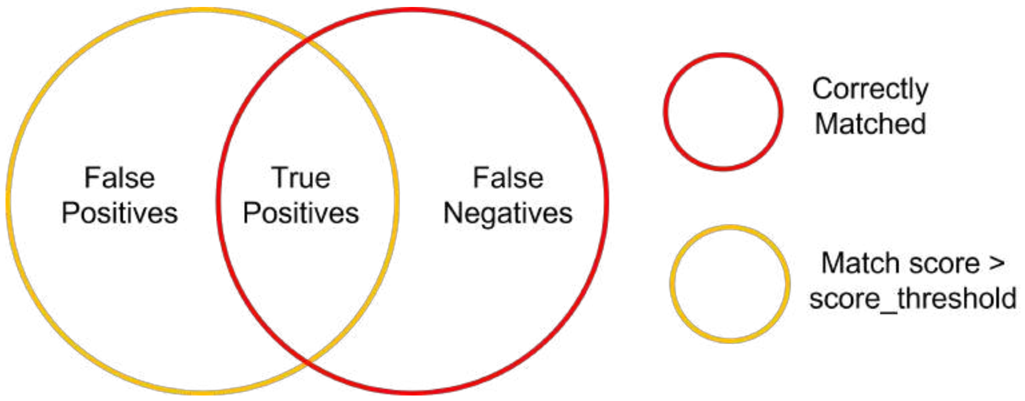

Figure 7.1: The Venn diagram shows the relationship between TP, FP and FN under the metric precision and recall.

### 7.1.2 ROC Curve

Precision and recall do not consider True-Negatives. True-Negatives mean the query images for which the ground truth correct reference match does not exist, which can be considered new places. Receiver-Operating -Characteristic(ROC) curve is a plot between true-positive rate(TPR) and false-positive-rate(FPR). We set True-Positive to TP, False-Positives to FP, False Negatives to FN, and True Negatives to TN.

$$\text{TPR} = \frac{TP}{TP + FN}$$

$$\text{FPR} = \frac{FP}{FP + TN}$$

Same as the PR curve, different score_threshold values lead to different TPR and FPR, forming the ROC curve. The area under the curve is called AUC-ROC. A perfect AUC-ROC is one, which means TPR equals one for all FPR. For AUC-ROC below 0.5, the algorithm cannot separate visited and new places or even create opposite labels.

Like the PR curve, We set the odometry difference tolerance to 2.5 meters.

## 7.2 Indoor Datasets

Two extra indoor datasets are collected at Carnegie Mellon University and Hawkins (A Va Pittsburgh Health Care Int) to make the evaluation datasets more comprehensive. These datasets provide confined spaces and narrow and long corridors with repeated features, as shown in figure 7.2

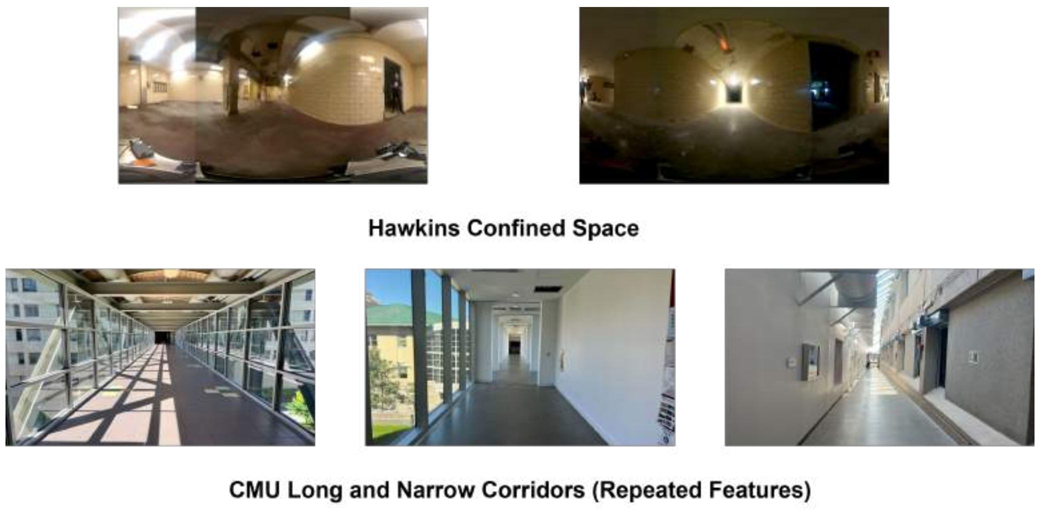

Figure 7.2: Examples of confined spaces, narrow and long corridors with repeated features in this indoor datasets.

To mimic the altitude changes mentioned in section 5.3, we collected sequences with high viewpoints and low viewpoints using the mapping rover robot with ground truth. Figure 7.3 shows the collection device.

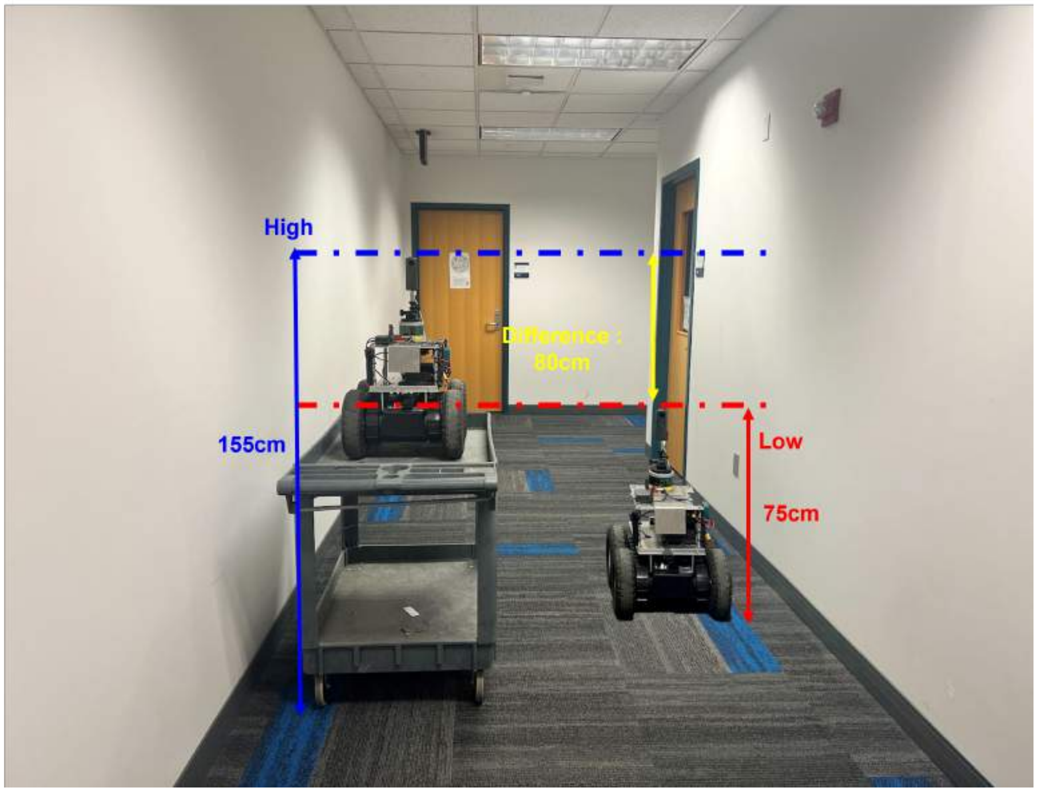

Figure 7.3: Mapping rover robot is used to collect dataset with low viewpoint. We put it on a cart to collect sequence with high viewpoint.

This way, we can conduct quantitative analysis to tune the parameters with altitude change. We also collected more drastic illumination changes. The reference sequence is recorded with lights. We turned off some lights and used our LED to record some not-bright sequences. We then turned off all the lights to record the dark query sequence. These changes are shown in figure 7.4.

The CMU indoor dataset contains one reference sequence with low view point and direction forward. And four non-reference sequences being :

- Non-reference 1: low viewpoint and direction forward.
- Non-reference 2: low viewpoint and direction backward.

- Non-reference 3: high viewpoint and direction forward.
- Non-reference 4: high viewpoint and direction backward.

The Hawkins indoor dataset contains one reference sequence with low view point and bright illumination condition. And four non-reference sequences being :

- Non-reference 1: low viewpoint and not-bright illumination condition.
- Non-reference 2: low viewpoint and dark illumination condition.
- Non-reference 3: high viewpoint and not-bright illumination condition.
- Non-reference 4: high viewpoint and dark illumination condition.

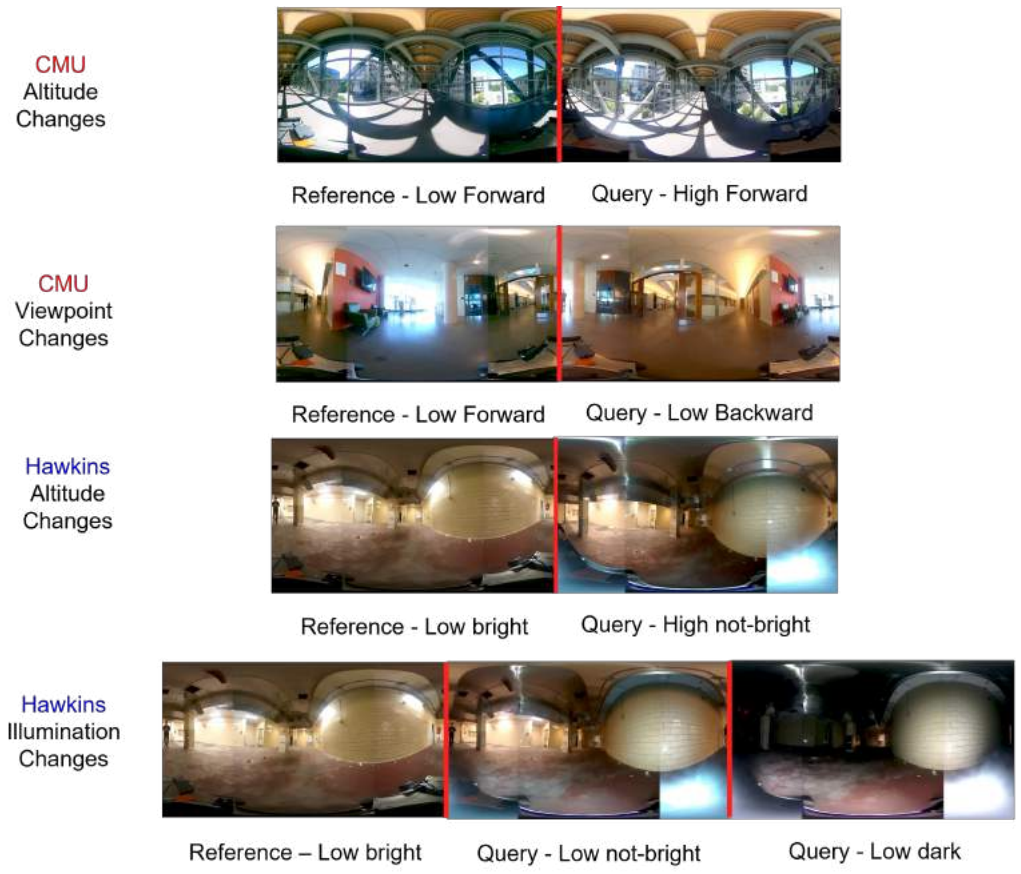

Figure 7.4: Viewpoint and Illumination changes in two extra indoor datasets.

## 7.3 Three different place match techniques

As mentioned in section 2.3, there are three different place match techniques. The most direct way is single image match. One can also exploit the characteristics of VPR by using image sequences, which leads to Seq-SLAM and Fast Seq-SLAM. Figure 7.6 illustrates the comparison between these three methods. Single Image match compares a single image with the entire reference sequence. The match result is the reference frame with the highest score. Seq-SLAM matches the query sequence with the entire reference sequence. And Fast Seq-SLAM uses multi-resolution sampling to match the query sequence efficiently with the entire reference sequence.

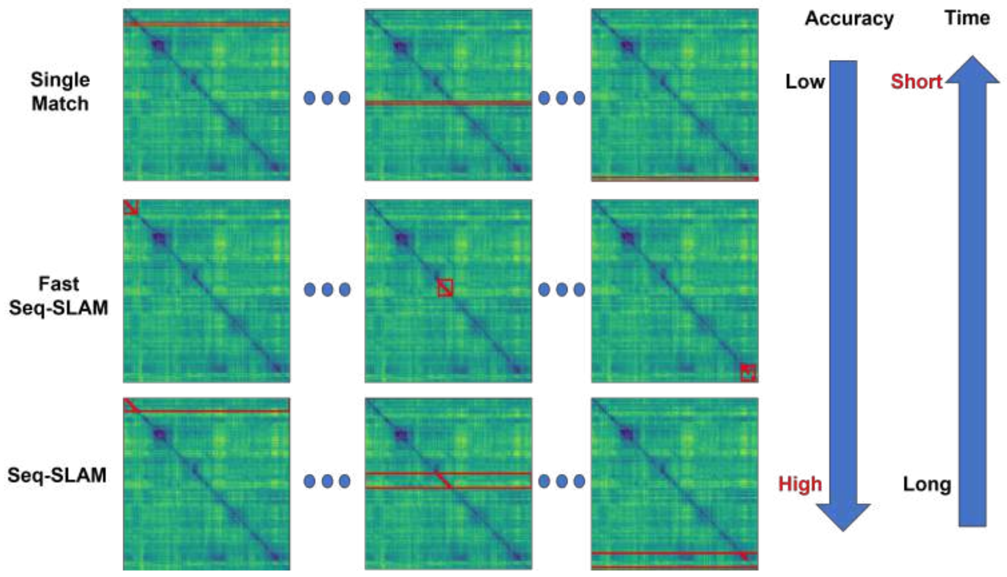

Figure 7.5: This graphs shows the difference between three different place match techniques. The first row shows the single image match. The second row represents Fast Seq-SLAM and the Last row shows the Seq-SLAM.

Theoretically: For AUC-PR, it should be Seq-SLAM > Fast Seq-SLAM > Single Match. And for run-time, it should Seq-SLAM > Fast Seq-SLAM >> Single Match. As mentioned in chapter 5, we choose Fast Seq-SLAM, which balances the run-time and accuracy as our global localization method. And use local Seq-SLAM to do local localization with initial positions, which has high accuracy and robustness to environmental changes. By doing the Seq-SLAM locally, we can avoid the high computation cost. The section 8.3.2 and 8.3.3 prove the theory mentioned above and our algorithm choices for tasks.

## 7.4 Global Localization Evaluation

We set the non-reference sequence length to be NR and the reference sequence length to be R. Frame numbers for the non-reference sequence are set from one to NR, with an interval equal to one. The query sequence used for sequence-based matching methods has length Q. Q should be << NR, which Q's value is discussed in section 8.1

All three methods mentioned in the previous sections are evaluated for global localization. We test with all the possible query frames to assess the global localization. Since sequence-based matching methods need enough frames to form a query sequence, the first query frame that can be used to evaluate is the frame with index Q of the non-reference sequence, which enables Q frames to be stored. The index range for usable query frames is from Q to NR, with an interval equal to one.

We record each usable query frame's matched reference frame index and matching score for a single image match. The matched result's odometry difference can be calculated using ground truth data and the matched reference frame index. If the difference is smaller than the odometry tolerance (mentioned in section 7.1), we set the match to be one and otherwise zero.

For sequence-based matching methods (including Seq-SLAM and Fast Seq-SLAM), 7.6 shows the evaluation process. For each evaluation between a non-reference sequence and a reference sequence, we obtain query sequences [1 to Q], [2 to Q+1], ...,[NR-Q+1, NR] of the non-reference sequence, which all have the length of Q. We then use these query sequences to perform Seq-SLAM or Fast Seq-SLAM with the reference sequence. We can get each query sequence's matched reference frame index and matching score. Just as single matching, matched reference frame indexes are transferred to either one or zero.

We can combine these matches and scores and use the precision and recall metric to get the AUC-PR and PR-curve for each non-reference sequence given a reference sequence. We integrate each non-reference sequence's matched results to form a trajectory's matched results, which can be evaluated to a final AUC-PR and PR-curve. Depending on different datasets, one trajectory usually has one reference sequence and four or seven non-reference sequences.

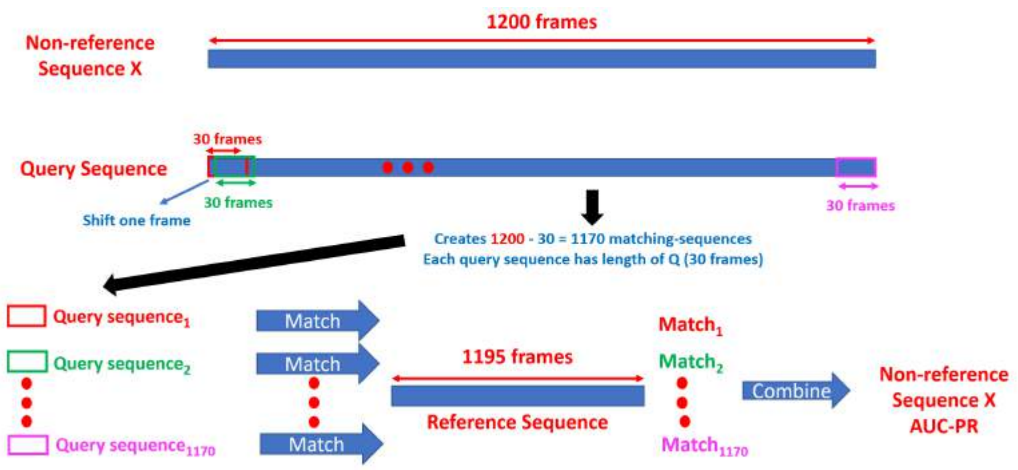

Figure 7.6: This process shows the global localization evaluation process for sequence-based matching methods. For this example, Q equals 30, R equals 1195, and NR equals 1200. The non-reference sequence X has been divided into many query sequences of length S. The global localization performs with each query sequence and the reference sequence. The resulting matches lead to X's final AUC-PR.

All the matching-time-related evaluations are performed on the Jetson AGX Xavier. For global localization, the time period starts when the first query image is received, and the time ends when a correct localization result is provided. Sequencing-based matching methods need to wait for the entire query sequence. And for a single image match, the matching-time is relatively short since it only requires one query image. Like AUC-PR, we average each non-reference sequence's average query sequences' matching-time results to form a trajectory's final matching-time.

## 7.5 Local Localization Evaluation

Instead of evaluating the entire localization system, we separate the evaluation into global and local due for the following reason. Since an incorrect global match leads to subsequent incorrect local matches, evaluating the entire system does not reveal the actual performance of the local localization. We set the non-reference sequence length to be NR and the reference sequence length to be R. Frame numbers for the non-reference sequence are set from one to NR, with an interval equal to one. The query sequence used for sequence-based matching methods has length Q.

Only single image match and Seq-SLAM are evaluated for local localization. Fast Seq-SLAM is not evaluated. During local localization, matches are performed between a matching sequence with length Q and a local reference sequence with length Q. In this case, the coarse to a fine concept does not exist. As mentioned in section 3.2, $P_{init} = \frac{R}{Q} * \tau$. With Q=R, the initial particle numbers are just $\tau$. And for these $\tau$ particles, they all have the same particle index, which makes multi-resolution sampling meaningless.

To evaluate local localization, we need to give the module the correct initial positions. For the same reason mentioned in the previous section, the index range for usable query frames is from Q to NR, with an interval equal to one. For each query frame X, the correct initial matched reference-sequence frame index Y is given at query frame X-1. The local localization first uses the query sequence with frame index ranging from X-Q+1 to X. After each local localization, Y is updated as stated in section 5.4 (here we assume an ideal case, which $I_M$=0). Then, the local localization runs again using the query sequence with frame index ranging from X-Q+2 to X+1 with the updated Y given at query frame X. For each query frame X, the local localization runs until the query frame's index increases to NR. For each non-reference sequence, there are all together (NR-(Q)+1) + (NR-(Q+1)+1) + (NR-(Q+2)+1) + .. + (NR-(NR)+1)=$\sum_{i=0}^{NR-Q} NR - Q - i + 1 = \frac{(NR-Q+2)*(NR-Q+1)}{2}$ matches. Each match includes a reference frame index Y and a match score. Y is then transferred to 0 and 1. The only difference for a single image match is that it compares the new received query frame X instead of the query sequence with the local reference sequence. We integrate each query sequence's matched results to form a trajectory's final AUC-PR.

All the matching-time related evaluation is performed on the Jetson AGX Xavier. For local localization, the time period starts when the new query image is received, and the time ends when a correct local localization result is provided. While for a single image match, the matching-time is extremely short since it only needs to make a comparison between one query image and a local reference sequence. Like AUC-PR, we average each non-reference sequence's average query sequences' matching-time results to form a trajectory's final matching-time.

# Chapter 8

# Results

The previous section explains our experiments' setups, and this section shows the experiments' results. The first section gives detailed explanations for choosing specific parameter values for Fast Seq-SLAM. The following section shows different place representation techniques performances and explains why we choose CoHOG[41]. Then, we present both quantitative and qualitative evaluations.

## 8.1 Fast Global Localization Parameters

As discussed in section 3.2, these three parameters ($\tau$, Q, and re_max) are very important to the performance of the Fast Seq-SLAM, which is the main algorithm for the global localization module in the system. To give a brief overview, $\tau$ determines the particles' density, Q represents the number of frames of the query sequence, and re_max controls the initial resolution level, which is $2^{\text{re\_max}}$.

We test with $\tau = [1, 1.33, 1.66, 2, 3, 4, 5, 6]$, which covers particle overlap rates from 0% to 80%, which is fairly comprehensive. To make this localization system usable, we cannot have a query sequence with too many frames. Q is set to be among [20,30,40,50,60] (unit frames). With 0.25 meters per frame, 10 to 15 meters is a reasonable distance for global localization. And for re_max, we test between 1 and 2. Since our choices for Q are not very large, re_max greater than 2 reduces too many frames. For example, with re_max equals 3, and Q equal to twenty. Q is only two after down-sampled with the resolution being 8, making the evaluated results meaningless.

We used CMU's indoor dataset and the collected dataset - trajectory one as our evaluation datasets. These two datasets are comprehensive, including illumination changes, directional viewpoint changes, and altitude viewpoint changes. They also have challenging components: dynamic objects and long and narrow corridors with repeated features. We test with different $\tau$ and Q combinations under different re_max. We use CoHOG[41] as our place recognition technique to set up the same environment for all the parameter values. The AUC-PR and time evaluation methods followed section 7.4.

From the results in table 8.1, table 8.2 and figure 8.1, we can have the following conclusions. re_max= 1 does not have significant accuracy improvement compared to re_max = 2, but costs more time. $\tau$ = 3.0 and Q = 40 balances best between accuracy and time in both datasets. Thus, for the Fast Seq-SLAM : $\tau$ = 3.0, Q = 40 and re_max = 2.

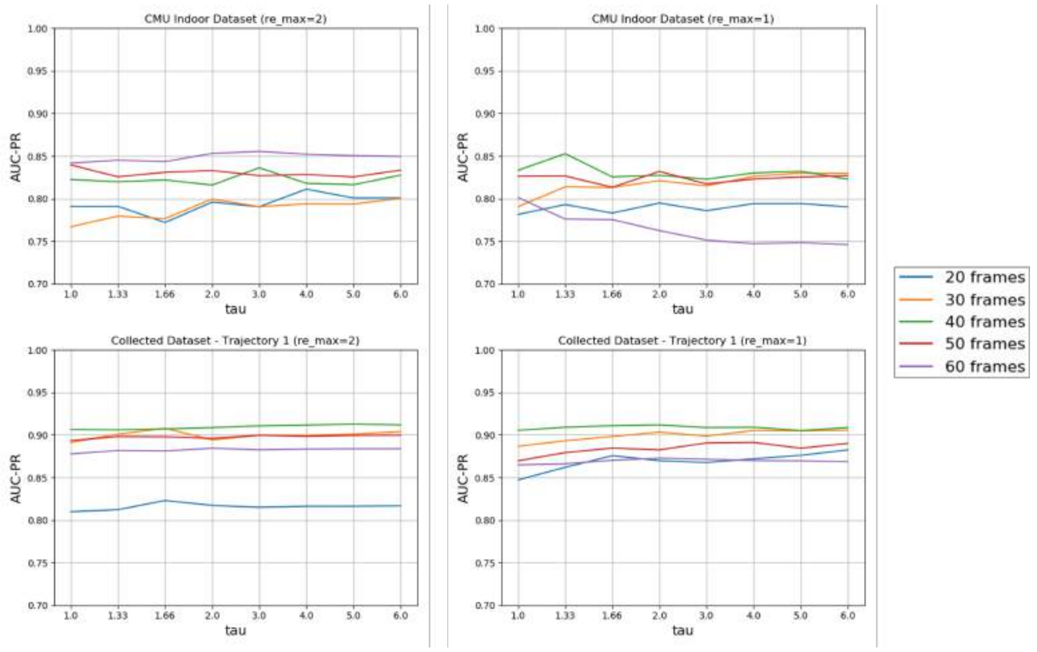

Figure 8.1: The four pictures show the AUC-PR results for two different datasets with different re_max values. The interval between each frame is around 0.25 meters. CMU indoor dataset's reference sequence has 1195 frames, around 298 meters. Collected dataset - trajectory one's reference sequence has 1783 frames, around 445 meters.

re_max=2

| | 20 | 30 | 40 | 50 | 60 |
|---|---|---|---|---|---|
| 1.0 | 9.0 | 12.1 | 16.2 | 21.0 | 25.3 |
| 1.33 | 8.5 | 12.7 | 17.1 | 21.4 | 25.7 |
| 1.66 | 8.6 | 13.2 | 17.9 | 22.7 | 26.1 |
| 2.0 | 8.5 | 13.2 | 18.6 | 22.9 | 27.9 |
| 3.0 | 9.9 | 14.4 | 19.0 | 23.5 | 29.5 |
| 4.0 | 9.6 | 14.9 | 19.4 | 24.1 | 28.5 |
| 5.0 | 10.0 | 14.4 | 20.2 | 23.0 | 30.5 |
| 6.0 | 11.3 | 14.3 | 19.5 | 24.7 | 31.1 |

re_max=1

| | 20 | 30 | 40 | 50 | 60 |
|---|---|---|---|---|---|
| 1.0 | 11.0 | 13.5 | 17.9 | 21.2 | 25.8 |
| 1.33 | 11.6 | 14.2 | 18.6 | 23.4 | 28.5 |
| 1.66 | 12.3 | 15.2 | 21.2 | 24.2 | 28.9 |
| 2.0 | 12.9 | 15.2 | 22.1 | 25.6 | 32.1 |
| 3.0 | 12.6 | 17.1 | 22.8 | 27.3 | 34.8 |
| 4.0 | 13.4 | 17.8 | 24.5 | 28.4 | 35.0 |
| 5.0 | 15.0 | 19.1 | 25.7 | 30.4 | 38.5 |
| 6.0 | 14.9 | 20.0 | 24.8 | 31.7 | 39.5 |

Table 8.1: Time(s) results with CMU indoor dataset. $\tau = [1, 1.33, 1.66, 2, 3, 4, 5, 6]$ as rows and Q = [20, 30, 40, 50, 60](frames) as columns. Reference Sequence has 1195 frames, around 298 meters.

re_max=2

| | 20 | 30 | 40 | 50 | 60 |
|---|---|---|---|---|---|
| 1.0 | 12.1 | 16.8 | 21,5 | 27.5 | 31.8 |
| 1.33 | 12.4 | 18.1 | 22.8 | 29.0 | 33.6 |
| 1.66 | 12.9 | 18.6 | 24.2 | 28.0 | 34.7 |
| 2.0 | 14.0 | 19.2 | 25.9 | 28.9 | 34.8 |
| 3.0 | 13.8 | 19.4 | 26.4 | 30.8 | 34.3 |
| 4.0 | 15.3 | 19.7 | 25.9 | 31.6 | 35.7 |
| 5.0 | 16.5 | 20 | 26.8 | 31.3 | 37.5 |
| 6.0 | 15.6 | 21.5 | 27.5 | 32.6 | 38.7 |

re_max=1

| | 20 | 30 | 40 | 50 | 60 |
|---|---|---|---|---|---|
| 1.0 | 13.6 | 17.3 | 21.7 | 27.1 | 31.8 |
| 1.33 | 14.2 | 18.1 | 23.5 | 29.4 | 35.7 |
| 1.66 | 15.4 | 19.6 | 25.7 | 31.5 | 38.5 |
| 2.0 | 15.9 | 20.8 | 26.9 | 32.8 | 41.2 |
| 3.0 | 15.7 | 23.8 | 29.9 | 36.2 | 44.4 |
| 4.0 | 16.8 | 25.1 | 31.5 | 39.3 | 47.7 |
| 5.0 | 18.9 | 26.9 | 33.8 | 40.7 | 48.9 |
| 6.0 | 20.4 | 29.3 | 34.7 | 42.7 | 52.7 |

Table 8.2: Time(s) results with collected dataset - trajectory one. $\tau = [1, 1.33, 1.66, 2, 3, 4, 5, 6]$ as rows and Q = [20, 30, 40, 50, 60](frames) as columns. Reference Sequence has 1783 frames, around 445 meters.

## 8.2 Place Representation Technique

There are many place representation techniques. Some of these have been introduced in section 2.2. With the help of [42], I can evaluate the following eight different representation techniques (CoHoG [41], HOG [11], CALC [23], NetVLAD [1], Region-VLAD [16], HybridNet [8], AMOSNet [8] and AlexNet [18]) and choose the best one for our localization system. For the learning methods above, we just used pre-trained model provided by [42].

As mentioned in the section 8.1, we used CMU's indoor dataset and the collected dataset - trajectory one as our evaluation datasets due to their comprehensiveness. Figure 8.2 and figure 8.3 show the resulting difference matrices. For each dataset, a difference matrix is created with each type of non-reference sequence and the reference sequence using the place representation techniques mentioned above.

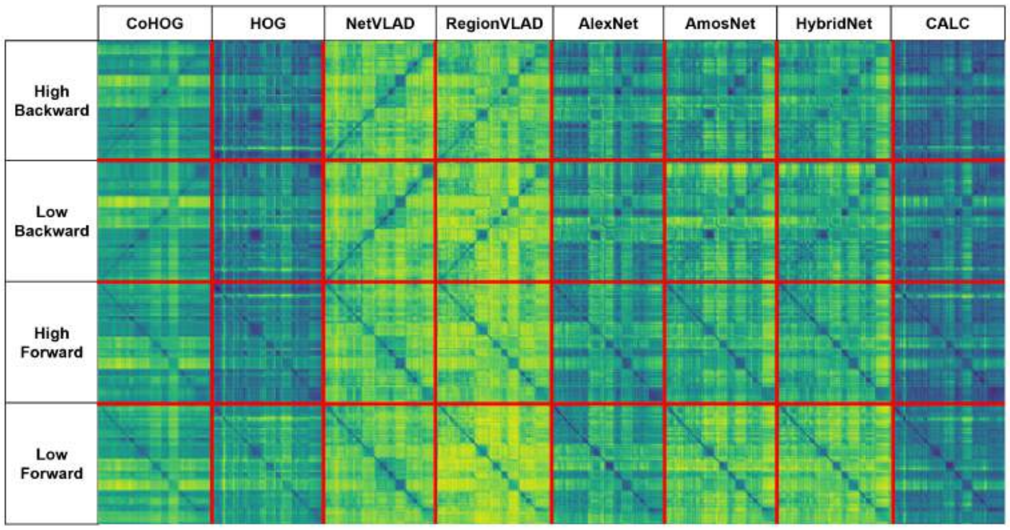

Figure 8.2: CMU's indoor dataset has four non-reference sequences' types: High Backward, Low Backward, High Forward, and Low Forward, which function as rows. Each column is a place representation technique. Each cell is the difference matrix between its row's non-reference sequences' type and the reference sequence. The reference sequence has type low forward, 1195 frames, around 298.75 meters. Details about these query types can be found in section 7.2 and figure 7.4.

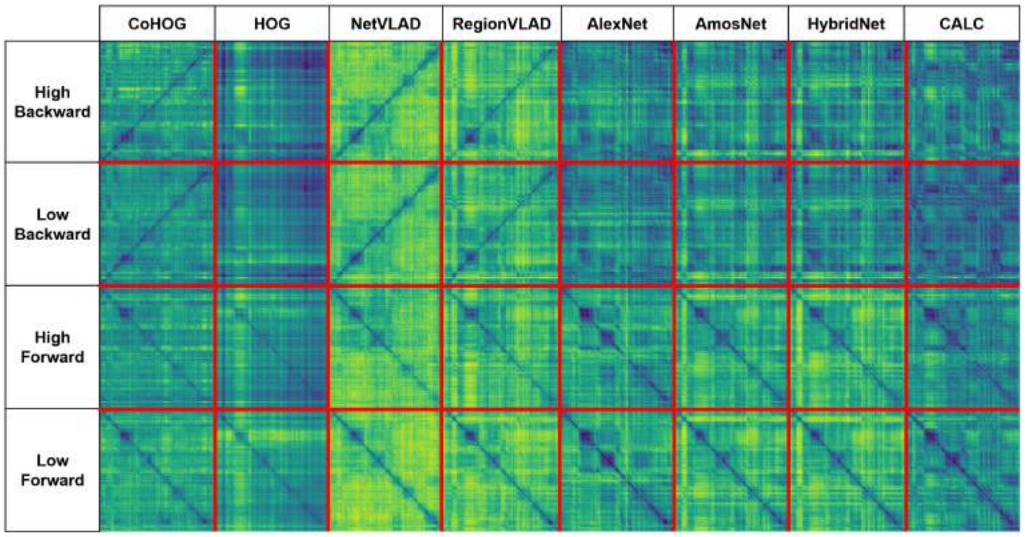

Figure 8.3: Collected datasets' trajectory one has four non-reference sequences' types: Day Forward, Night Forward, Day Backward, and Night Backward, which function as rows. Each column is a place representation technique. Each cell is the difference matrix between its row's non-reference sequences' type and the reference sequence. The reference sequence has the type Day Forward, 1783 frames, around 445 meters. Details about these query types can be found in section 6.1 and figure 6.2.

As one can see from these two graphs, HOG [11], CALC [23], HybridNet [8], AMOSNet [8] and AlexNet [18] cannot handle viewpoint changes (between forward and backward). Thus, we do not consider to use these methods.

For the rest methods : CoHoG [41], NetVLAD [1], RegionVLAD [16], we evaluate their performances for global localization with three different methods and local localization with two different methods. This evaluation shows two points: First, we can show the theory' correctness and generality presented in 7.3 using different place representation techniques. Second, we can determine the place representation technique most suitable for our localization pipeline.

Table 8.3 and table 8.4 show the AUC-PR with CoHOG, NetVLAD and Region-VLAD on both datasets. Table 8.5 and table 8.6 show the matching-time. AUC-PR and Time are measured followed methods in section 7.4 and section 7.5. The PR-Curves behind the AUC-PR can be found in figure 8.4 and figure 8.5. The PR-Curve shows that Seq-SLAM has the highest precision when recall equals one, followed by Fast Seq-SLAM. Single match has a relatively low precision value when recall equals one compared to sequence-based methods.

From these tables, we can see the following patterns.
AUC-PR: Seq-SLAM > Fast Seq-SLAM > Single Match
Match time : Seq-SLAM > Fast Seq-SLAM >> Single Match
Our theory in section 7.3 is proved regardless of representation techniques. And using Fast Seq-SLAM for global localization and Seq-SLAM for localization is the right choice based on the evaluation results. To balance time and accuracy, we choose CoHoG [41] to be our place recognition technique.

CMU Indoor Dataset : AUC-PR

| Global Localization | CoHOG | NetVLAD | RegionVLAD |
|---|---|---|---|
| Single Match | 0.84 | 0.93 | 0.87 |
| Fast Seq-SLAM | 0.87 | 0.92 | 0.89 |
| Seq-SLAM | 0.97 | 0.96 | 0.97 |
| Local Localization | CoHOG | NetVLAD | RegionVLAD |
| Single Match | 0.29 | 0.68 | 0.23 |
| Seq-SLAM | 0.88 | 0.93 | 0.85 |

Table 8.3: These tables show the AUC-PR results on CMU indoor dataset. Each row represents different matching method. The top parts show results for global localization and bottom parts show results for local localization. The reference sequence has 1195 frames, around 298.75 meters.

Collected Dataset - Trajectory 1 : AUC-PR

| Global Localization | CoHOG | NetVLAD | RegionVLAD |
|---|---|---|---|
| Single Match | 0.78 | 0.91 | 0.73 |
| Fast Seq-SLAM | 0.93 | 0.93 | 0.94 |
| Seq-SLAM | 0.93 | 0.94 | 0.94 |
| Local Localization | CoHOG | NetVLAD | RegionVLAD |
| Single Match | 0.12 | 0.28 | 0.12 |
| Seq-SLAM | 0.92 | 0.93 | 0.93 |

Table 8.4: These tables show the AUC-PR results on collected dataset - trajectory one. Each row represents different matching method. The top parts show results for global localization and bottom parts show results for local localization. The reference sequence has 1783 frames, around 445 meters.

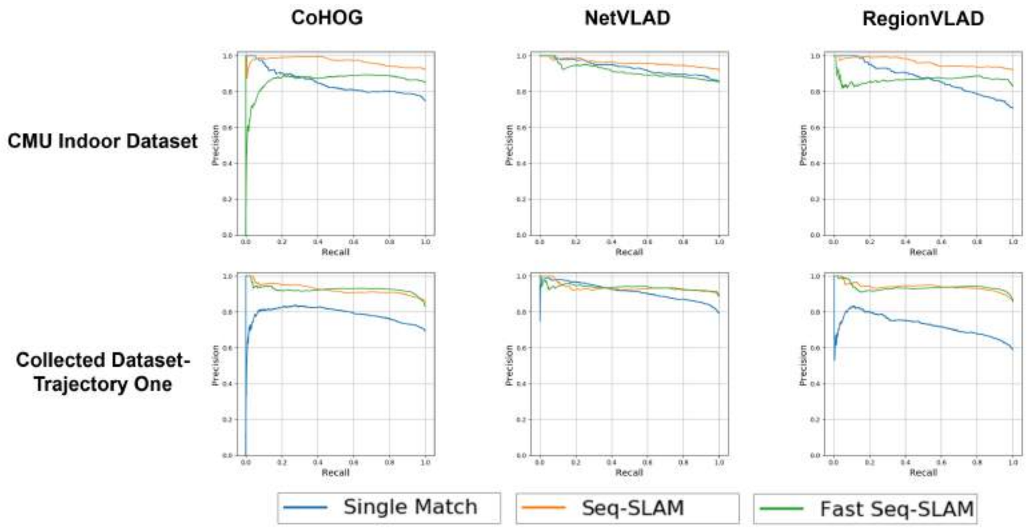

Figure 8.4: The Precision-Recall curves behind table 8.3 and table 8.4's AUC-PR results related to global localization for both datasets.

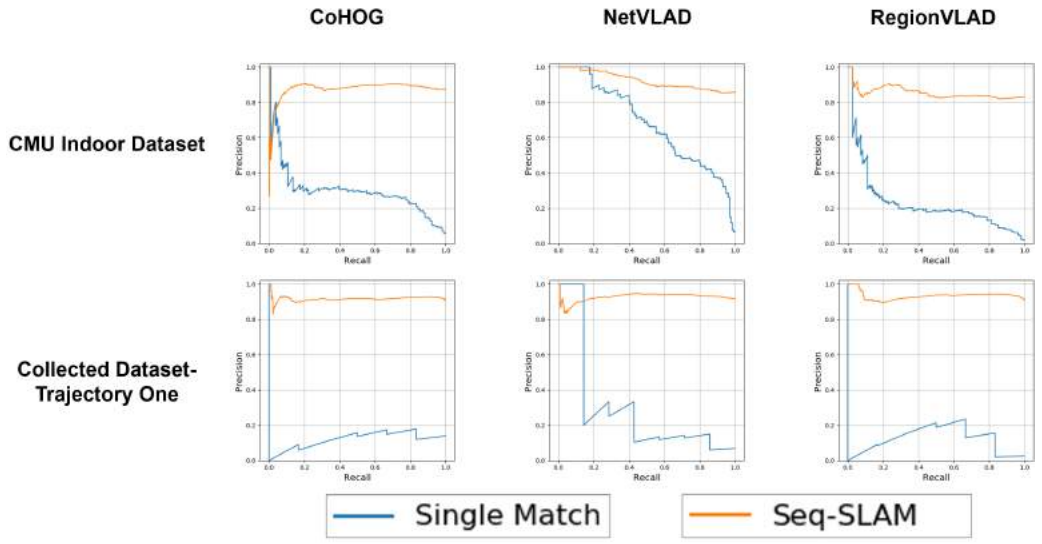

Figure 8.5: The Precision-Recall curves behind table 8.3 and table 8.4's AUC-PR results related to local localization for both datasets.

CMU Indoor Dataset : Time(s)

| Global Localization | CoHOG | NetVLAD | RegionVLAD |
|---|---|---|---|
| Single Match | 0.13 | 3.21 | 3.13 |
| Fast Seq-SLAM | 19.05 | 121.16 | 91.63 |
| Seq-SLAM | 33.15 | 176.17 | 160.62 |
| **Local Localization** | **CoHOG** | **NetVLAD** | **RegionVLAD** |
| Single Match | $0.04*10^{-1}$ | $0.04*10^{-2}$ | $0.03*10^{-1}$ |
| Seq-SLAM | 0.20 | 0.31 | 0.24 |

Table 8.5: These tables show the matching-time(s) results on CMU indoor dataset. Each row represents different matching method. The top parts show results for global localization and bottom parts show results for local localization. The reference sequence has 1195 frames, around 298.75 meters.

Collected Dataset - Trajectory 1 : Time(s)

| Global Localization | CoHOG | NetVLAD | RegionVLAD |
|---|---|---|---|
| Single Match | 0.21 | 3.52 | 3.34 |
| Fast Seq-SLAM | 26.46 | 130.82 | 97.48 |
| Seq-SLAM | 36.52 | 184.37 | 167.55 |
| **Local Localization** | **CoHOG** | **NetVLAD** | **RegionVLAD** |
| Single Match | $0.04*10^{-1}$ | $0.04*10^{-2}$ | $0.03*10^{-1}$ |
| Seq-SLAM | 0.19 | 0.35 | 0.27 |

Table 8.6: These tables show the matching-time(s) results on collected dataset - trajectory one. Each row represents different matching method. The top parts show results for global localization and bottom parts show results for local localization. The reference sequence has 1783 frames, around 445 meters.

## 8.3 Quantitative Results

Quantitative experiments are necessary to evaluate the localization system's different modules. This section presents the quantitative evaluation results. The first subsection explains the datasets used for evaluation. The following two subsections show the evaluation results of the global and local localization modules. The last section displays the system's ability to detect off-path.

### 8.3.1 Evaluation Environment

Carnegie Mellon University and Hawkins Indoor Datasets (section 7.2) are used. They include confined spaces, long narrow corridors with repeated features, drastic illumination changes, viewpoint changes, and altitude changes. We also use the collected dataset with all ten trajectories, which have 80 sequences. They include dynamic objects, illumination changes, viewpoint changes, and outdoor features.

Each non-reference and reference sequence's frames are generated with intervals of 0.25 meters. The ground truth data is obtained by Interactive SLAM[17] and LOAM[43] (Details mentioned in chapter 6).

### 8.3.2 Global Localization

We evaluate the three place matching techniques for global localization using the methods mentioned in section 7.4. The following table 8.7 shows the AUC-PR for each trajectory. For all the sequences, CoHOG is used as the place representation technique. PR-Curves behind the AUC-PR can be found in figure 8.6.

From table 8.7, single match performs the worst on average. Due to the very challenging environments, we can see that single match performs poorly compared to sequence-based methods in the Hawkins dataset. Fast Seq-SLAM does not perform as excellent as Seq-SLAM in more challenging environments(CMU and Hawkins Indoor datasets). With outdoor datasets, it performs as well as Seq-SLAM. From figure 8.6, we can see that Seq-SLAM has the highest precision values for most trajectories when recall equals one. Fast Seq-SLAM has lower precision values than Seq-SLAM when recall equals one but is still acceptable. And single match has the lowest precision values when recall equals one for all datasets.

We also evaluate the matching-time for global localization using the methods stated in section 7.4. Due to their comprehensiveness, we used CMU indoor dataset and the collected dataset - trajectory one as our evaluation datasets. For matching-time performance, the number of frames of the reference sequence is the key factor. We want to evaluate different methods' performance with reference sequences containing different numbers of frames. A reference sequence with different numbers of frames is created with different frame intervals. We cut some frames to round up the number of frames. We set the number of frames to be [500, 1000, 2000, 3000, 4000, 8000].

With frame intervals being 0.25 (our standard setup mentioned in 5.1), 500 and 1000 frames would be a small map, and 2000, 3000, and 4000 frames make a normal map. 8000 frames make a large map, which can compare different algorithms' performance in extreme conditions.

Global Localization : AUC-PR

| | CMU | Hawkins | O1 | O2 | O3 | O4 |
|---|---|---|---|---|---|---|
| Single Match | 0.85 | 0.66 | 0.78 | 0.87 | 0.87 | 0.78 |
| Fast Seq-SLAM | 0.86 | 0.92 | 0.93 | 0.90 | 0.97 | 0.97 |
| Seq-SLAM | 0.97 | 0.95 | 0.93 | 0.88 | 0.99 | 0.98 |
| Reference Sequence's Frames | 1200 | 341 | 1943 | 1412 | 2142 | 2940 |
| Reference Sequence's Distance(m) | 304 | 86 | 487 | 354 | 537 | 738 |
| | O5 | O6 | O7 | O8 | O9 | O10 |
| Single Match | 0.90 | 0.94 | 0.89 | 0.87 | 0.88 | 0.84 |
| Fast Seq-SLAM | 0.96 | 0.98 | 0.97 | 0.91 | 0.87 | 0.85 |
| Seq-SLAM | 0.95 | 0.98 | 0.97 | 0.93 | 0.84 | 0.84 |
| Reference Sequence's Frames | 2240 | 2248 | 1588 | 769 | 1302 | 1180 |
| Reference Sequence's Distance(m) | 561 | 563 | 390 | 193 | 326 | 295 |

Table 8.7: This table shows the AUC-PR for each algorithm in each trajectory for global localization. CMU indoor dataset is abbreviated as CMU and Hawkins indoor dataset is abbreviated to Hawkins. Collected dataset - trajectory one is abbreviated to be O1 and two be O2 and etc.

CMU Indoor Dataset : Times(s)

| Number of Reference Frames | 500 | 1000 | 2000 | 3000 | 4000 | 8000 |
|---|---|---|---|---|---|---|
| Single Match | 0.21 | 0.29 | 0.40 | 0.51 | 0.64 | 1.10 |
| Fast Seq-SLAM | 14.04 | 17.09 | 22.80 | 41.25 | 59.42 | 90.53 |
| Seq-SLAM | 20.8 | 54.3 | 188.1 | 411.3 | 752.2 | 4687.39 |

Table 8.8: This table shows the matching-time(s) for global localization with three different matching algorithms. They are evaluated under reference sequences with different numbers of frames on Xavier using CMU indoor dataset.

Collected Dataset - Trajectory 1 : Times(s)

| Number of Reference Frames | 500 | 1000 | 2000 | 3000 | 4000 | 8000 |
|---|---|---|---|---|---|---|
| Single Match | 0.22 | 0.29 | 0.41 | 0.50 | 0.63 | 1.12 |
| Fast Seq-SLAM | 15.13 | 20.12 | 27.18 | 41.80 | 49.52 | 107.22 |
| Seq-SLAM | 22.73 | 57.51 | 187.51 | 411.09 | 750.97 | 4703.45 |

Table 8.9: This table shows the matching-time(s) for global localization with three different matching algorithms. They are evaluated under reference sequences with different numbers of frames on Xavier using collected dataset - trajectory one.

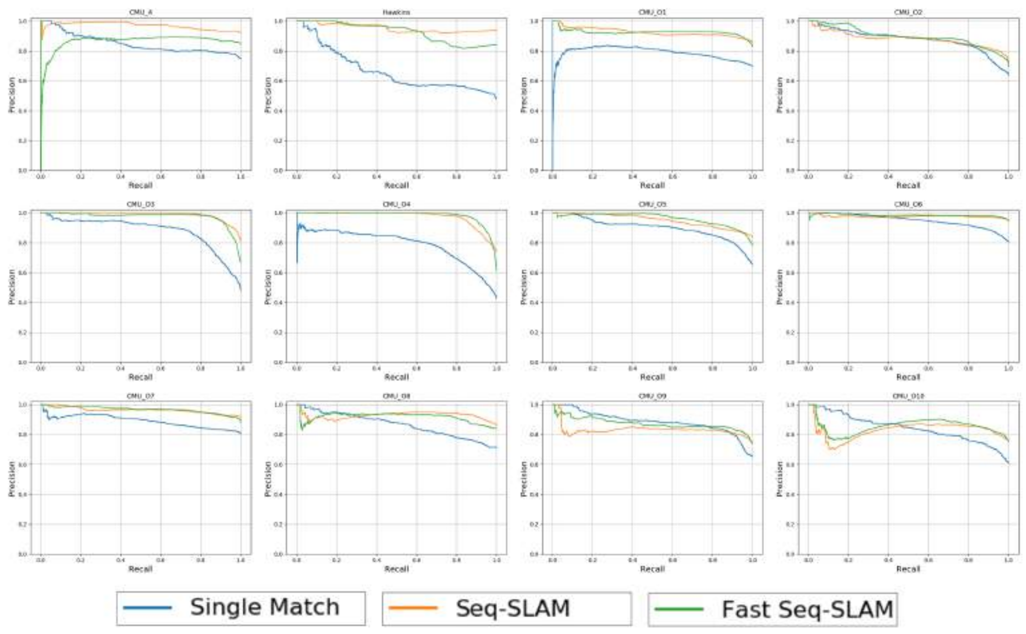

Figure 8.6: The Precision-Recall curves behind table 8.7's AUC-PR results for all 12 trajectories.

Table 8.8 and table 8.9 shows the run-time performance for global localization. Single match is undoubtedly the fastest since it does not need to store frames for the query sequence. Fast Seq-SLAM is roughly two times faster than Seq-SLAM with small and normal maps, and the speed boost is even larger when the map's size becomes larger. It also shows fast Seq-SLAM balances time and accuracy best among these three different place matching techniques. The result is relatively stable within different datasets, proving this choice's generality. Overall, The evaluation results matched our theory and proved our choice to use Fast Seq-SLAM for global localization is correct.

### 8.3.3 Local Localization

We evaluate two place matching techniques for local localization using the mentioned methods in section 7.5. The following table 8.10 shows the AUC-PR for each trajectory. For all the sequences, CoHOG is used as the place representation technique. PR-Curves behind the AUC-PR can be found in figure 8.7.

From table 8.10, we can see that single-match performs as well as Seq-SLAM in local localization. We also evaluated the local localization's matching time on Xavier. The evaluation method is the same as the global localization matching time's evaluation.

Local Localization : AUC-PR

| | CMU | Hawkins | O1 | O2 | O3 | O4 |
|---|---|---|---|---|---|---|
| Single Match | 0.40 | 0.31 | 0.13 | 0.35 | 0.39 | 0.34 |
| Seq-SLAM | 0.87 | 0.97 | 0.92 | 0.87 | 0.96 | 0.98 |
| Reference Sequence's Frames | 1200 | 341 | 1943 | 1412 | 2142 | 2940 |
| Reference Sequence's Distance(m) | 304 | 86 | 487 | 354 | 537 | 738 |
| | O5 | O6 | O7 | O8 | O9 | O10 |
| Single Match | 0.41 | 0.48 | 0.32 | 0.50 | 0.50 | 0.41 |
| Seq-SLAM | 0.94 | 0.98 | 0.98 | 0.96 | 0.83 | 0.81 |
| Reference Sequence's Frames | 2240 | 2248 | 1588 | 769 | 1302 | 1180 |
| Reference Sequence's Distance(m) | 561 | 563 | 390 | 193 | 326 | 295 |

Table 8.10: This table shows the AUC-PR for each algorithm in each trajectory for local localization. CMU indoor dataset is abbreviated as CMU and Hawkins indoor dataset is abbreviated to Hawkins. Collected dataset- trajectory one is abbreviated to be O1 and two be O2 and etc.

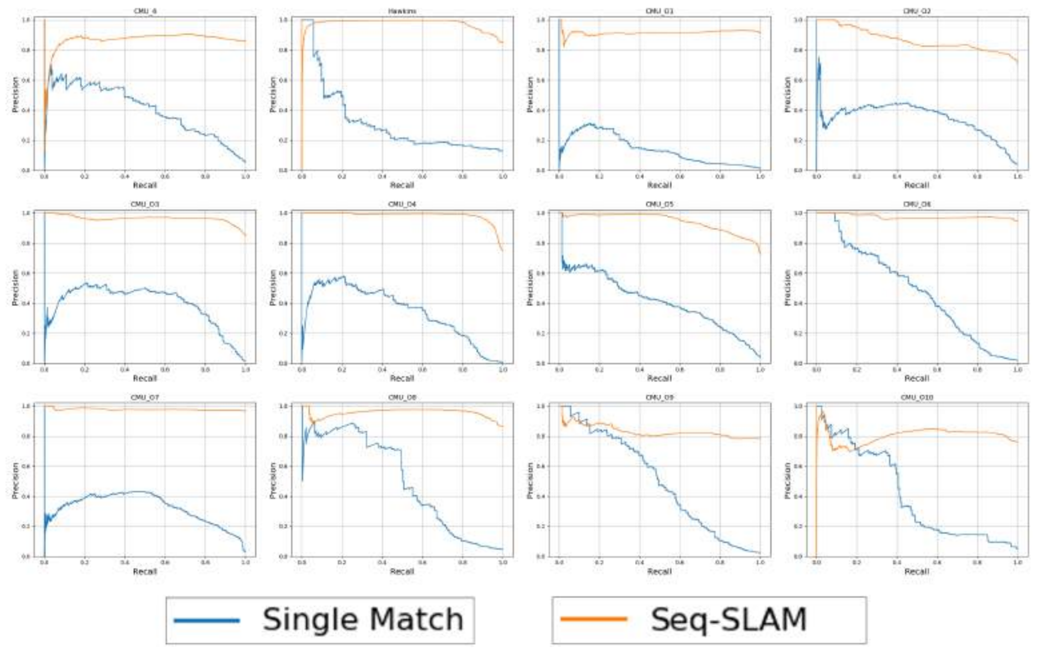

Figure 8.7: The Precision-Recall curves behind table 8.10's AUC-PR results related for all 12 trajectories

CMU Indoor Dataset : Times(s)

| Number of Reference Frames | 500 | 1000 | 2000 | 3000 | 4000 | 8000 |
|---|---|---|---|---|---|---|
| Single Match ($*10^{-1}$) | 0.04 | 0.05 | 0.05 | 0.04 | 0.05 | 0.05 |
| Seq-SLAM | 0.32 | 0.33 | 0.33 | 0.34 | 0.33 | 0.34 |

Table 8.11: This table shows the matching-time(s) for local localization with two different matching algorithms. They are evaluated under reference sequences with different numbers of frames on Xavier using CMU indoor dataset.

Collected Dataset - Trajectory 1 : Times(s)

| Number of Reference Frames | 500 | 1000 | 2000 | 3000 | 4000 | 8000 |
|---|---|---|---|---|---|---|
| Single Match ($*10^{-1}$) | 0.05 | 0.03 | 0.05 | 0.05 | 0.05 | 0.04 |
| Seq-SLAM | 0.34 | 0.32 | 0.33 | 0.33 | 0.34 | 0.34 |

Table 8.12: This table shows the matching-time(s) for local localization with two different matching algorithms. They are evaluated under reference sequences with different numbers of frames on Xavier using collected dataset - trajectory one.

Table 8.11 and table 8.12 show the run-time performances for local localization on both datasets. The difference between single image and sequence matches is less obvious than global localization since the sequence match only needs to perform with local reference sequences. By only matching with local reference sequence instead of global, local localization's times for both methods are relatively stable with reference sequences with different numbers of frames. This way, the system can provide stable local localization matches at around 3HZ regardless of the map size and environmental conditions. Seq-SLAM has high AUC-PR, is robust to environmental changes, and can provide a 3HZ matching frequency. Thus, it is correct to choose Seq-SLAM for the local localization module. The evaluation result is relatively stable within different datasets, proving this choice's generality.

### 8.3.4 Off-Path Detection

Our proposed localization system can do off-path detection and agents re-localization. Thus, evaluating the global and local localization module's ability to discriminate between visited and new places is necessary. We used the same methods mentioned in section 7.4 and section 7.5 to evaluate the differentiation ability. The only difference is that we use the ROC metric (section 7.1.2) instead of the PR metric, which considers true negatives.

As mentioned in the section 8.1, we used CMU indoor dataset and the collected dataset - trajectory one as our evaluation datasets due to their comprehensiveness. We modified these two datasets so that each non-reference sequence has the same number of new and visited places. The number of true-positives and true-negatives is kept equal because ROC curves work well for balanced classification problems [42].

Figure 8.8 shows the ROC curves for global and local localization under two modified datasets. As one can see from the graph, Fast Seq-SLAM has a very good class separation capacity during global localization, and Seq-SLAM also has a very good class separation capacity during local localization. Thus, it is reasonable for this pipeline to detect off-path and re-localize based on the evaluation results. Detailed AUC-ROC values can be seen from table 8.13.

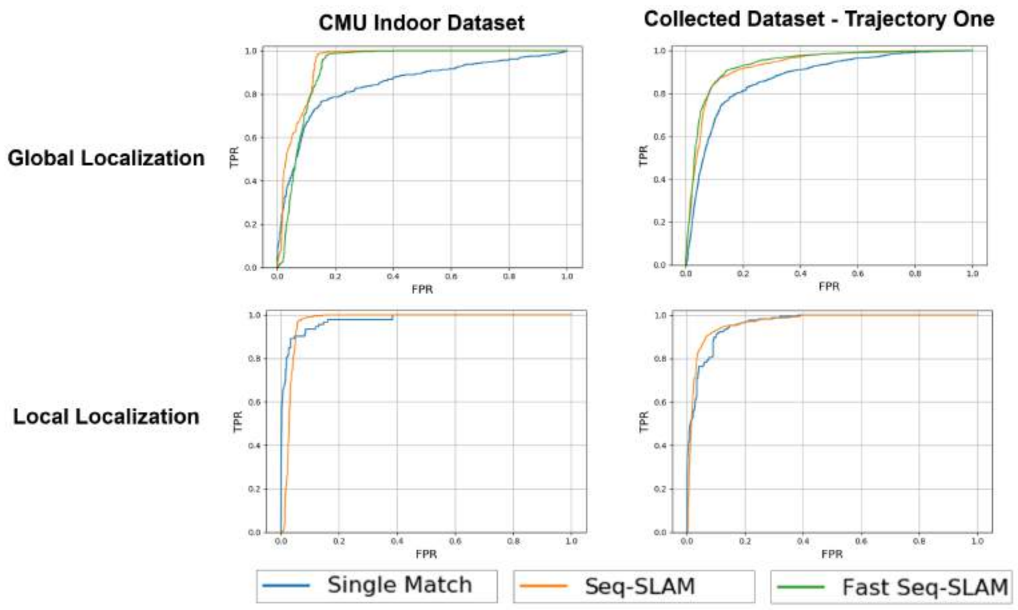

Figure 8.8: The ROC curves for both global and local localization under two modified datasets. The top row is the evaluation results for off-path detection on global localization module and the bottom row shows the results for off-path detection on local localization module.

AUC-ROC

| Global Localization | CMU Indoor Dataset | Collected Dataset - Trajectory One |
|---|---|---|
| Single Match | 0.89 | 0.87 |
| Fast Seq-SLAM | 0.92 | 0.94 |
| Seq-SLAM | 0.94 | 0.93 |
| Local Localization | CMU Indoor Dataset | Collected Dataset - Trajectory One |
| Single Match | 0.97 | 0.96 |
| Seq-SLAM | 0.96 | 0.96 |

Table 8.13: The AUC-ROC values of the ROC curves presented in figure 8.8.

## 8.4 Qualitative Results

We state that this localization pipeline works in real-time with portable devices. It is necessary to conduct a real-time demo. We collected a map at Carnegie Mellon

University, including both indoor and outdoor features. Section 8.4.1 explains more details about the environment and the query frames' types we used for qualitative evaluation. Section 8.4.2 shows qualitative results without off-path conditions. Section 8.4.3 shows a qualitative result with off-path conditions.

### 8.4.1 Demo Environment

The reference map is collected at Carnegie Mellon University, including indoor and outdoor features. It has 1190 frames and 478 meters, with a frame interval being 0.25 meters. Figure 8.9 shows the map overview with trajectory generated by LiDAR Odometry and point cloud data. The reference map is collected in the day-light in the forward direction.

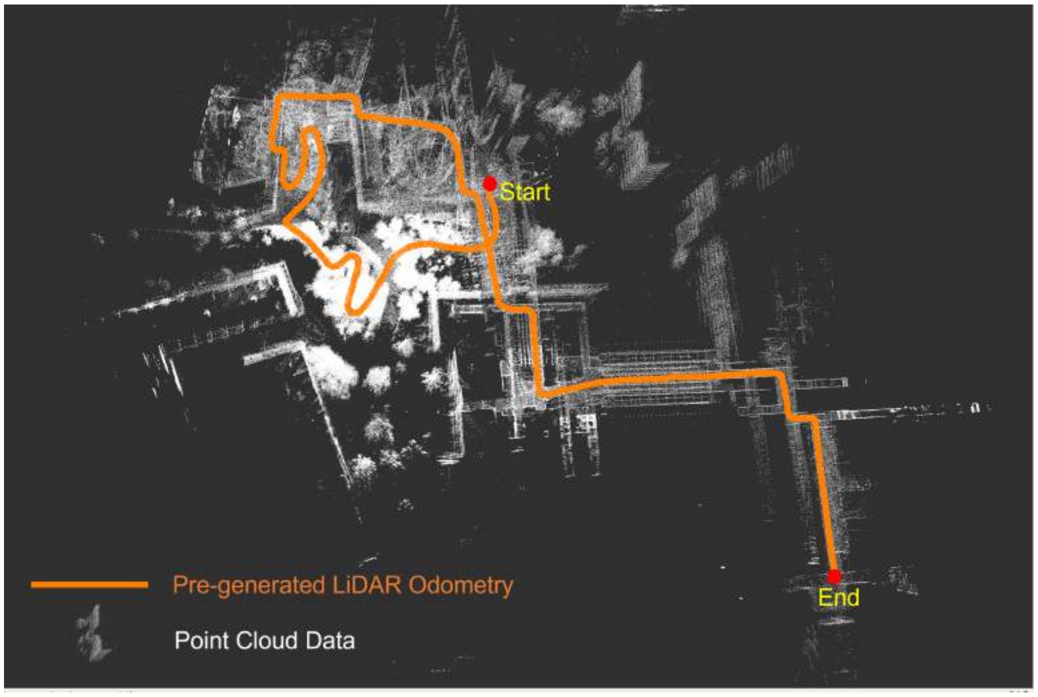

Figure 8.9: This is the collected reference sequence(map). The orange path is the LiDAR Odometry, and the white points are the point cloud data. The reference map's start and end points are marked in red dots. The forward direction is from start to end, and the backward direction otherwise. The reference map is collected in the daytime in the forward direction.

Figure 8.10 shows the query frames' types used in the following sections. Types include day-light with forward direction, day-light with backward direction, and dusk-light with backward direction. All query sequences have altitude changes since it is performed with the localization helmet. (mentioned in section 5.3).The quality of the stitched image and the Ricoh Theta V's omnidirectional image also have some differences, making the localization more challenging. The query image has a resolution of 256*128, and the reference image has a resolution of 1024*512. The map must be high-quality with high-resolution images for any query sequence types. The query image can only have low resolution due to real-time and portable device requirements.

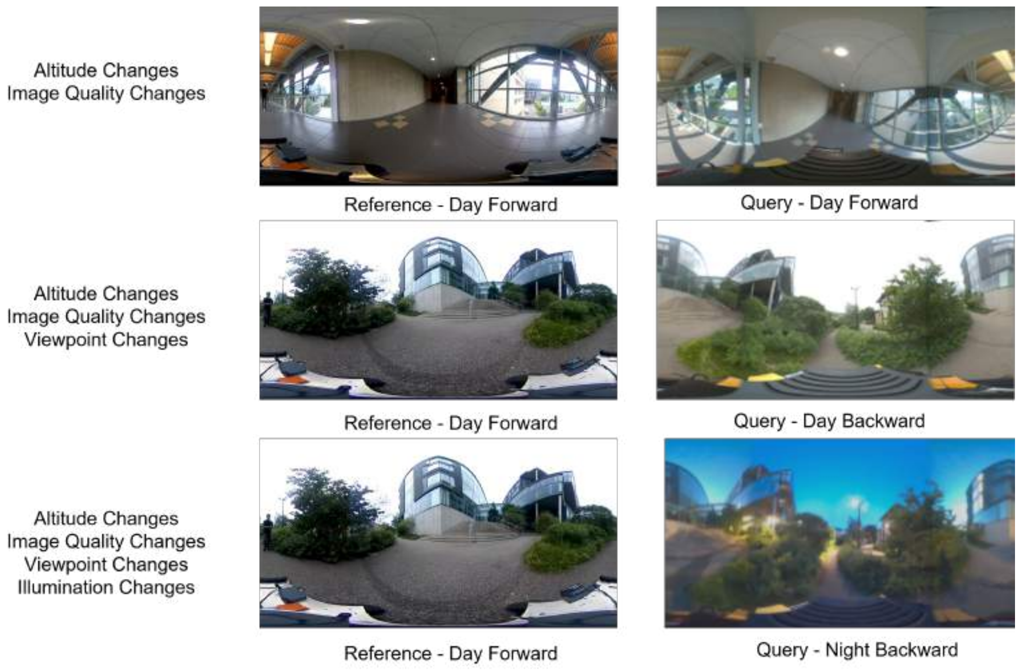

Figure 8.10: First row shows the the altitude and image quality changes between reference image from mapping robot and query image from the localization helmet. Second rows show adds viewpoint changes. Third row adds illumination changes.

## 8.4.2 Non off-path Results

This section shows the qualitative evaluation results of the localization system without off-path conditions.

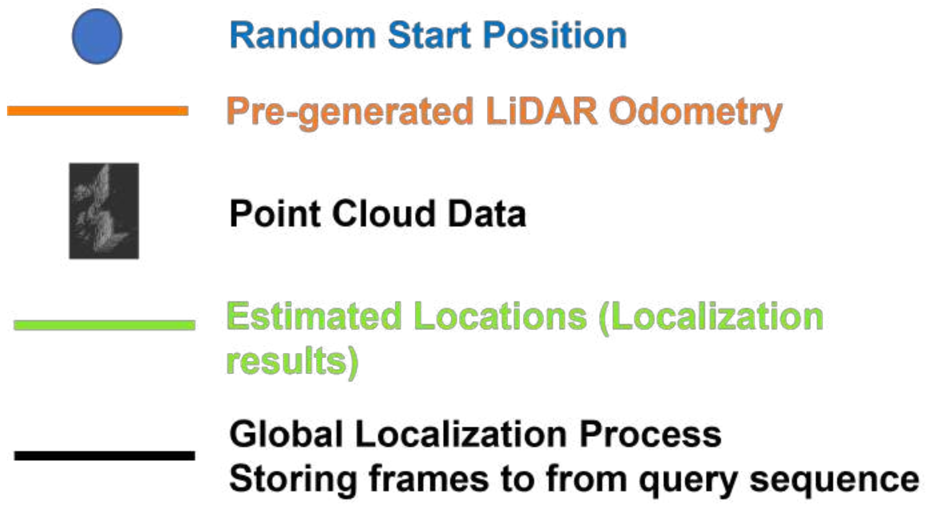
Random Start Position
Pre-generated LiDAR Odometry
Point Cloud Data
Estimated Locations (Localization results)
Global Localization Process
Storing frames to from query sequence

Figure 8.11: This figure explains different symbols used in this chapter's rest figures. The agent's start position is marked in a blue circle on the map. The orange line is the map trajectory, the white points are point cloud data, and the green line is the estimated localization results of the agent. As said in chapter 5, the agent needs to walk a certain distance to store frames for the query sequence for the global localization. This process happens on the path marked in black, which is between the blue start point and the green estimated localization results.

Figure 8.12 shows the qualitative evaluation results, in which the query images are received in a forward direction and day-light. The only differences between the query image and the reference image are the altitude and pictures' qualities.

Figure 8.13 shows the qualitative evaluation results, in which the query images are received in a backward direction and day-light. The differences between the query image and the reference image are the viewpoint, altitude, and pictures' qualities.

Figure 8.14 shows the qualitative evaluation results, in which the query images are received in a backward direction and dusk-light. The differences between the query sequence and the reference image are the viewpoint, illumination, altitude, and pictures' qualities. As one can see, the system's performance decreases given a very challenging and changing environment. Our system eventually succeeded in localizing most of the agent's positions.

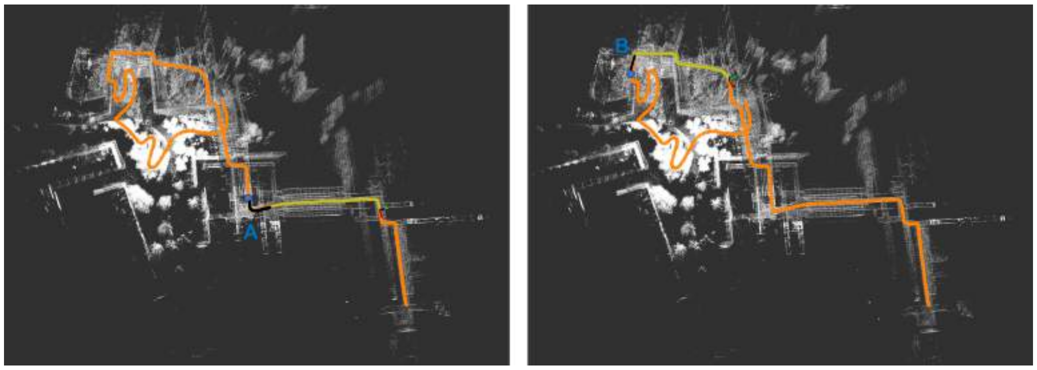

Figure 8.12: Figure shows the qualitative localization results. The left picture shows a localization process that starts from a random location A. The right picture shows the process starts from a random location B. Figure 8.11 explains components on the graph. The entire demo video can be seen at video links.

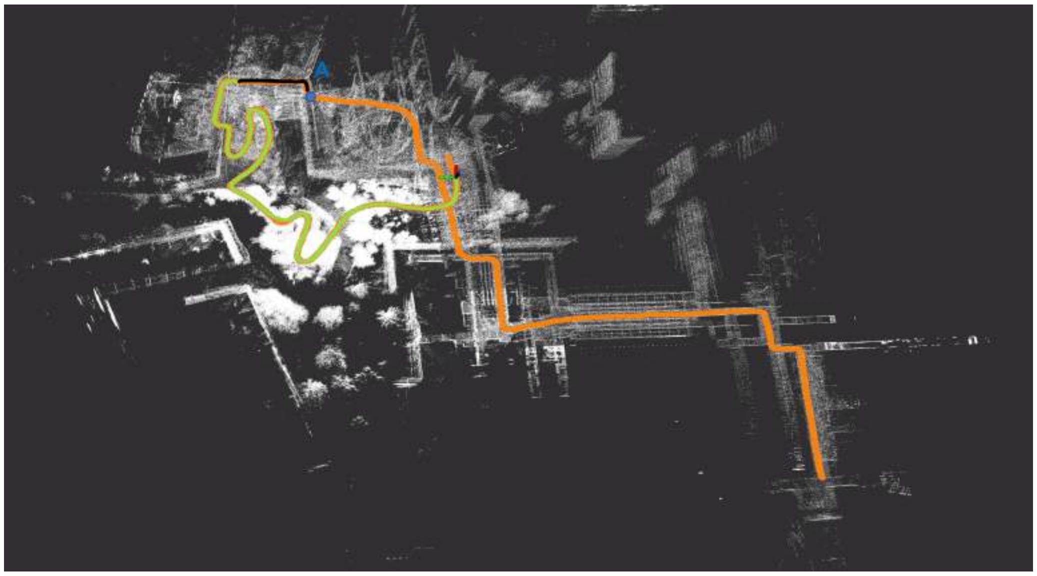

Figure 8.13: Figure shows the qualitative localization results. The picture shows a localization process that starts from a random location A. Figure 8.11 explains components on the graph. The entire demo video can be seen at video links.

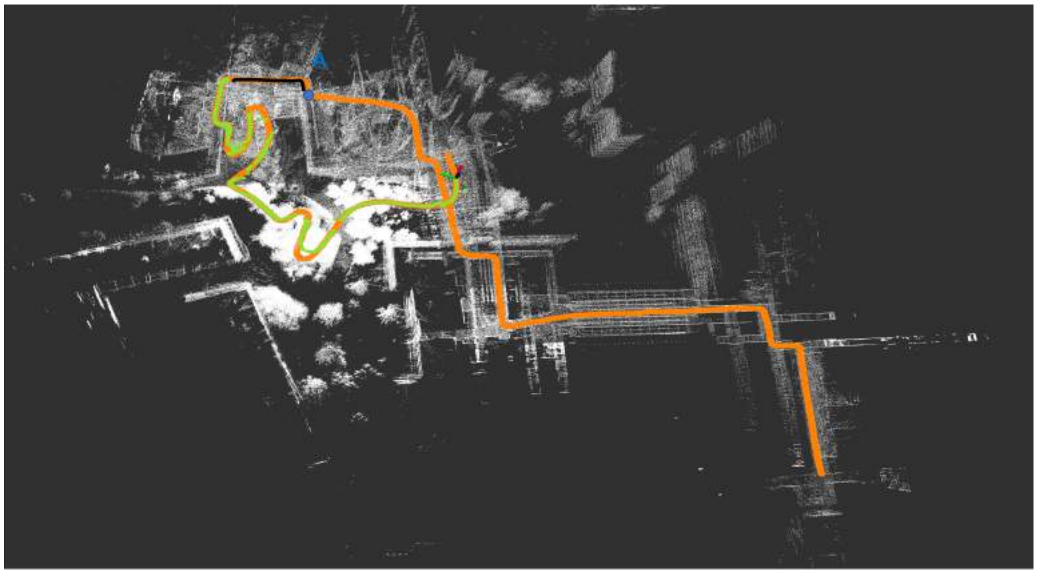

Figure 8.14: Figure shows the qualitative localization results. The picture shows a localization process that starts from a random location A. Figure 8.11 explains components on the graph. The entire demo video can be seen at video links.

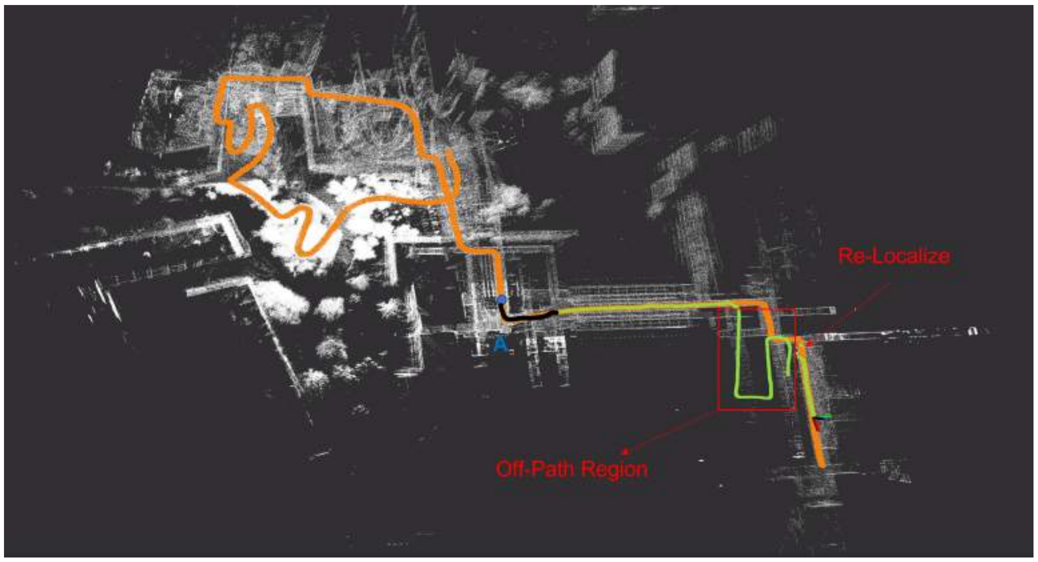

Figure 8.15: Figure shows the qualitative localization results. The picture shows a localization process that starts from a random location A. Figure 8.11 explains components on the graph. The off path region is marked in a red box. The entire demo video can be seen at video links.

### 8.4.3 Off-path Results

The previous sections show the qualitative evaluations without off-path conditions. The current section displays the qualitative evaluation with off-path conditions.
Figure 8.15 shows the qualitative evaluation results, in which the query images are received in a forward direction and day-light. The only differences between the query and the reference images are the altitude and pictures' qualities. As shown in figure 8.15, the agent walks into the area marked in the red box, which is not on the map. So the system runs purely on odometry and waits for the agent to return to the path again. When the agent is back on the path, the red arrow indicates the location where the system re-localizes the agent on the map.

# Chapter 9

# Conclusions and Future Works

## 9.1 Conclusions

In this thesis, we introduced a real-time localization system that takes works robustly and efficiently in changing and challenging environments.We first presented the motivation to use the camera-based visual place recognition method instead of Visual-Inertial and LiDAR-based SLAM systems. Then, We introduced the main research focus in the field of VPR and the current research gap. We explained why developing an efficient and robust localization system is necessary. We built our mapping rover robot and the portable localization device with omnidirectional cameras to achieve our pipeline. The proposed system contains a global localization module using Fast Seq-SLAM, a local localization module using Seq-SLAM, and an off-path detection module. The system also integrates odometry fusion for more robust performances. We perform quantitative and qualitative evaluations on our system using the collected datasets in changing and challenging environments. Our global localization module can achieve 0.925 AUC-PR, and the localization time is around 20 seconds on a 400-meters map. And our local localization module can output the current location's estimation at 3HZ regardless of the size of the map. The local localization module reaches 0.922 AUC-PR. To perform off-path detection, both global and local localization modules must be able to differentiate between visited and new places. Global and Localization modules achieve 0.93 and 0.96 AUC-ROC, respectively. Qualitative experiments are also performed at Carnegie Mellon University with the self-built mapping robot and localization helmet, which is provided with screenshots and videos. We observed that our algorithm choices for global localization, local localization, and off-path modules are correct from the quantitative and qualitative evaluation results.

## 9.2 Future Works

### 9.2.1 Localization Device

The current localization device is too big and heavy. Our team will develop and design a new device with a smaller size and better computation power. Furthermore, it can easily be attached to an agent, a robot, or a drone. In this way, this localization system can be used in more scenarios. Figure 9.1 shows the prototype of the new localization device.

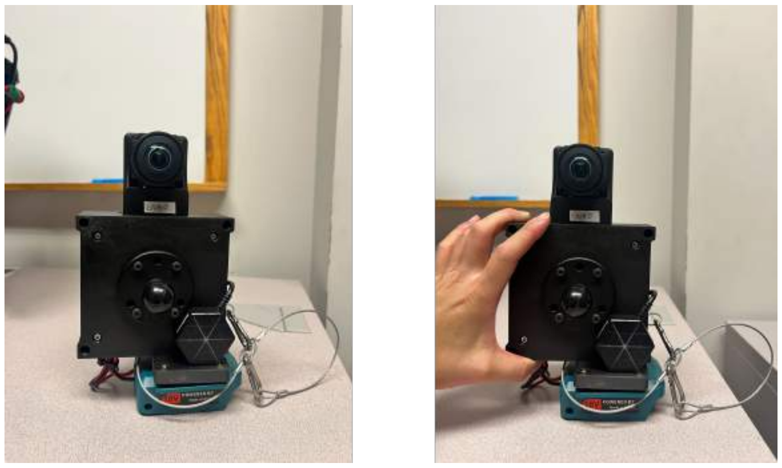

Figure 9.1: This is a newly designed localization device. It is smaller, more robust, and with higher computer power with a new version of Xavier.

### 9.2.2 Complicated Map

The current localization system only works on a one-way path. This pipeline cannot localize agents on a map with intersections. However, in the real world, most maps have intersections. In most cases, an agent can have multiple paths to move from point A to point B. With these complex maps, the ideal matches are not always at the diagonal line. Figure 9.2 shows the comparison between a simple map and a complex map.

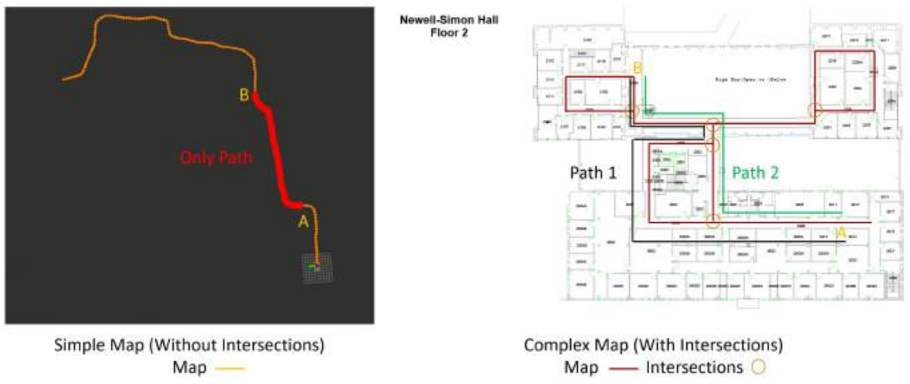

Figure 9.2: The picture on the left is a simple one-way map. For any two locations A and B on the map, the agent can only move in one path, marked in red. However, for the picture on the right with intersections, the agent can have multiple ways (black and green paths, there should be more paths) to move from A to B on the Map.

Furthermore, the current pipeline cannot localize an agent with multi-direction paths. However, it's also common for an agent or a robot to move back and forth on the map. It would be exciting while challenging to upgrade the current localization system to work on maps with intersections and multi-direction paths.

# Bibliography

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016. 2.2, 8.2, 8.2

[2] Roberto Arroyo, Pablo F. Alcantarilla, Luis M. Bergasa, and Eduardo Romera. Towards life-long visual localization using an efficient matching of binary sequences from images. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6328–6335, 2015. doi: 10.1109/ICRA.2015.7140088. 2.3

[3] Houssem-Eddine Benseddik, Fabio Morbidi, and Guillaume Caron. Panoramis: An ultra-wide field of view image dataset for vision-based robot-motion estimation. *The International Journal of Robotics Research*, 39(9):1037–1051, 2020. 2.4

[4] Peter Biber and Tom Duckett. Experimental analysis of sample-based maps for long-term slam. *The International Journal of Robotics Research*, 28(1):20–33, 2009. 2.2

[5] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6): 1874–1890, 2021. doi: 10.1109/TRO.2021.3075644. 2.1

[6] Nicholas Carlevaris-Bianco, Arash K Ushani, and Ryan M Eustice. University of michigan north campus long-term vision and lidar dataset. *The International Journal of Robotics Research*, 35(9):1023–1035, 2016. 2.4

[7] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss. OverlapNet: Loop Closing for LiDAR-based SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020. 2.2

[8] Zetao Chen, Adam Jacobson, Niko Sünderhauf, Ben Upcroft, Lingqiao Liu, Chunhua Shen, Ian Reid, and Michael Milford. Deep learning features at scale for visual place recognition. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3223–3230, 2017. doi: 10.1109/ICRA.2017.7989366.

2.2, 8.2, 8.2

[9] Chih-Chung Chou and Cheng-Fu Chou. Efficient and accurate tightly-coupled visual-lidar slam. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–15, 2021. doi: 10.1109/TITS.2021.3130089. 2.1

[10] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008. 2.2

[11] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005. doi: 10.1109/CVPR.2005.177. 2.2, 8.2, 8.2

[12] Feras Dayoub and Tom Duckett. An adaptive appearance-based map for long-term topological localization of mobile robots. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3364–3369, 2008. doi: 10.1109/IROS.2008.4650701. 2.2

[13] Frank Dellaert and Chris Beall. Gtsam 4.0. *URL: https://bitbucket.org/gtborg/gtsam*, 2017. 5.5

[14] Peter Hansen and Brett Browning. Visual place recognition using hmm sequence matching. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4549–4555, 2014. doi: 10.1109/IROS.2014.6943207. 2.3

[15] Tuan Ho, Ioannis D. Schizas, K. R. Rao, and Madhukar Budagavi. 360-degree video stitching for dual-fisheye lens cameras based on rigid moving least squares. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 51–55, 2017. doi: 10.1109/ICIP.2017.8296241. 4.3

[16] Ahmad Khaliq, Shoaib Ehsan, Zetao Chen, Michael Milford, and Klaus McDonald-Maier. A holistic visual place recognition approach using lightweight cnns for significant viewpoint and appearance changes. *IEEE Transactions on Robotics*, 36(2):561–569, 2020. doi: 10.1109/TRO.2019.2956352. 2.2, 8.2, 8.2

[17] Kenji Koide, Jun Miura, Masashi Yokozuka, Shuji Oishi, and Atsuhiko Banno. Interactive 3d graph slam for map correction. *IEEE Robotics and Automation Letters*, 6(1):40–47, 2021. doi: 10.1109/LRA.2020.3028828. (document), 6.2, 6.3, 6.3, 6.4, 8.3.1

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`. 8.2, 8.2

[19] Yang Liu and Hong Zhang. Towards improving the efficiency of sequence-based slam. In *2013 IEEE International Conference on Mechatronics and Automation*, pages 1261–1266, 2013. doi: 10.1109/ICMA.2013.6618095. 2.3

[20] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J. Leonard, David Cox, Peter Corke, and Michael J. Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, 2016. doi: 10.1109/TRO.2015.2496823. 2.1

[21] Will Maddern, Michael Milford, and Gordon Wyeth. Cat-slam: probabilistic localisation and mapping using a continuous appearance-based trajectory. *The International Journal of Robotics Research*, 31(4):429–451, 2012. 2.2

[22] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017. 2.4

[23] Nate Merrill and Guoquan Huang. Lightweight unsupervised deep loop closure. *arXiv preprint arXiv:1805.07703*, 2018. 8.2, 8.2

[24] Michael J. Milford and Gordon. F. Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *2012 IEEE International Conference on Robotics and Automation*, pages 1643–1649, 2012. doi: 10.1109/ICRA.2012.6224623. 2.2, 2.3, 3, 3.1, 3.1.1, 3.1.2, 3.1.2, 3.1.3, 3.2, 3.2.1, 3.2.2, 3.2.4

[25] Timothy Morris, Feras Dayoub, Peter Corke, Gordon Wyeth, and Ben Upcroft. Multiple map hypotheses for planning and navigating in non-stationary environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2765–2770, 2014. doi: 10.1109/ICRA.2014.6907255. 2.2

[26] Michał R. Nowicki, Jan Wietrzykowski, and Piotr Skrzypczyński. Real-time visual place recognition for personal localization on a mobile device. *Wireless Personal Communications*, 97:213–244, 2017. 2.2

[27] Gaurav Pandey, James R McBride, and Ryan M Eustice. Ford campus vision and lidar data set. *The International Journal of Robotics Research*, 30(13):1543–1552, 2011. 2.4

[28] Edward Pepperell, Peter I. Corke, and Michael J. Milford. All-environment visual place recognition with smart. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1612–1618, 2014. doi: 10.1109/ICRA.2014.6907067. 2.2, 2.3

[29] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018. doi: 10.1109/TRO.2018.2853729. 2.1, 5.1, 5.5

[30] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using

moving least squares. In *ACM SIGGRAPH 2006 Papers*, pages 533–540. 2006. 4.3

[31] Sayem Mohammad Siam and Hong Zhang. Fast-seqslam: A fast appearance based place recognition algorithm. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5702–5708, 2017. doi: 10.1109/ICRA.2017.7989671. 2.3

[32] S Skrede. Nordland dataset, 2013. 2.4

[33] Ben Talbot, Sourav Garg, and Michael Milford. Openseqslam2.0: An open source toolbox for visual place recognition under changing conditions. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7758–7765, 2018. doi: 10.1109/IROS.2018.8593761. 2.3

[34] Mahdi Tavakoli, Jay Carriere, and Ali Torabi. Robotics, smart wearable technologies, and autonomous intelligent systems for healthcare during the covid-19 pandemic: An analysis of the state of the art and future vision. *Advanced Intelligent Systems*, 2(7):2000071, 2020. 1.1

[35] Akihiko Torii, Josef Sivic, Tomas Pajdla, and Masatoshi Okutomi. Visual place recognition with repetitive structures. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 883–890, 2013. 2.4

[36] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1808–1817, 2015. 2.4

[37] Konstantinos A. Tsintotas, Loukas Bampis, and Antonios Gasteratos. Doseqslam: Dynamic on-line sequence based loop closure detection algorithm for slam. In *2018 IEEE International Conference on Imaging Systems and Techniques (IST)*, pages 1–6, 2018. doi: 10.1109/IST.2018.8577113. 2.3

[38] Guang-Zhong Yang, Bradley J. Nelson, Robin R. Murphy, Howie Choset, Henrik Christensen, Steven H. Collins, Paolo Dario, Ken Goldberg, Koji Ikuta, Neil Jacobstein, Danica Kragic, Russell H. Taylor, and Marcia McNutt. Combating covid-19&#x2014;the role of robotics in managing public health and infectious diseases. *Science Robotics*, 5(40):eabb5589, 2020. doi: 10.1126/scirobotics.abb5589. URL `https://www.science.org/doi/abs/10.1126/scirobotics.abb5589`. 1.1

[39] Peng Yin, Rangaprasad Arun Srivatsan, Yin Chen, Xueqian Li, Hongda Zhang, Lingyun Xu, Lu Li, Zhenzhong Jia, Jianmin Ji, and Yuqing He. Mrs-vpr: a multi-resolution sampling based global visual place recognition method. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7137–7142, 2019. doi: 10.1109/ICRA.2019.8793853. 2.3, 3

[40] Peng Yin, Fuying Wang, Anton Egorov, Jiafan Hou, Zhenzhong Jia, and Jianda

Han. Fast sequence-matching enhanced viewpoint-invariant 3-d place recognition. *IEEE Transactions on Industrial Electronics*, 69(2):2127–2135, 2022. doi: 10.1109/TIE.2021.3057025. 2.3, 3

[41] Mubariz Zaffar, Shoaib Ehsan, Michael Milford, and Klaus McDonald-Maier. Cohog: A light-weight, compute-efficient, and training-free visual place recognition technique for changing environments. *IEEE Robotics and Automation Letters*, 5(2):1835–1842, 2020. doi: 10.1109/LRA.2020.2969917. 2.2, 5.2, 8, 8.1, 8.2, 8.2

[42] Mubariz Zaffar, Sourav Garg, Michael Milford, Julian Kooij, David Flynn, Klaus McDonald-Maier, and Shoaib Ehsan. Vpr-bench: An open-source visual place recognition evaluation framework with quantifiable viewpoint and appearance change. *International Journal of Computer Vision*, pages 1–39, 2021. 8.2, 8.3.4

[43] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, pages 1–9. Berkeley, CA, 2014. 5.1, 6.2, 6.3, 8.3.1

[44] Ji Zhang and Sanjiv Singh. Visual-lidar odometry and mapping: Low drift, robust, and fast. In *IEEE International Conference on Robotics and Automation(ICRA)*, Seattle, WA, May 2015. 2.1

[45] Shibo Zhao, Hengrui Zhang, Peng Wang, Lucas Nogueira, and Sebastian Scherer. Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8729–8736, 2021. doi: 10.1109/IROS51168.2021.9635862. 2.1